(1)




# FLIGHT CONTROL SYSTEM FOR THE CRCA USING A COMMAND GENERATOR TRACKER WITH PI FEEDBACK AND KALMAN FILTER VOLUME I

THESIS

Steven Spencer Payson
Captain, USAF

AFIT/GE/ENG/89M-6



DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/89M-6



# FLIGHT CONTROL SYSTEM FOR THE CRCA USING A COMMAND GENERATOR TRACKER WITH PI FEEDBACK AND KALMAN FILTER VOLUME I

THESIS

Steven Spencer Payson
Captain, USAF

AFIT/GE/ENG/89M-6

AFIT/GE/ENG/89M-6

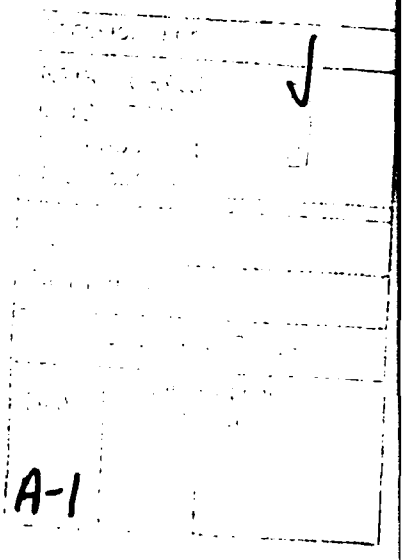
# FLIGHT CONTROL SYSTEM FOR THE CRCA USING A COMMAND GENERATOR TRACKER WITH PI FEEDBACK AND KALMAN FILTER VOLUME I

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Steven Spencer Payson, B.S.
Captain, USAF

March, 1989



## *Abstract*

This research develops an integrated software design package useful in the synthesis of CGT/PI/KF control systems, and uses this software package to design and evaluate a longitudinal flight control system for the Control Reconfigurable Combat Aircraft (CRCA). The software package, called CGTPIKF and built with $MATRIX_X$ commands, allows for the synthesis and evaluation of a Command Generator Tracker (CGT) which provides inputs to the system and acts as a precompensator, and a regulator with proportional plus integral (PI) feedback which forces the system outputs to mimic the model output. The software also allows the incorporation of a Kalman filter for estimation of the system states. Certainty equivalence can be invoked by adopting the LQG assumptions, thereby allowing the Kalman filter to be designed independently of the CGT/PI controller. The total CGT/PI/KF controller can then be evaluated and the design refined. CGTPIKF is an interactive, menu-driven CAD package which can be used in the development of any CGT/PI/KF control system, regardless of application.

A flight control system was designed for the CRCA air combat mode (ACM) entry using CGTPIKF. This control system was designed to force the aircraft to emulate a first order responce in pitch rate. The command model of the command generator tracker represented a first order pitch rate response with a rise time of .6 sec. Various weighting matrices were evaluated and refined in the development of the PI controller; the different controller designs were tested against the simulation containing various modelling errors, particularly failure conditions. The Kalman filter was later added, and the controller was again tested against the failure conditions. Loop Transmission Recovery (LTR) was successfully implemented to enhance robustness. The results confirm that a robust control system can be designed using the software package developed in this research.

*Acknowledgments*

I'd like to express my deepest thanks to Dr. Peter S. Maybeck for his concern, dedication, and patience. Without his guidance, this thesis effort would not have been the outstanding learning experience it turned out to be.

I would also like to thank Daryl Hammond and Kurt Neumann for helping me understand the CRCA. Dan Zambon provided invaluable assistance as the resident computer guru. Tom Kobylarz provided some valuable insights into MATRIX$_X$ . His parties, along with the efforts of Chuck Sokol and Dave Rizzo, helped me keep my perspective. To my climbing buddies, thanks for keeping me on the rock, and not letting go of the rope.

Finally, I'd like to thank Bob Dudley, my Little Father, for a ray of the sun source, and Tim Sakulich, for my sanity.

Steven Spencer Payson

*Table of Contents*

## *List of Figures*

# *List of Tables*

Table Page

# FLIGHT CONTROL SYSTEM FOR THE CRCA USING A COMMAND GENERATOR TRACKER WITH PI FEEDBACK AND KALMAN FILTER

## *I. Introduction*

### *1.1 Background*

The primary concern in the formulation of control systems is that the system perform with certain desired performance characteristics at nominal conditions, and that the controlled system be stable at off-nominal conditions. (This stability at off-nominal conditions is called *robustness.*) The introduction of unknown or unmodeled variations between the "real world" system and that which is mathematically modeled can greatly complicate the design of an effective control system. These variations can include disturbances which affect the system or the sensors which measure system response, variations of the actual system due to either operation at off-nominal conditions or failures of the system or sensors, and simplification of the mathematical model to reduce the complexity of the design problem [14]. Therefore, the design of a control system must in some way account for variations of the actual system from the design model used to represent it, while *simultaneously* forcing the system to respond in a desired manner.

In the development of control systems, two primary methods exist, frequency domain techniques and time domain approaches. Frequency domain techniques are well established and are the basis of classical control theory [3,4]. However, these techniques were developed for use with single-input, single-output (SISO) control problems, and when applied to multiple-input, multiple-output (MIMO)

systems, such as a flight control system for a modern aircraft, they can become tedious. On the other hand, a time domain technique which is inherently suited to MIMO control system design is LQG, which assumes a Linear system model, that a Quadratic cost is to be optimized, and that the system is driven by Gaussian noises. LQG design [5,11,13] takes in o account the stochastic nature of a problem; that is, that random occurrences can affect the system dynamics, and that the actual state of the system is never known perfectly, due to incomplete and noisy measurements.

One approach to designing an LQG controller is to separate the performance requirements of the closed-loop system from its desired stability characteristics. By implementing a command generator tracker (CGT) as a precompensator, which incorporates the performance characteristics desired of the system, with a proportional-plus-integral (PI) feedback loop, which handles stability of the system, an effective controller can be designed [11]. The CGT portion of the controller processes commanded inputs into system inputs by means of feedforward gains. The standard PI controller forces the system to be stable by imposing quadratic penalties on state or output perturbation deviations from zero. These quadratic penalties are implemented via feedforward and feedback gains. This is known as explicit model-following. An alternative approach to designing a PI controller is to impose a quadratic penalty on dynamics deviations from a model which has desired stability characteristics, thereby enhancing the robustness of the closed-loop system. This approach is called implicit model-following [14].

When the unrealistic assumption of full state feedback is removed (on which the CGT/PI controller is based), a Kalman filter is employed. The Kalman filter estimates the system states, taking into account the uncertainties of the system and the precision of the measurements available. Because the addition of the Kalman filter can degrade the robustness of the closed-loop system, Loop Transmission Recovery (LTR) is employed as a means of asymptotically recovering the robustness obtained by the deterministic CGT/PI controller [9].

## 1.2 Problem

The use of a computer aided design (CAD) package which allows the ready formulation of CGT/PI/KF control systems is critical. Floyd [5] designed a CAD package which allowed for the design and evaluation of a deterministic full-state feedback CGT/PI controller, and also the separate design and evaluation of a Kalman filter. Miller [14] subsequently built another CAD package which allowed for incorporation of the Kalman filter with the CGT/PI controller; however, the use of both CAD packages in concert with one another proved unwieldy and cumbersome. Those who used these software packages [7,9,16,17] found that an integrated CAD package would significantly streamline the design process.

The primary focus of this thesis effort was the integration of the functions performed by the two previous CAD packages. The new software is implemented on MATRIX$_X$ [10]. MATRIX$_X$ is a powerful CAD package which allows for matrix manipulation as well as classical and modern controller design. Hosting a program on MATRIX$_X$ which allows for the design and evaluation of CGT/PI/KF control systems would permit the use of all of *MATRIX$_X$*'s capabilities to the control system designer, not just those found in the specific CAD package itself. The resulting integrated software package, CGTPIKF, incorporates those key features of MATRIX$_X$ necessary for the design and evaluation of either the deterministic CGT/PI controller, the Kalman filter, or the composite CGT/PI/KF controller. This software, a user-oriented and interactive design tool, is highly modularized and thoroughly documented, thereby allowing the user to make problem-specific changes to the code which may increase the flexibility of the program and further enhance the ease of designing control systems.

To validate the above software package, a flight control system for the Control Reconfigurable Combat Aircraft (CRCA) [18] was designed. While this aircraft does not actually exist in a physical sense, detailed models of the aircraft at various points in the flight envelope and under several failure conditions do ex-

ist. The ability of both a CGT/PI and a CGT/PI/KF controller to enhance the handling qualities and stability robustness of the CRCA will be investigated. The applicability of implicit model-following compared to the standard (explicit model-following) PI controller, as a means of improving robustness in the face of failure conditions and unmodeled actuator dynamics, is addressed. The effect of actuator saturation nonlinearities, and the applicability of antiwindup compensation [11] to alleviate problems caused by these nonlinearities, will be evaluated. LTR's ability to improve system stability robustness in the face of failures will also be explored.

In addition to the flight control system designed in this thesis, concurrent thesis efforts [8,15] have also addressed flight controller designs using alternate design methods, for purposes of comparison and evaluation.

### *1.3 Sequence of Presentation*

While LQG theory is powerful and versatile, reader knowledge of the design techniques will not be assumed. Therefore, Chapter 2 presents the theoretical background required to understand how a CGT/PI/KF controller is designed. The use of implicit model-following and Loop Transmission Recovery are specifically addressed as two means of enhancing the robustness of the controller. Antiwindup compensation, which helps alleviate problems which may arise when PI controllers are used in conjunction with systems which have actuators that can saturate, is also discussed.

Chapter 3 provides background material on the CRCA, including the model which defines its nominal flight condition. Other physical characteristics of the aircraft are also presented. Specific failure conditions which will be evaluated against controllers are described.

Chapter 4 describes in detail the controller was development, including choices of design and command models. Several CGT/PI controllers are designed with various quadratic weights, and their performance is evaluated against the nominal flight

condition and various failure conditions. The controller is first evaluated against flight condition models which do not possess actuator dynamics, to aid in the later identification of design deficiencies which may be revealed by including those dynamics, which is investigated next. Actuator nonlinearities are introduced, and the effect of antiwindup compensation is presented. The Kalman filter is designed, and the CGT PI/KF controller is evaluated against all of the flight conditions which possess actuator dynamics.

Chapter 5 presents conclusions from the research performed, as well as recommendations for future research. Appendix A is a user's manual for CGTPIKF, a general purpose CAD package for use in the design of CGT/PI/KF control systems. This appendix includes a detailed description of how to design a control system using the software. An outline of the program is presented, along with an explanation of the many options available to the user. A list of the files required to run CGTPIKF is also presented. It should be noted that familiarity with $MATRIX_X$ is a prerequisite to using the software.

Appendix B (in Volume II) includes a program listing of CGTPIKF. The code is thoroughly documented, thereby allowing the interested user to delve into the software implementation of LQG theory. However, knowledge of algorithm coding is not required for use of the program. Rather, the source code is provided primarily to further user understanding of the specific implementation, and to facilitate software modifications, if desired. The high degree of modularity of CGTPIKF allows such modifications to be implemented easily and quickly.

# *II. Theoretical Development of LQG Controllers*

## *2.1 Introduction*

This chapter presents the theoretical development of LQG controllers. The basic assumptions behind an LQG control system are that there is a Linear system model, that a Quadratic cost is to be optimized, and that the system is driven by Gaussian noises. Inherent to the concept of an LQG controller is *certainty equivalence*. Certainty equivalence states that the design of an optimal stochastic controller can be divided into two parts, that of the deterministic full-state feedback controller and the design of an appropriate Kalman filter. The deterministic controller is independent of the uncertainty of the system; that is, the state $\mathbf{x}(t_i)$ of the system is known perfectly for all times $t_i$ and there is assumed to be no noise driving state dynamics between these sample times [11]. The Kalman filter incorporates all of the uncertainty of the system, and is designed independent of the deterministic controller. From the Kalman filter's point of view, a control problem does not exist (although the filter is informed of the deterministic control to be applied to the system over each sample period). These two parts of the controller are designed separately, greatly reducing the complexity of the design problem, and are then cascaded together to form the optimal stochastic controller.

The deterministic part of the controller in this research consists of the Command Generator Tracker (CGT) cascaded with a proportional plus integral (PI) controller. The command generator incorporates all of the system performance requirements, such as desired handling qualities of an aircraft, by generating an optimal trajectory that the plant (the system being controlled) should emulate. This is discussed in Section 2.2. The PI controller provides guaranteed system stability at design conditions (under nonrestrictive stabilizability and detectability conditions on the system model), and also provides type-1 feedback characteristics. In addition, the PI controller can be designed using *implicit model-following*, which

forces the plant to emulate the transient characteristics of a system model with desired stability and robustness characteristics. *Robustness* refers to system stability in the face of unmodeled system characteristics, failures, and other deviations from assumed design conditions. Implicit model-following can greatly enhance robustness. Sections 2.3 and 2.4 present the standard, or explicit model-following, form of the PI controller to be used in conjunction with a command generator tracker, while Section 2.6 addresses implicit model-following as a means of establishing the gains in this PI controller.

The Kalman filter is both well established and thoroughly documented, so only a cursory development is presented in Section 2.5. Interested readers are referred to [12] for more details.

When the unrealistic assumption of full-state feedback is removed by introducing the Kalman filter in cascade with the deterministic full-state feedback controller, all stability robustness guarantees are lost. Loop Transfer (or Transmission) Recovery (LTR) is a method which allows the designer to recover the stability robustness characteristics of the full-state feedback controller asymptotically. This is presented in Section 2.7.

The structure of the closed-loop CGT/PI/KF controller is given in Figure 2.1. The command inputs are fed through the command model to generate model states corresponding to the reference trajectory. These model states, along with disturbance states or state estimates, are then multiplied by the gains generated in the CGT control law formulation. The PI controller incorporates the system inputs and states (or state estimates) as well as the command inputs and command model states, and its purpose is to force the plant to track the desired trajectory closely. The Kalman filter uses system measurements and inputs to estimate system and disturbance states, and these best estimates are used in place of the (inaccessible) real world states.

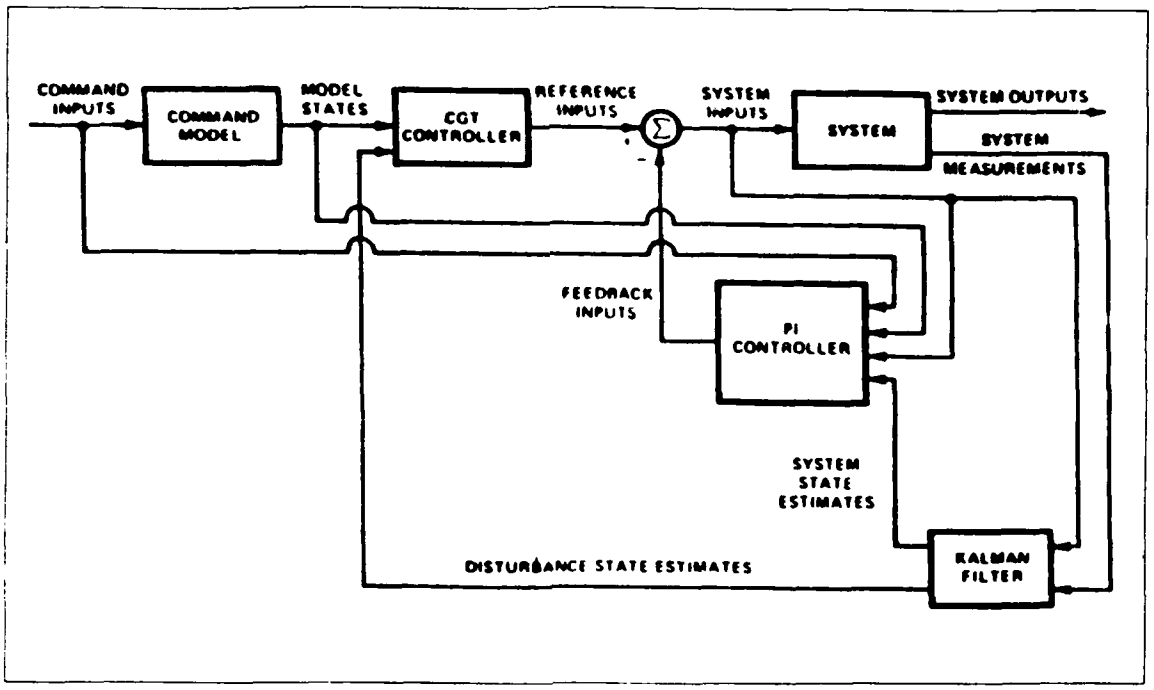



Figure 2.1. The CGT/PI/KF Controller

## *2.2 Command Generator Trackers*

Assume that a system is required to follow a certain trajectory despite disturbances which may be present. For example, this trajectory could exhibit desired handling qualities that an aircraft should track. The desired trajectory could be modeled as the output of a shaping filter driven by a commanded input. This shaping filter can be considered to be a command generator, represented by a command model. The overall control law that tries to force the actual plant to track the command model is the Command Generator Tracker (CGT). Command generator tracking allows a system to track commanded inputs with desired response characteristics while simultaneously rejecting disturbances.

Assume that a system is modeled by the discrete-time equations

$$\mathbf{x}(t_{i+1}) = \mathbf{\Phi x}(t_i) + \mathbf{B u}(t_i) + \mathbf{E_x n_d}(t_i) + \mathbf{w_d}(t_i) \tag{2.1}$$

$$\mathbf{y}(t_i) = \mathbf{C x}(t_i) + \mathbf{D u}(t_i) + \mathbf{E_y n_d}(t_i) \tag{2.2}$$

where $\mathbf{w_d}(t_i)$ is a zero-mean white Gaussian noise sequence of covariance $\mathbf{Q_d}$, and $\mathbf{n_d}$ is a time correlated noise sequence (a disturbance) modeled by

$$\mathbf{n_d}(t_{i+1}) = \mathbf{\Phi_n n_d}(t_i) + \mathbf{G_n w_{dn}}(t_i) \tag{2.3}$$

with $\mathbf{w_{dn}}(t_i)$ a zero-mean white Gaussian noise sequence of covariance $\mathbf{Q_{dn}}$, usually assumed independent of $\mathbf{w_d}(t_i)$.

The goal of the CGT controller is to force the actual plant outputs $\mathbf{y}(t_i)$ to follow the output of a command generator model

$$\mathbf{x_m}(t_{i+1}) = \mathbf{\Phi_m x_m}(t_i) + \mathbf{B_m u_m} \tag{2.4}$$

$$\mathbf{y_m}(t_i) = \mathbf{C_m x_m}(t_i) + \mathbf{D_m u_m} \tag{2.5}$$

where $\mathbf{y_m}$ and $\mathbf{y}$ must be of the same dimension p. For the purposes of this thesis, the model input $\mathbf{u_m}$ is considered to be time invariant. This is a valid approximation based on flight control sampling rates, where the pilot's commanded input is slowly varying compared to a 40 Hz or higher sampling rate.

As previously stated, the objective of the CGT controller is to track the output of the command model perfectly, that is, to zero out the error

$$\begin{aligned} \mathbf{e}(t_i) &= \mathbf{y}(t_i) - \mathbf{y_m}(t_i) \\ &= \begin{bmatrix} \mathbf{C} & \mathbf{D} & \mathbf{E_y} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t_i) \\ \mathbf{u}(t_i) \\ \mathbf{n_d}(t_i) \end{bmatrix} - \begin{bmatrix} \mathbf{C_m} & \mathbf{D_m} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \end{bmatrix} \end{aligned} \tag{2.6}$$

When this error is zero, the states and controls are said to be tracking the ideal plant trajectory ([11], p153), which is defined for convenience in constructing the CGT law but will not be used in the actual implementation. The ideal trajectory must satisfy the system dynamics

$$\mathbf{x_I}(t_{i+1}) = \mathbf{\Phi x_I}(t_i) + \mathbf{B u_I}(t_i) + \mathbf{E_x n_d}(t_i) \tag{2.7}$$

Where the $\Phi$, $\mathbf{B}$ and $\mathbf{E_x}$ matrices are the same as in Equation(2.1). Also, the ideal plant trajectory must cause the command error to be zero, so

$$\begin{bmatrix} \mathbf{C} & \mathbf{D_y} & \mathbf{E_y} \end{bmatrix} \begin{bmatrix} \mathbf{x_I}(t_i) \\ \mathbf{u_I}(t_i) \\ \mathbf{n_d}(t_i) \end{bmatrix} = \begin{bmatrix} \mathbf{C_m} & \mathbf{D_m} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \end{bmatrix} \tag{2.8}$$

In addition to the above requirements, the ideal plant response must be a linear combination of model states and inputs, and of the disturbance state. This produces

$$\begin{bmatrix} \mathbf{x_I}(t_i) \\ \mathbf{u_I}(t_i) \end{bmatrix} = \begin{bmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \mathbf{A_{13}} \\ \mathbf{A_{21}} & \mathbf{A_{22}} & \mathbf{A_{23}} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \\ \mathbf{n_d}(t_i) \end{bmatrix} \tag{2.9}$$

The solution to the equations that the constant matrices $\mathbf{A_{11}}$ through $\mathbf{A_{23}}$ must satisfy provides the feedforward gains which are the solution to the open-loop CGT problem. Setting up the problem, we first examine the ideal plant state equation

$$\begin{bmatrix} \mathbf{x_I}(t_{i+1}) - \mathbf{x_I}(t_i) \\ \mathbf{y_I}(t_i) \end{bmatrix} = \begin{bmatrix} (\Phi - \mathbf{I}) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x_I}(t_i) \\ \mathbf{u_I}(t_i) \end{bmatrix} + \begin{bmatrix} \mathbf{E_x} \\ \mathbf{E_y} \end{bmatrix} \mathbf{n_d}(t_i) \tag{2.10}$$

Substituting Equation (2.9) into (2.10) yields

$$\begin{bmatrix} \mathbf{x_I}(t_{i+1}) - \mathbf{x_I}(t_i) \\ \mathbf{y_I}(t_i) \end{bmatrix} = \begin{bmatrix} (\Phi - \mathbf{I}) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \mathbf{A_{13}} \\ \mathbf{A_{21}} & \mathbf{A_{22}} & \mathbf{A_{23}} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \\ \mathbf{n_d}(t_i) \end{bmatrix} + \begin{bmatrix} \mathbf{E_x} \\ \mathbf{E_y} \end{bmatrix} \mathbf{n_d}(t_i) \tag{2.11}$$

Now we seek a second expression for the left hand side of the above equation, which will allow us to solve for $\mathbf{A_{11}}$ through $\mathbf{A_{23}}$ by equating the two expressions. From

Equation(2.9),

$$[\mathbf{x_I}(t_{i+1}) - \mathbf{x_I}(t_i)] = \begin{bmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \mathbf{A_{13}} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_{i+1}) - \mathbf{x_m}(t_i) \\ \mathbf{u_m} - \mathbf{u_m} \\ \mathbf{n_d}(t_{i+1}) - \mathbf{n_d}(t_i) \end{bmatrix} \tag{2.12}$$

and then (2.3) and (2.4) can be used to write

$$[\mathbf{x_I}(t_{i+1}) - \mathbf{x_I}(t_i)] = \begin{bmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \mathbf{A_{13}} \end{bmatrix} \begin{bmatrix} (\Phi_\mathbf{m} - \mathbf{I}) & \mathbf{B_m} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\Phi_\mathbf{n} - \mathbf{I}) \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \\ \mathbf{n_d}(t_i) \end{bmatrix} \tag{2.13}$$

Also, since $\mathbf{y_I}(t_i) = \mathbf{y_m}(t_i)$,

$$\mathbf{y_I}(t_i) = \begin{bmatrix} \mathbf{C_m} & \mathbf{D_m} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \end{bmatrix} \tag{2.14}$$

Combining the above results yields

$$\begin{bmatrix} \mathbf{x_I}(t_{i+1}) - \mathbf{x_I}(t_i) \\ \mathbf{y_I}(t_i) \end{bmatrix} = \begin{bmatrix} \mathbf{A_{11}}(\Phi_\mathbf{m} - \mathbf{I}) & \mathbf{A_{11}B_m} & \mathbf{A_{13}}(\Phi_\mathbf{n} - \mathbf{I}) \\ \mathbf{C_m} & \mathbf{D_m} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \\ \mathbf{n_d}(t_i) \end{bmatrix} \tag{2.15}$$

Equating (2.11) and (2.15) yields

$$\begin{bmatrix} (\Phi - \mathbf{I}) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \mathbf{A_{13}} \\ \mathbf{A_{21}} & \mathbf{A_{22}} & \mathbf{A_{23}} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \\ \mathbf{n_d}(t_i) \end{bmatrix} + \begin{bmatrix} \mathbf{E_x} \\ \mathbf{E_y} \end{bmatrix} \mathbf{n_d}(t_i)$$

$$= \begin{bmatrix} \mathbf{A_{11}}(\Phi_\mathbf{m} - \mathbf{I}) & \mathbf{A_{11}B_m} & \mathbf{A_{13}}(\Phi_\mathbf{n} - \mathbf{I}) \\ \mathbf{C_m} & \mathbf{D_m} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x_m}(t_i) \\ \mathbf{u_m} \\ \mathbf{n_d}(t_i) \end{bmatrix} \tag{2.16}$$

Letting

$$\begin{bmatrix} (\Phi - \mathbf{I}) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{\Pi_{11}} & \mathbf{\Pi_{12}} \\ \mathbf{\Pi_{21}} & \mathbf{\Pi_{22}} \end{bmatrix} \tag{2.17}$$

we get the solution to the constant $\mathbf{A_{ij}}$ matrices as

$$\begin{bmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \mathbf{A_{13}} \\ \mathbf{A_{21}} & \mathbf{A_{22}} & \mathbf{A_{23}} \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{\Pi_{11}} & \mathbf{\Pi_{12}} \\ \mathbf{\Pi_{21}} & \mathbf{\Pi_{22}} \end{bmatrix} \begin{bmatrix} \mathbf{A_{11}}(\mathbf{\Phi_m} - \mathbf{I}) & \mathbf{A_{11}B_m} & \mathbf{A_{13}}(\mathbf{\Phi_n} - \mathbf{I}) - \mathbf{E_x} \\ \mathbf{C_m} & \mathbf{D_m} & -\mathbf{E_y} \end{bmatrix} \tag{2.18}$$

The partitioned equations to solve are now

$$\mathbf{A_{11}} = \mathbf{\Pi_{11}A_{11}}(\mathbf{\Phi_m} - \mathbf{I}) + \mathbf{\Pi_{12}C_m} \tag{2.19}$$

$$\mathbf{A_{12}} = \mathbf{\Pi_{11}A_{11}B_m} + \mathbf{\Pi_{12}D_m} \tag{2.20}$$

$$\mathbf{A_{13}} = \mathbf{\Pi_{11}A_{13}}(\mathbf{\Phi_n} - \mathbf{I}) - \mathbf{\Pi_{11}E_x} - \mathbf{\Pi_{12}E_y} \tag{2.21}$$

$$\mathbf{A_{21}} = \mathbf{\Pi_{21}A_{11}}(\mathbf{\Phi_m} - \mathbf{I}) + \mathbf{\Pi_{22}C_m} \tag{2.22}$$

$$\mathbf{A_{22}} = \mathbf{\Pi_{21}A_{11}B_m} + \mathbf{\Pi_{22}D_m} \tag{2.23}$$

$$\mathbf{A_{23}} = \mathbf{\Pi_{21}A_{13}}(\mathbf{\Phi_n} - \mathbf{I}) - \mathbf{\Pi_{21}E_x} - \mathbf{\Pi_{22}E_y} \tag{2.24}$$

$\mathbf{A_{11}}$ and $\mathbf{A_{13}}$ are the nontrivial equations, while the other $\mathbf{A_{ij}}$ matrix solutions are straightforward. $\mathbf{A_{11}}$ and $\mathbf{A_{13}}$ are of the form $\mathbf{X} = \mathbf{AXB} + \mathbf{C}$. A solution for $\mathbf{X}$ exists [2], provided the eigenvalues ($\lambda$) of $\mathbf{A}$ and $\mathbf{B}$ satisfy $\lambda_{\mathbf{A}_i}\lambda_{\mathbf{B}_j} \neq 1$, for all i and j. The resulting open-loop command generator control law is

$$\mathbf{u_I}(t_i) = \mathbf{A_{21}}\mathbf{x_m}(t_i) + \mathbf{A_{22}u_m} + \mathbf{A_{23}n_d}(t_i) \tag{2.25}$$

The structure of the open-loop CGT control law is given in Figure 2.2. This figure graphically illustrates how the command generator tracker works, and that it is indeed an open-loop controller.

An open-loop CGT control law is limited in application to stable systems. If the plant to be controlled is unstable or marginally stable, or if uncertainties or unmodeled disturbances exist, a closed-loop CGT controller is required. The synthesis of the feedback path, in the form of a PI controller, is desired. Such a controller can be synthesized by linear/quadratic (LQ) regulator methods applied

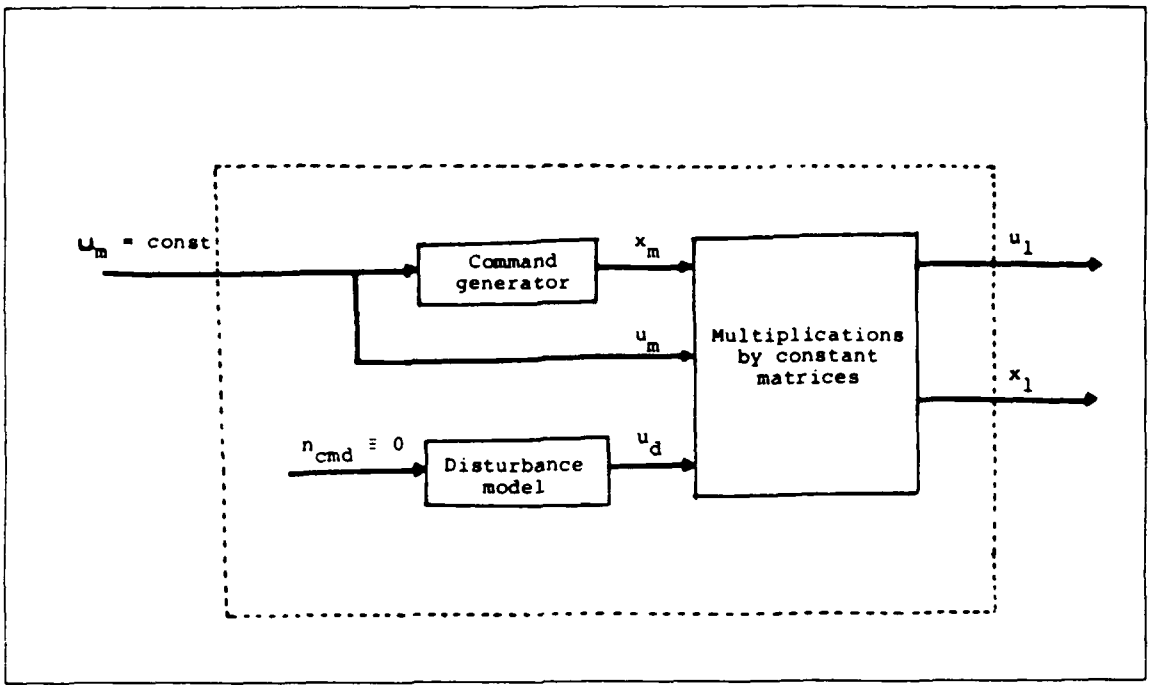



Figure 2.2. Open-loop command generator tracker

to an augmented plant model. Therefore, LQ regulator synthesis will be discussed in general, followed by explicit synthesis of PI controllers.

### *2.9 Synthesis of LQ Regulators*

Given a linear deterministic discrete-time state equation,

$$\mathbf{x}(t_{i+1}) = \mathbf{\Phi x}(t_i) + \mathbf{B u}(t_i) \tag{2.26}$$

$$\mathbf{y}(t_i) = \mathbf{C x}(t_i) + \mathbf{D u}(t_i) \tag{2.27}$$

where $\mathbf{\Phi}$ is the time invariant state transition matrix, $\mathbf{B}$ is the discrete-time, time invariant control input matrix, $\mathbf{C}$ is the time invariant output matrix, and $\mathbf{y}$ is the vector of controlled outputs, we desire a feedback control law that will make the system behave in a desired way. One way of generating this control law is to determine the optimal control function $\mathbf{u}^*$ which minimizes a scalar quadratic cost function

$$\mathbf{J} = \sum_{i+1}^{N} 1/2 \left[\mathbf{y}^T(t_i)\mathbf{Y}\mathbf{y}(t_i) + \mathbf{u}^T(t_i)\mathbf{U_M}\mathbf{u}(t_i)\right] \tag{2.28}$$

with $\mathbf{Y}$ and $\mathbf{U_M}$ the constant weighting matrices for deviations in magnitude of $\mathbf{y}$ and $\mathbf{u}$, respectively. A weighting term $\mathbf{Y_f}$ associated with the terminal output deviations from zero ($\mathbf{y}(t_{N+1})$) should generally be included if $N$ is finite, that is, if terminal transients in controller gains must be considered. This thesis, however, will assume that $N \to \infty$. Note that, when considering the stochastic nature of a problem, the expected value $\mathbf{E}\{\cdot\}$ of Equation (2.28) is the appropriate cost function for an optimal stochastic controller designed via LQG synthesis.

The above cost function allows a quadratic penalty to be assigned to the magnitude of output ($\mathbf{y}$) deviations from zero and to the controls ($\mathbf{u}$) expended. $\mathbf{Y}$ is a p-by-p symmetric positive definite matrix, and $\mathbf{U}$ is an r-by-r symmetric positive definite matrix, where p is the number of controlled outputs, and r is the number of control inputs. The cost function can be changed to a form that weights state deviations

$$\mathbf{J} = \sum_{i=1}^{N} 1/2 \left[\mathbf{x}^T(t_i)\mathbf{X}\mathbf{x}(t_i) + \mathbf{u}^T(t_i)\mathbf{U}\mathbf{u}(t_i) + 2\mathbf{x}^T(t_i)\mathbf{S}\mathbf{u}(t_i)\right] \tag{2.29}$$

where

$$\mathbf{X} = \mathbf{C}^T\mathbf{Y}\mathbf{C} \tag{2.30}$$

$$\mathbf{S} = \mathbf{C}^T\mathbf{Y}\mathbf{D} \tag{2.31}$$

$$\mathbf{U} = \mathbf{U_M} + \mathbf{D}^T\mathbf{Y}\mathbf{D} \tag{2.32}$$

$\mathbf{X}$ is thus an n-by-n (number of states), positive semidefinite weighting matrix. The $\mathbf{S}$ matrix allows quadratic penalties to be placed on rates of change of outputs and/or states, or on any variables which are expressed as linear combinations of $\mathbf{x}$ and $\mathbf{u}$ [13]. $\mathbf{S}$ also appears when continuous cost weighting matrices are discretized for the design of digital control systems [11], thus necessitating the objective of exerting desirable control influence on the states across sample periods. The cost minimizing controller is then

$$\mathbf{u}^*(t_i) = -\mathbf{G}_c^*\mathbf{x}(t_i) \tag{2.33}$$

where $\mathbf{G}_c^*$ is the constant controller gain, found from the equations

$$\mathbf{G}_c^* = \left[\mathbf{U} + \mathbf{B}^T\mathbf{K}_c\mathbf{B}\right]^{-1}\left[\mathbf{B}^T\mathbf{K}_c\mathbf{\Phi} + \mathbf{S}^T\right] \tag{2.34}$$

$$\mathbf{K}_c = \mathbf{X} + \mathbf{\Phi}^T\mathbf{K}_c\mathbf{\Phi} - \left[\mathbf{B}^T\mathbf{K}_c\mathbf{\Phi} + \mathbf{S}^T\right]^T\mathbf{G}_c^* \tag{2.35}$$

$\mathbf{K}_c$ is the algebraic Riccati equation solution.

Recall that, by certainty equivalence, the same $\mathbf{G}_c^*$ will be used as a multiplier of $\hat{\mathbf{x}}(t_i^+)$ (the estimated state value found in the Kalman filter), instead of a multiplier of the real-world plant state, $\mathbf{x}(t_i)$.

## *2.4 Synthesis of PI Controllers Via LQ Methods*

The LQ regulator presented in the previous section is somewhat limited in application. If the desired output of the system is to maintain a constant value $\mathbf{y_m}$ with zero steady state error, an LQ regulator for perturbations from $\mathbf{y_m}$ will drive the perturbation to zero only if there are no modelling errors in the actual physical system. Additionally, if there are any constant disturbances affecting the system, or if there is an apparent constant disturbance caused by omission of higher order terms in the formation of linear perturbation system models (to be discussed later in this section), then the LQ regulator will again be insufficient for our needs, because of the type-0 control system properties [4]. From classical control theory, however, it is expected that by adding integral control to the proportional control derived in the previous section, we can overcome these difficulties and obtain the desired type-1 system response.

Augmenting the original system states with a specific additional set of dynamic variables attains the desired integral control. These additional variables are the control differences (or pseudorates) $\Delta\mathbf{u}(t_i)$ such that

$$\Delta\mathbf{u}(t_i) = \mathbf{u}(t_{i+1}) - \mathbf{u}(t_i) \tag{2.36}$$

The dynamic variables can also be the pseudointegral (or summation) of the regulation error, $\mathbf{q}(t_i) = \mathbf{q}(t_{i-1}) + (\mathbf{y}(t_{i-1}) - \mathbf{y_m})$ [11], but that will not be pursued in this thesis.

There are two possible implementation forms that can be used for discrete-time control laws: position and incremental. In the position form, the control law is specified in terms of the current system state, as in Equation (2.33) for the regulator. This form is very useful for gaining insight into the PI controller function, but will not be used for the actual implementation of the control law in this thesis. Rather, the incremental form will be used. The incremental form uses changes in states and control inputs to determine incremental command values to be added to the previous commands. An incremental type regulator is of the form

$$\mathbf{u}^*(t_i) = \mathbf{u}^*(t_{i-1}) - \mathbf{G}_\mathbf{c}^* [\mathbf{x}(t_i) - \mathbf{x}(t_{i-1})] \tag{2.37}$$

Both the position and incremental forms have the same basic input/output characteristics. The incremental form controller, however, is preferable because initial conditions are not needed for the controller states, as in the position form. Also, for applications involving nonlinear systems (i.e., *real world* systems), the incremental form makes relinearization about nominal values, as well as anti-windup compensation (to be discussed later), easier to implement [11:p. 149] [17:p. 27].

Consider a nominal control $\mathbf{u_o}$ needed to maintain the system described by Equation (2.1) in a non-zero equilibrium condition, such that $\mathbf{y} = \mathbf{y_m}$. (Note that $\mathbf{y_m}$ is not necessarily the command model's output vector from Equation (2.5)). It follows that, since the defining matrices remain constant, the nominal control is found as the solution to [11:p122]

$$\mathbf{x_o} = \mathbf{\Phi x_o} + \mathbf{B u_o} \tag{2.38}$$

$$\mathbf{y_m} = \mathbf{C x_o} + \mathbf{D u_o} \tag{2.39}$$

In matrix form this becomes

$$\begin{bmatrix} \Phi - \mathbf{I} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x_o} \\ \mathbf{u_o} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{y_m} \end{bmatrix} \tag{2.40}$$

The solution for $\mathbf{x_o}$ and $\mathbf{u_o}$ is thus

$$\begin{bmatrix} \mathbf{x_o} \\ \mathbf{u_o} \end{bmatrix} = \begin{bmatrix} \Phi - \mathbf{I} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{y_m} \end{bmatrix} \tag{2.41}$$

The augmented matrix inverse can be partitioned as before

$$\begin{bmatrix} \Phi - \mathbf{I} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{\Pi}_{11} & \mathbf{\Pi}_{12} \\ \mathbf{\Pi}_{21} & \mathbf{\Pi}_{22} \end{bmatrix} \tag{2.42}$$

For this development, it is assumed that the number of controls equals the number of controlled outputs. If this is not the case, pseudoinverse techniques may be used to invert the above matrix ([11],p123).

We can now define perturbation variables, which we will eventually try to regulate to zero. The perturbation variables are

$$\delta \mathbf{x}(t_i) = \mathbf{x}(t_i) - \mathbf{x_o} \tag{2.43}$$

$$\delta \mathbf{u}(t_i) = \mathbf{u}(t_i) - \mathbf{u_o} \tag{2.44}$$

$$\delta \mathbf{y}(t_i) = \mathbf{y}(t_i) - \mathbf{y_m} \tag{2.45}$$

To attain the desired integral feedback, we augment the state equations with the control differences. The difference in $\delta \mathbf{u}$ from one sample time to the next is

$$\delta \mathbf{u}(t_{i+1}) - \delta \mathbf{u}(t_i) = (\mathbf{u}(t_{i+1}) - \mathbf{u_o}) - (\mathbf{u}(t_i) - \mathbf{u_o})) \tag{2.46}$$

which can be expressed as

$$\delta \mathbf{u}(t_{i+1}) = \delta \mathbf{u}(t_i) + (\mathbf{u}(t_{i+1}) - \mathbf{u}(t_i)) \tag{2.47}$$

Recalling Equation (2.36), $\delta \mathbf{u}(t_{i+1})$ becomes

$$\delta \mathbf{u}(t_{i+1}) = \delta \mathbf{u}(t_i) + \Delta \mathbf{u}(t_i) \tag{2.48}$$

We can then write the augmented perturbation state space equation as

$$\begin{bmatrix} \delta \mathbf{x}(t_{i+1}) \\ \delta \mathbf{u}(\mathrm{t_{i+1}}) \end{bmatrix} = \begin{bmatrix} \Phi & \mathbf{B} \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x}(t_i) \\ \delta \mathbf{u}(t_i) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \Delta \mathbf{u}(t_i) \tag{2.49}$$

The optimal control law for the system of the above equation is found from the discrete quadratic cost

$$\mathbf{J} = 1/2 \sum_{i=0}^{N} \begin{bmatrix} \delta \mathbf{x}(t_i) \\ \delta \mathbf{u}(t_i) \\ \Delta \mathbf{u}(t_i) \end{bmatrix}^T \begin{bmatrix} \mathbf{X_{11}} & \mathbf{X_{12}} & \mathbf{S_1} \\ \mathbf{X_{12}}^T & \mathbf{X_{22}} & \mathbf{S_2} \\ \mathbf{S_1}^T & \mathbf{S_2}^T & \mathbf{U_R} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x}(t_i) \\ \delta \mathbf{u}(t_i) \\ \Delta \mathbf{u}(t_i) \end{bmatrix} \tag{2.50}$$

where $N \rightarrow \infty$. The values of the weighting matrices are found by

$$\mathbf{X_{11}} = \mathbf{C}^T \mathbf{Y} \mathbf{C} \tag{2.51}$$

$$\mathbf{X_{22}} = \mathbf{U_M} + \mathbf{D}^T \mathbf{Y} \mathbf{D} \tag{2.52}$$

$$\mathbf{X_{12}} = \mathbf{C}^T \mathbf{Y} \mathbf{D} \tag{2.53}$$

where $\mathbf{Y}$ and $\mathbf{U_M}$ are as defined in Equation (2.28). The $\mathbf{X_{11}}$ term therefore weights state trajectory deviations from the nominal and $\mathbf{X_{22}}$ weights control magnitudes. The $\mathbf{U_R}$ term weights control pseudorates, which we were not able to do in the LQ regulator problem of the previous section.

The upper partition of the weighting matrix in Equation (2.50) comprised of the discrete-time weighting matrices $\mathbf{X_{ii}}$ must be positive semidefinite, while $\mathbf{U_R}$ is positive definite. Furthermore, the entire augmented matrix in Equation (2.50) must be positive semidefinite.

Solving for the optimal control law that minimizes the above quadratic cost yields

$$\Delta \mathbf{u}^*(t_i) = -\mathbf{G_{c1}^*} \delta \mathbf{x}(t_i) - \mathbf{G_{c2}^*} \delta \mathbf{u}(ti) \tag{2.54}$$

where the gains $\mathbf{G}_c^*$ can be solved in the same manner as in Section 2.3.

Although Equation (2.54) represents an optimal perturbation regulator capable of regulating state and control magnitudes in addition to control rates, it does not yet possess the integral action we have been seeking. To attain the type-1 property, we will incorporate a signal proportional to the regulation error, $\mathbf{y_m} - \mathbf{y}(t_i)$, which is $-\delta\mathbf{y}(t_i)$. The optimal control signal constructed in terms of perturbation variables becomes

$$\begin{aligned} \delta\mathbf{u}(t_{i+1}) &= \delta\mathbf{u}(t_i) - \mathbf{K_x}[\delta\mathbf{x}(t_{i+1}) - \delta\mathbf{x}(t_i)] + \mathbf{K_z}[-\delta\mathbf{y}(t_i)] \\ &= \delta\mathbf{u}(t_i) - \mathbf{K_x}[\delta\mathbf{x}(t_{i+1}) - \delta\mathbf{x}(t_i)] - \mathbf{K_z}[\mathbf{C}\delta\mathbf{x}(t_i) + \mathbf{D}\delta\mathbf{u}(t_i)] \end{aligned} \quad (2.55)$$

The upper partition of the augmented perturbation state equation (2.49) can be written as

$$[\delta\mathbf{x}(t_{i+1}) - \delta\mathbf{x}(t_i)] = [\mathbf{\Phi} - \mathbf{I}]\,\delta\mathbf{x}(t_i) + \mathbf{B}\delta\mathbf{u}(t_i) \quad (2.56)$$

Combining the previous two equations and writing in matrix form yields

$$\delta\mathbf{u}(t_{i+1}) - \delta\mathbf{u}(t_i) = -\begin{bmatrix} \mathbf{K_x} & \mathbf{K_z} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi} - \mathbf{I} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}(t_i) \\ \delta\mathbf{u}(t_i) \end{bmatrix} \quad (2.57)$$

Observing that the left hand side of the above equation is $\Delta\mathbf{u}$, Equations (2.42) and (2.54) can be used to show that $\mathbf{K_x}$ and $\mathbf{K_z}$ are evaluated as

$$\mathbf{K_x} = \mathbf{G}_{c1}^*\mathbf{\Pi}_{11} + \mathbf{G}_{c2}^*\mathbf{\Pi}_{21} \quad (2.58)$$

$$\mathbf{K_z} = \mathbf{G}_{c1}^*\mathbf{\Pi}_{12} + \mathbf{G}_{c2}^*\mathbf{\Pi}_{22} \quad (2.59)$$

Thus, once $\mathbf{G}_c^*$ is established via solution to the Riccati equation for the augmented system, $\mathbf{K_x}$ and $\mathbf{K_z}$ can be computed.

The final form of the PI control law is

$$\mathbf{u}^*(t_i) = \mathbf{u}^*(t_{i-1}) - \mathbf{K_x}[\mathbf{x}(t_i) - \mathbf{x}(t_{i-1})] + \mathbf{K_z}[\mathbf{y_m}(t_{i-1}) - \mathbf{y}(t_{i-1})] \quad (2.60)$$

When combined into a closed-loop CGT/PI controller, the final incremental form CGT/PI control law becomes

$$\begin{aligned}
\mathbf{u}(t_i) &= \mathbf{u}(t_{i-1}) - \mathbf{K_x}\left[\mathbf{x}(t_i) - \mathbf{x}(t_{i-1})\right] \\
&+ \mathbf{K_z}\left\{\begin{bmatrix} \mathbf{C_m} & \mathbf{D_m} \end{bmatrix}\begin{bmatrix} \mathbf{x_m}(t_{i-1}) \\ \mathbf{u_m}(t_i) \end{bmatrix} - \begin{bmatrix} \mathbf{C} & \mathbf{D} \end{bmatrix}\begin{bmatrix} \mathbf{x}(t_{i-1}) \\ \mathbf{u}(t_{i-1}) \end{bmatrix}\right\} \\
&+ \left[\mathbf{K_x A_{11}} + \mathbf{A_{21}}\right]\left[\mathbf{x_m}(t_i) - \mathbf{x_m}(t_{i-1})\right] \\
&+ \left[\mathbf{K_x A_{13}} + \mathbf{A_{23}}\right]\left[\mathbf{n_d}(t_i) - \mathbf{n_d}(t_{i-1})\right] \qquad (2.61)
\end{aligned}$$

The time argument on $\mathbf{u_m}$ is not $t_{i-1}$ because, whenever $\mathbf{u_m}$ changes, we return to time $t_{i-1}$ and all variables which can be controlled are restarted, assuming that $\mathbf{u_m}(t_{i-1})$ actually equals $\mathbf{u_m}$ [11]. This "restart" mechanism also reduces the dynamic lag caused by the command generator model, since it speeds up its response by a full sample period.

When using a PI controller, a phenomenon known as *windup* can occur [11] in systems with actuators which can saturate, that is, actuators with hard limits. If a large change in desired system setpoint is commanded, the proportional channel of the controller may cause the actuators to reach their limits. The integration characteristic of the controller will integrate large errors, eventually reaching a commanded control level which by itself would also cause saturation. When the tracking error signal begins to decrease, the proportional signal also decreases, but the output of the integrator will remain at a saturation level until after the error signal has *changed sign*. This is known as windup, and can result in large system overshoots and unsatisfactory performance.

One antiwindup compensation scheme which is easily implemented with an incremental control law consists of passing the optimal control through a limiter which would preclude the actuators from being commanded to attain values outside their saturation limits (Figure 2.3). This limited command input replaces $\mathbf{u}(t_{i-1})$ of the incremental control law.

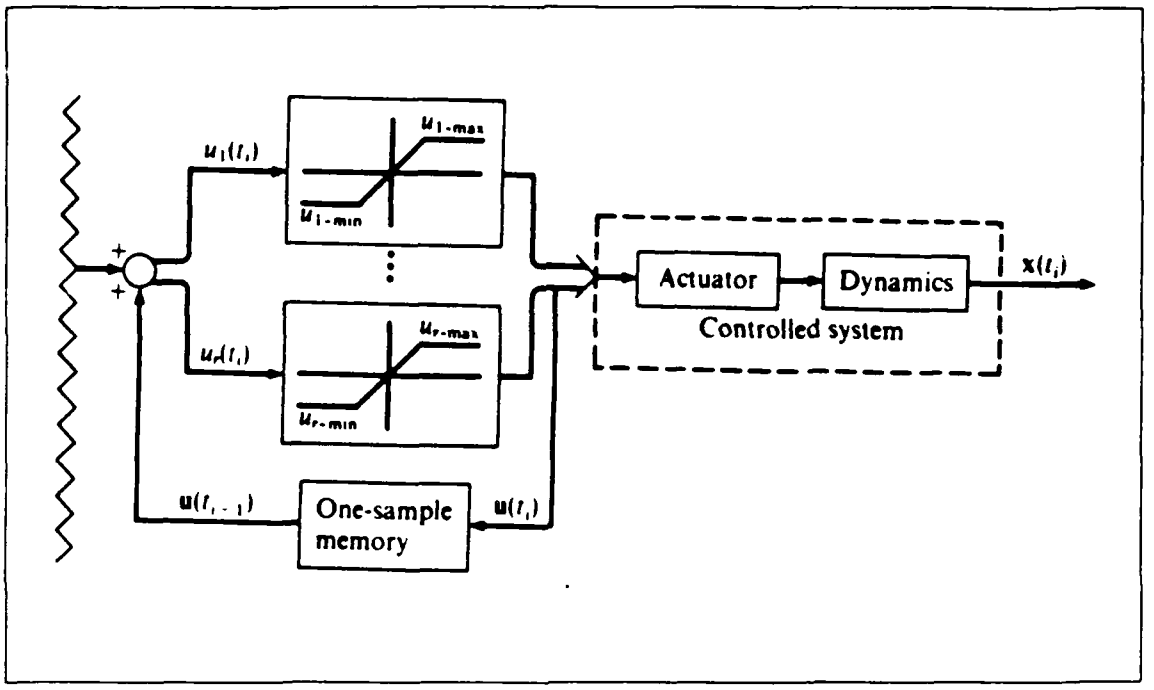



Figure 2.3. Antiwindup Compensation

### *2.5 Kalman Filter*

The assumption of full-state feedback used in the previous development is unrealistic. Complete knowledge of all the states is rarely available, and the measurements that are available are usually corrupted by noise (uncertainty). To account for this uncertainty, a Kalman filter is employed to provide estimates of the states based on measurements and what is known about the system model, and the approximations used in deriving the model. Because of certainty equivalence, the design of the Kalman filter can be performed independently of the design of the full-state feedback CGT/PI controller. After both designs are complete they are combined and, if necessary, retuned for performance, possibly by use of LTR tuning (Section 2.7). This section discusses the design of the Kalman filter.

When used in conjunction with a CGT/PI controller, the Kalman filter must provide estimates of both the design model states and disturbance states (if they are modeled) of Equations (2.1) and (2.3). The appropriate system model [5] upon

which to base the Kalman filter design is therefore generally an augmented system model

$$\mathbf{x_a}(t_{i+1}) = \Phi_\mathbf{a}\mathbf{x_a}(t_i) + \mathbf{B_a}\mathbf{u}(t_i) + \mathbf{w_{da}}(t_i) \tag{2.62}$$

(2.63)

where

$$\mathbf{x_a}(t_i) = \begin{bmatrix} \mathbf{x}(t_i) \\ \mathbf{n_d}(t_i) \end{bmatrix} \tag{2.64}$$

$$\mathbf{w_{da}}(t_i) = \begin{bmatrix} \mathbf{w_d}(t_i) \\ \mathbf{G_n}\mathbf{w_{dn}}(t_i) \end{bmatrix} \tag{2.65}$$

$$\mathbf{\Phi_a} = \begin{bmatrix} \mathbf{\Phi} & \mathbf{E_x} \\ \mathbf{0} & \mathbf{\Phi_n} \end{bmatrix} \tag{2.66}$$

$$\mathbf{B_a} = \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \tag{2.67}$$

$$\mathbf{Q_{da}} = \begin{bmatrix} \mathbf{Q_d} & \mathbf{0} \\ \mathbf{0} & \mathbf{G_n}\mathbf{Q_{dn}}\mathbf{G_n}^T \end{bmatrix} \tag{2.68}$$

The measurement equation is assumed to provide discrete time measurements given by

$$\mathbf{z}(t_i) = \mathbf{H}\mathbf{x}(t_i) + \mathbf{H_n}\mathbf{n_d}(t_i) + \mathbf{v}(t_i) \tag{2.69}$$

where $\mathbf{v}$ is a zero-mean white Gaussian noise of covariance $\mathbf{R}$, usually assumed independent of $\mathbf{w_d}$ and $\mathbf{w_{dn}}$. In the augmented notation used above, this becomes

$$\mathbf{z}(t_i) = \mathbf{H_a}\mathbf{x_a}(t_i) + \mathbf{v}(t_i) \tag{2.70}$$

where

$$\mathbf{H_a} = \begin{bmatrix} \mathbf{H} & \mathbf{H_n} \end{bmatrix} \tag{2.71}$$

The Kalman filter uses the augmented model to propagate the augmented state $\hat{\mathbf{x}}_\mathbf{a}$ and covariance $\mathbf{P}_\mathbf{a}$ estimates forward in time from $(t_{i-1}^+)$ to $(t_i^-)$ by the propagation equations

$$\hat{\mathbf{x}}_\mathbf{a}(t_i^-) = \mathbf{\Phi}_\mathbf{a}\hat{\mathbf{x}}_\mathbf{a}(t_{i-1}^+) + \mathbf{B}_\mathbf{a}\mathbf{u} \tag{2.72}$$

$$\mathbf{P}_\mathbf{a}(t_i^-) = \mathbf{\Phi}_\mathbf{a}\mathbf{P}_\mathbf{a}(t_{i-1}^+)\mathbf{\Phi}_\mathbf{a}{}^T + \mathbf{Q}_\mathbf{da} \tag{2.73}$$

At time $t_i$, the measurement $\mathbf{z}$ becomes available. The estimate is updated by incorporating the measurement into the filter gain $\mathbf{K}$, which is used to update the state and covariance estimates through the relations

$$\mathbf{K} = \mathbf{P}_\mathbf{a}(t_i^-)\mathbf{H}_\mathbf{a}^\mathbf{T}\left[\mathbf{H}_\mathbf{a}\mathbf{P}_\mathbf{a}(t_i^-)\mathbf{H}_\mathbf{a}^\mathbf{T} + \mathbf{R}\right]^{-1} \tag{2.74}$$

$$\hat{\mathbf{x}}_\mathbf{a}(t_i^+) = \hat{\mathbf{x}}_\mathbf{a}(t_i^-) + \mathbf{K}\left[\mathbf{z}(t_i) - \mathbf{H}_\mathbf{a}\hat{\mathbf{x}}_\mathbf{a}(t_i^-)\right] \tag{2.75}$$

$$\mathbf{P}_\mathbf{a}(t_i^+) = \mathbf{P}_\mathbf{a}(t_i^-) - \mathbf{K}\mathbf{H}_\mathbf{a}\mathbf{P}_\mathbf{a}(t_i^-) \tag{2.76}$$

In the above development, constant gain $\mathbf{K}$ is assumed (the Kalman filter is in steady state), and $\mathbf{P}_\mathbf{a}(t_i^-)$ has the same value for all time, as does $\mathbf{P}_\mathbf{a}(t_i^+)$. The initial condition for the above recursion is $\hat{\mathbf{x}}_\mathbf{a}(t_0)$, where these values are the augmented state initial conditions.

When the Kalman filter is cascaded with the CGT/PI controller, the resulting CGT/PI/KF control law is

$$\begin{aligned}\mathbf{u}(t_i) = {} & \mathbf{u}(t_{i-1}) - \mathbf{K}_\mathbf{x}\left[\hat{\mathbf{x}}(t_i^+) - \hat{\mathbf{x}}(t_{i-1}^+)\right] \\ & + \mathbf{K}_\mathbf{z}\left\{\begin{bmatrix}\mathbf{C}_\mathbf{m} & \mathbf{D}_\mathbf{m}\end{bmatrix}\begin{bmatrix}\mathbf{x}_\mathbf{m}(t_{i-1}) \\ \mathbf{u}_\mathbf{m}(t_i)\end{bmatrix} - \begin{bmatrix}\mathbf{C} & \mathbf{D}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{x}}(t_{i-1}^+) \\ \mathbf{u}(t_{i-1})\end{bmatrix}\right\} \\ & + \left[\mathbf{K}_\mathbf{x}\mathbf{A}_{11} + \mathbf{A}_{21}\right]\left[\mathbf{x}_\mathbf{m}(t_i) - \mathbf{x}_\mathbf{m}(t_{i-1})\right] \\ & + \left[\mathbf{K}_\mathbf{x}\mathbf{A}_{13} + \mathbf{A}_{23}\right]\left[\hat{\mathbf{n}}_\mathbf{d}(t_i^+) - \hat{\mathbf{n}}_\mathbf{d}(t_{i-1}^+)\right]\end{aligned} \tag{2.77}$$

The state estimates are incorporated into the control law after measurement update, rather than using the suboptimal estimates at time $t_i^-$. The suboptimal estimates may be preferred in some applications, since Equation (2.77) cannot be

totally precomputed before time $t_i$, thus it imposes a computational delay at time $t_i$; if $\hat{\mathbf{x}}_a(t_i^-)$ estimates replaced $\hat{\mathbf{x}}_p(t_i^+)$ estimates in Equation (2.77) this computational delay would be eliminated.

### *2.6 Implicit Model-Following*

In the development of Sections 2.2 and 2.3, explicit weights were placed on perturbation deviations from zero. This is known as explicit model-following. If instead a cost were to be placed on deviations from the dynamic response of a model which possessed desired stability and robustness characteristics, the system would be forced to emulate those characteristics. This is known as implicit model-following. With regard to choosing an implicit model, Gilbert [6] has shown that the sensitivity of the system's poles to variations in the system's state transition matrix $\Phi$ is minimized when the nominal eigenvectors of the plant are orthogonal. These variations could be caused by either modeling errors or system failures. Thus orthogonal system eigenvectors increase the closed loop system robustness, and so it is appropriate to seek an implicit model with dynamics characterized by orthogonal eigenvectors.

The use of an LQ weighting matrix (developed in Sections 2.3 and 2.4) based on an implicit model has a direct effect on the plant's closed-loop poles. This is because the feedback control causes the eigenvalues and eigenvectors to approach those of the implicit model [14]. The amount of control energy which can be expended affects how close the plant's closed-loop poles come to matching the implicit model's poles, with convergence occurring when controls have zero weighting (and deviations in outputs from the implicit model are weighted in a positive definite manner). While the command model of Section 2.2 can be used as an implicit model, this command model typically contains performance characteristics such as handling qualities, rather than stability robustness characteristics, and hence will not enhance the system's robustness in the face of varying plant param-

eters. Rather, an implicit model which possesses desired eigenvalues and nearly orthogonal eigenvectors will force the nominal closed-loop system to approach these eigenvalues and eigenvectors, enhancing the overall robustness of the system.

Previously, the objective of the PI controller was to regulate perturbation outputs to zero. This was presented in Equation (2.45) and is rewritten here for convenience,

$$\delta \mathbf{y}(t_i) = \mathbf{y}(t_i) - \mathbf{y_m} \tag{2.78}$$

With implicit model-following, the objective instead is to force the design model *dynamics* to mimic the model system, such that

$$\delta \mathbf{y}(t_{i+1}) = \mathbf{\Phi_m} \delta \mathbf{y}(t_i) \tag{2.79}$$

where $\mathbf{\Phi_m}$ is the implicit model state transition matrix. Again, note that the plant is not necessarily mimicking the command model, but rather an implicit model that contains the desired eigenvalues and eigenvectors that the designer wishes the closed-loop system to possess.

Using Equation (2.79), the resulting optimal control law is found from the quadratic cost

$$\begin{aligned} \mathbf{J} \;=\; & 1/2 \sum_{i=0}^{N} [\delta \mathbf{y}(t_{i+1}) - \mathbf{\Phi_m} \delta \mathbf{y}(t_i)]^T \mathbf{Q_I} [\delta \mathbf{y}(t_{i+1}) - \mathbf{\Phi_m} \delta \mathbf{y}(t_i)] \\ & + \delta \mathbf{u}(t_i)^T \mathbf{R_I} \delta \mathbf{u}(t_i) + \Delta \mathbf{u}(t_i)^T \mathbf{U_R} \Delta \mathbf{u}(t_i) \end{aligned} \tag{2.80}$$

where again $N \rightarrow \infty$. The $\mathbf{Q_I}$ term in this equation weights output deviations from the implicit model dynamics and $\mathbf{R_I}$ weights control magnitudes. The $\mathbf{U_R}$ term weights control pseudorates, as before.

The above equation can be rewritten as

$$\mathbf{J} = 1/2 \sum_{i=0}^{N} \begin{bmatrix} \delta \mathbf{x}(t_i) \\ \delta \mathbf{u}(t_i) \\ \Delta \mathbf{u}(t_i) \end{bmatrix}^T \begin{bmatrix} \hat{\mathbf{Q}}_\mathbf{I} & \hat{\mathbf{S}}_\mathbf{I} & \mathbf{0} \\ \hat{\mathbf{S}}_\mathbf{I}^T & \hat{\mathbf{R}}_\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U_R} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x}(t_i) \\ \delta \mathbf{u}(t_i) \\ \Delta \mathbf{u}(t_i) \end{bmatrix} \tag{2.81}$$

where

$$\hat{\mathbf{Q}}_\mathbf{I} = [\mathbf{C\Phi} - \mathbf{\Phi_m C}]^T \mathbf{Q_I} [\mathbf{C\Phi} - \mathbf{\Phi_m C}] \tag{2.82}$$

$$\hat{\mathbf{S}}_\mathbf{I} = [\mathbf{C\Phi} - \mathbf{\Phi_m C}]^T \mathbf{Q_I CB} \tag{2.83}$$

$$\hat{\mathbf{R}}_\mathbf{I} = \mathbf{R_I} + \mathbf{B}^T\mathbf{C}^T\mathbf{Q_I CB} \tag{2.84}$$

where $\mathbf{D} = \mathbf{0}$ is assumed. The above equation is of the same form as Equation (2.50). However, the desired stability characteristics are directly embedded in the cost function definition. The desired state model does not explicitly appear in the final controller structure, but is rather implicit to the structure. Hence the name implicit model-following.

### *2.7 Loop Transfer Recovery*

As previously stated, when the Kalman filter and the deterministic CGT/PI controller are combined to form the optimal stochastic controller, all stability robustness guarantees are lost. The good robustness characteristics of the full-state controller can be recovered asymptotically by tuning the filter using Loop Transfer Recovery (LTR) techniques [14].

The LTR technique introduces pseudonoise into the design model upon which the Kalman filter is based. Using the continuous-time noise covariance kernel matrix of Equation (A.1) in Appendix A, and assuming $\mathbf{G}$ is the identity matrix, we use LTR to replace the $\mathbf{Q}$ with

$$\mathbf{Q}' = \mathbf{Q} + q^2\mathbf{BVB}^T \tag{2.85}$$

where $\mathbf{B}$ is the continuous-time control matrix and $\mathbf{V}$ is a positive definite matrix which the designer chooses to affect the relative rates of recovery in various loops. ($\mathbf{Q}'$ is a continuous-time noise covariance matrix which is discretized to $\mathbf{Q}'_\mathbf{d}$ for use in our Kalman filter implementation, Equations (2.1) and (2.68). This discretization is performed via $\int \mathbf{\Phi G Q' G}^T\mathbf{\Phi}^T d\tau$.) As q increases, the system will

asymptotically approach the robustness characteristics exhibited by the full-state controller. It should be noted, however, that as pseudonoise increases, performance at the nominal condition will degrade. Therefore, the designer must trade off robustness for performance at the nominal condition.

# *III. Aircraft Description and Models*

## *3.1 Introduction*

This chapter provides a description of the Control Reconfigurable Combat Aircraft (CRCA), the state space matrices which describe the aircraft and its failure conditions, and how these matrices were simplified in the development of a pitch rate controller. All of the matrices in this chapter represent continuous time aircraft behavior, and were provided by the Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio [18] .

## *3.2 Aircraft Description*

The CRCA is a modified version of the NASA/Grumman Advanced Tactical Fighter (ATF) class aircraft. This aircraft is a "paper airplane"; that is, it is a mathematical model simulating an aircraft. It does not physically exist, but it does represent a potential design of a mid-1990's fighter type aircraft. A sketch of the CRCA is given in Figure 3.1. There are nine control surfaces: two canards, four flaps, two elevators, and one rudder. The canards provide the main pitching moment, as well as some roll and yaw moments when used diferentially. The flaps on each wing provide rolling, pitching, and yawing moments, while the elevators provide pitching and rolling moments. The rudder is used to supply yaw moment.

The control surfaces have a maximum deflection rate limit of $\pm$ 100 deg/sec. The flaps, elevators and rudder have maximum position limits of $\pm$ 30 deg, while the canards have deflection limits of $-30$ deg to $+60$ deg.

The flight condition chosen in this thesis for controller design and evaluation was Air Combat Mode (ACM) Entry. This flight condition represents the CRCA at 30,000 feet, a forward velocity of .9 Mach, and the aircraft initially trimmed with wings level. In this flight condition the aircraft is expected to encounter hostile

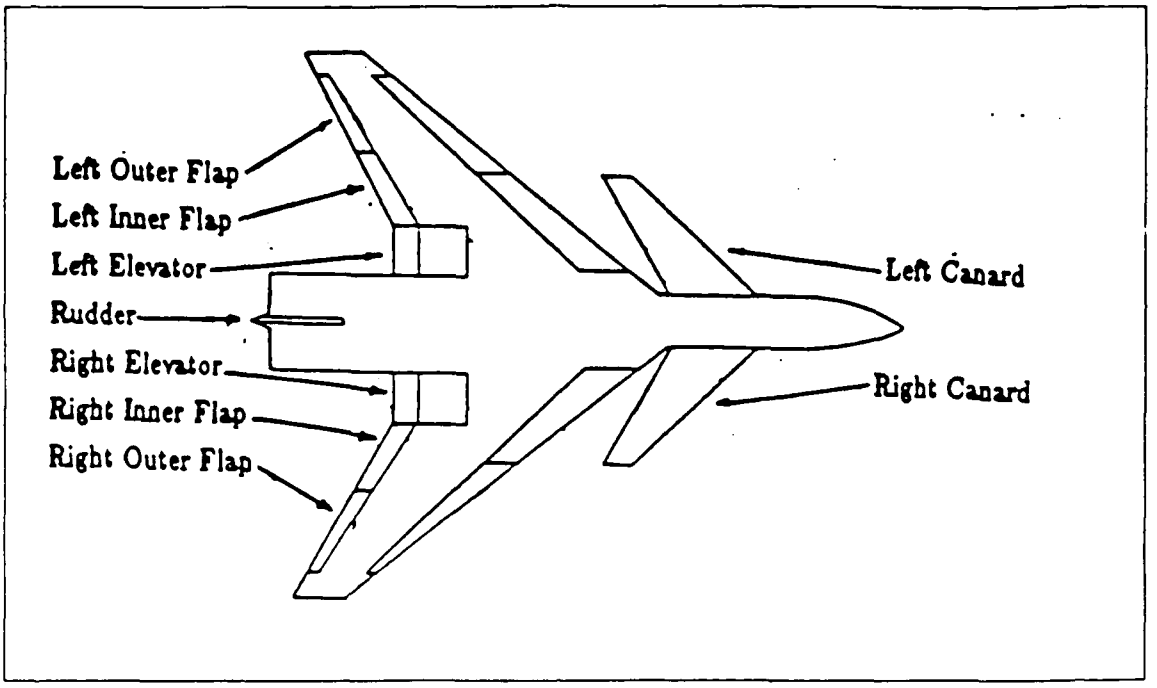



Figure 3.1. Control Reconfigurable Combat Aircraft

aircraft, and the CRCA must respond with Level 1 handling qualities, as defined in Mil-F-8785C [1].

The CRCA can be represented via the continuous time state space equation

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned} \tag{3.1}$$

where:

$$\mathbf{x} = \begin{bmatrix} u \\ w \\ \theta \\ q \\ \beta \\ p \\ r \\ \phi \end{bmatrix} = \begin{bmatrix} \textit{forward velocity} \\ \textit{vertical velocity} \\ \textit{pitch angle} \\ \textit{pitch rate} \\ \textit{sideslip angle} \\ \textit{roll rate} \\ \textit{yaw rate} \\ \textit{bank angle} \end{bmatrix} \tag{3.2}$$

and

$$\mathbf{u} = \begin{bmatrix} \textit{left canard deflection angle} \\ \textit{right canard deflection angle} \\ \textit{left inner flap deflection angle} \\ \textit{left outer flap deflection angle} \\ \textit{right inner flap deflection angle} \\ \textit{right outer flap deflection angle} \\ \textit{left elevator deflection angle} \\ \textit{right elevator deflection angle} \\ \textit{rudder deflection angle} \end{bmatrix} \tag{3.3}$$

The **A** matrix is 8x8 and the **B** matrix is 8x9. The **C** matrix must have 8 columns, but the number of rows (system outputs) is determined by the designer. The **D** matrix is assumed to be **0**.

### *3.3 Aircraft Truth Models*

For the case of ACM Entry, the nominal (no failures) **A** and **B** matrices of Equation (3.1) are

$$\mathbf{A} = \begin{bmatrix} -0.0119 & -0.0186 & -32.1804 & -31.2350 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.0324 & -1.0634 & -1.1238 & 894.4548 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.0002 & 0.0069 & 0.0000 & -0.6015 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.0929 & 0.0349 & -0.9994 & 0.0360 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & -27.8066 & -2.0376 & 0.4913 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 2.4582 & -0.0241 & -0.4377 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0349 & 0.0000 \end{bmatrix} \tag{3.4}$$

$$\mathbf{B} = \begin{bmatrix} .0411 & .0411 & .1322 & .0866 & .1322 & .0866 & .1018 & .1018 & .0000 \\ -.3163 & -.3163 & -.9597 & -.6194 & -.9597 & -.6194 & -1.0183 & -1.0183 & .0000 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .1014 & .1014 & -.0284 & -.0215 & -.0284 & -.0215 & -.0200 & -.0200 & .0000 \\ .0003 & -.0003 & -.0002 & -.0001 & .0002 & .0001 & -.0001 & .0001 & .0000 \\ .0762 & -.0762 & .2219 & .2011 & -.2219 & -.2011 & .1109 & -.1109 & .1144 \\ .0486 & -.0486 & .0029 & .0021 & -.0029 & -.0021 & .0021 & -.0021 & -.0544 \\ .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 & .0000 \end{bmatrix} \tag{3.5}$$

From the above equations it can be seen that, while all control surfaces affect all states, the effects of the control surfaces can be separated into longitudinal and lateral responses. For example, excitation of one channel does not crossfeed into the other channel. Also, the **A** matrix is block diagonal, indicating that there is no cross-coupling between the first four and last four states. Thus, a flight control system can initially be divided into a longitudinal controller and a lateral controller. This greatly reduces the complexity of the design task. After both controllers are designed, they can be combined and evaluated against the complete aircraft model. While this was the original intent of this endeavor, the effort required to design the CGTPIKF CAD package required the majority of the time allocated for this

thesis. Hence, the design presented in this thesis only covers the longitudinal flight control system.

Examining the **B** matrix, it can be seen that, when considering only the longitudinal states of the system, each control surface's left and right matrix entries are identical. This is as expected for such symmetric surfaces being used together (rather than differentially, as for affecting the lateral mode). For purposes of comparison with other thesis efforts to design a controller for the CRCA [8,15], the **B** matrix left and right components of each surface were combined. Also, due to the CGTPIKF software constraint that the number of inputs must equal the number of outputs, it was decided to combine the inner and outer trailing edge flaps and elevators into one pseudosurface, called the flaperon. The outputs to be controlled were then $\theta$ (pitch angle) and q (pitch rate). Greater (equal) number of inputs and outputs could be accomodated by the LQG methodology (and the CGTPIKF software), but two inputs and outputs were considered as a reasonable starting point for a design. Additional rationale for these choices is provided in Chapter 4.

The **A** and **B** matrices resulting from the above simplifications are

$$\mathbf{A} = \begin{bmatrix} -0.0119 & -0.0186 & -32.1804 & -31.2350 \\ -0.0324 & -1.0634 & -1.1238 & 894.4548 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \\ -0.0002 & 0.0069 & 0.0000 & -0.6015 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} .0822 & .6412 \\ -.6326 & -5.1948 \\ .0000 & .0000 \\ .2028 & -.1398 \end{bmatrix} \tag{3.6}$$

The first column of the **B** matrix represents the effect of the canards on the states, while the second column represents the effects of the flaperons. Because of the

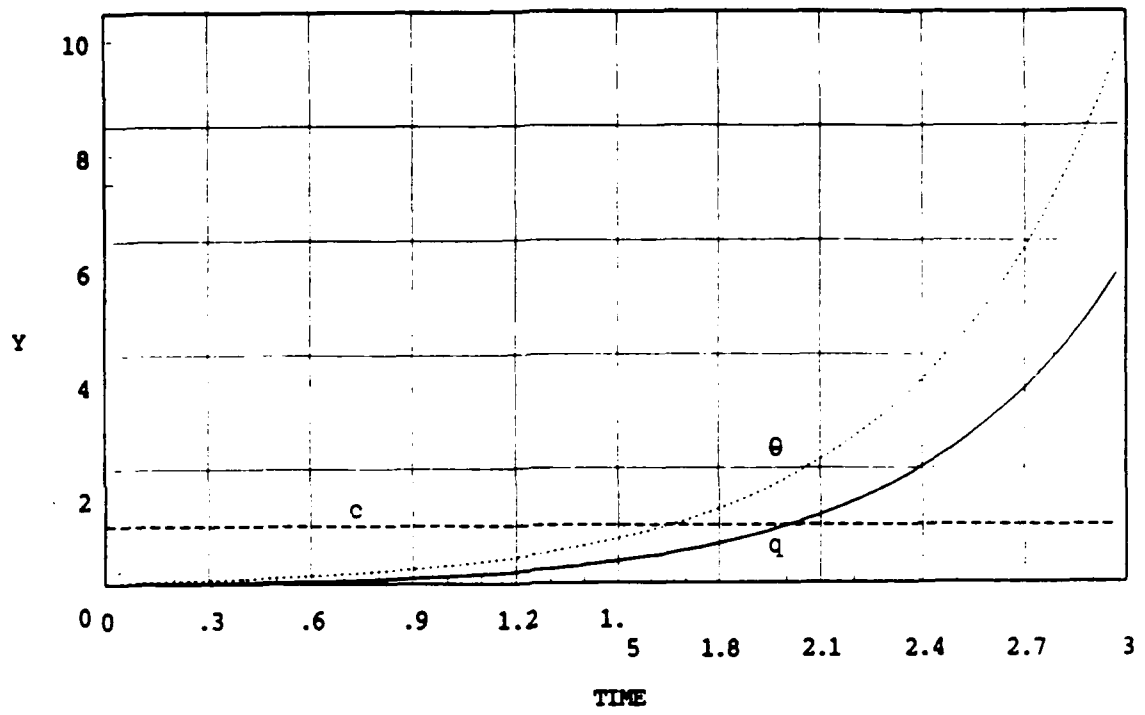



Figure 3.2. Design Model Response To A Canard Step Input

desire to control $\theta$ and q, the C matrix becomes

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

It should be noted that there is a unit discrepancy in the above equations with the numerical values as given. The control inputs to the **B** matrix are deflection of control surfaces measured in degrees. The resulting behavior of the states is in radians. Also, positive canard deflection results in a pitch up manuever; positive flaperon deflection results in a pitch down manuever. The response of the aircraft model of Equations (3.7) and (3.8) (without actuator dynamics) to a step input is given in Figures 3.2 and 3.3. These figures represent a fundamental characteristic of the uncontrolled aircraft, which does not appreciably change when actuator dynamics are included in the model.

The effect of actuator dynamics on system response is considered in this thesis. The dynamics associated with the canards and flaperons can be modeled

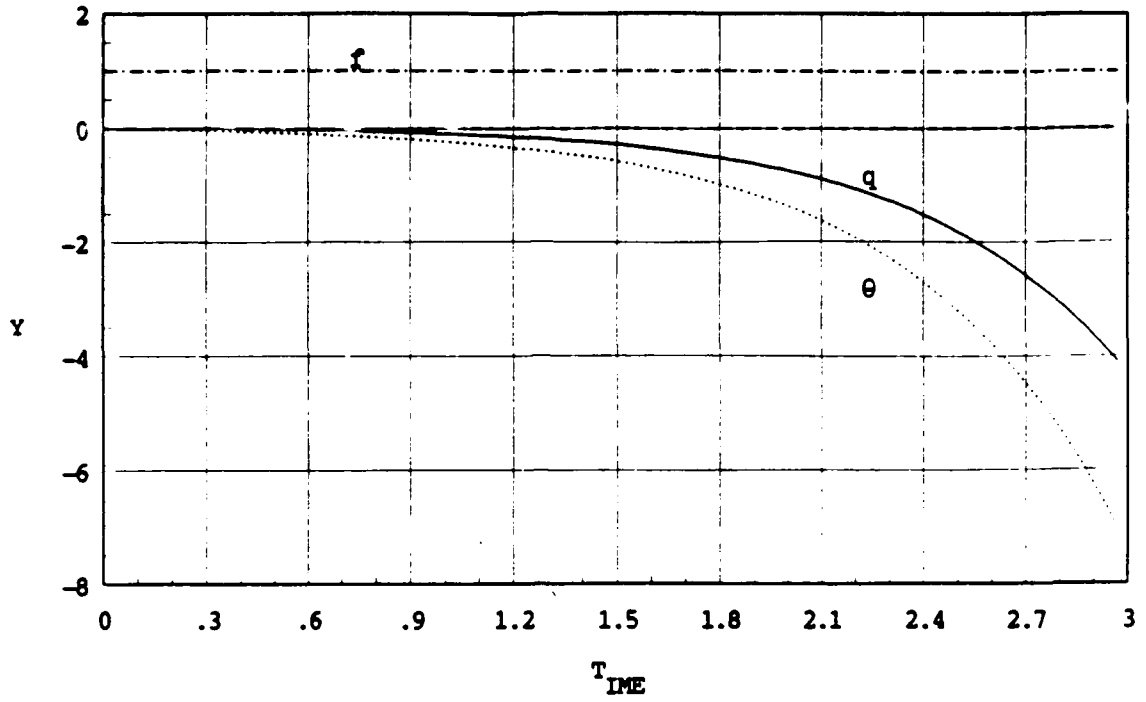



Figure 3.3. Design Model Response To A Flaperon Step Input

via a fourth order transfer function, but based on Neumann's work with the CRCA [15], it was decided to limit modeling of the actuators to a first order approximation. Both the canards and flaperons were modeled by

$$\dot{\delta}_{surface} = -20\delta_{surface} + 20\delta_{cmd} \tag{3.8}$$

Various failure conditions were considered in the evaluation of the pitch rate controller. These failure conditions and their respective names (which will be referred to in the next chapter) are listed in Table 3.1. The specific **A** and **B** matrices are not presented explicitly (to allow for wider distribution of this thesis), but they are available from AFWAL/FIGL, the organization which sponsored this research. Because the controller was based on a design model which did not include actuator dynamics, it was first evaluated against the truth models without actuator dynamics, then against the same truth models with actuator dynamics added. The results of the controller design are presented in the next chapter.

| Failure Description | Truth Model Designation | |
|---|---|---|
| | w/o actuators | with actuators |
| unimpaired aircraft | TM01 | TM101 |
| 25 percent canard loss | TM03 | TM103 |
| 25 percent outer flap loss | TM06 | TM106 |
| 25 percent outer flap and elevator loss | TM16 | TM116 |
| canard fail to trail | TM19 | TM119 |
| outer flap and elevator fail to trail | TM23 | TM123 |

Table 3.1. Failure Conditions

# *IV. CGT/PI/KF Controller Design and Evaluation*

## *4.1 Introduction*

This chapter presents the design methodology of the CGT/PI/KF flight control system for the CRCA. First, the choice of design and command models for the deterministic full-state feedback CGT/PI controller, as well as initial weighting matrices and selection of an implicit model, are presented. This section includes evaluation of the controller against the nominal truth model of Equation (3.6). Section 4.3 presents the evaluation results of the controller against truth models representing various failure conditions. These truth models do not include actuator dynamics. Section 4.4 includes the evaluation of the controller against the same failure conditions, but with the actuator dynamics of Equation (3.8) included in the truth models. This section also includes a revised controller which was designed to improve performance.

## *4.2 Initial Deterministic Controller Design and Evaluation Against Nominal Truth Model*

The first step in designing a CGT/PI controller using the CGTPIKF computer aided design package is to pick a design model. Because the control system was to be a pitch rate controller, q (pitch rate) became the primary variable of interest. As stated in Chapter 3, the control surfaces were combined into two pseudo-surfaces, those being the canard and the flaperon. This was done for purposes of simplifying the design task. Because of the software limitation of the number of inputs and outputs being equal, one more controlled variable was required. This was selected as $\theta$ (pitch angle), as the control of $\theta$ is a natural result of controlling q, and would simplify the design process. These selections were also the basis of the formulation of the nominal truth model (TM1) **B** and **C** matrices, because the initial design model was to be identical to the nominal truth model. Although the

actual truth model consisted of an 8 by 9 **B** matrix, this was considered to be a valid manipulation which simplified the design task.

The model consists of the four longitudinal states,

$$\mathbf{x} = \begin{bmatrix} u \\ w \\ \theta \\ q \end{bmatrix} \tag{4.1}$$

The continuous time **A**, **B** and **C** matrices are

$$\mathbf{A} = \begin{bmatrix} -0.0119 & -0.0186 & -32.1804 & -31.2350 \\ -0.0324 & -1.0634 & -1.1238 & 894.4548 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \\ -0.0002 & 0.0069 & 0.0000 & -0.6015 \end{bmatrix} \tag{4.2}$$

$$\mathbf{B} = \begin{bmatrix} .0822 & .6412 \\ -.6326 & -5.1948 \\ .0000 & .0000 \\ .2028 & -.1398 \end{bmatrix} \tag{4.3}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

As already stated, the **D** matrix is equal to **0**. The use of noise shaping filter models was not investigated in this thesis, so the $\mathbf{E_x}, \mathbf{E_y}, \mathbf{\Phi_n}$, and $\mathbf{G_n}$ matrices were all set to **0**. For the remainder of this development, unless otherwise stated, all matrices are in the continuous time domain.

With the design model and controlled outputs selected, the next step was to choose a command model. The input to the command model drives the entire CGT/PI/KF controller, so it was decided to make the input to the command model the desired values of the controlled output q. The output of the command model would then be the desired state trajectory that $\theta$ and q of the actual system

would try to emulate. Because of the requirement of Level I handling qualities [1], a rise time of .6 sec for q was chosen as a criterion to define a suitable reference trajectory. While this is somewhat faster than actually required, it is a good idea to have the reference trajectory slightly faster than what is desired for the plant, to offset possible performance constraints later in the design process. With this in mind, and basing the design on Floyd's work [5:158] tempered by the physical relationship of $\theta$ and q, the command model became

$$\mathbf{A_m} = \begin{bmatrix} 0 & 1 \\ 0 & -5 \end{bmatrix} \tag{4.5}$$

$$\mathbf{B_m} = \begin{bmatrix} 0 & 0 \\ 0 & 5 \end{bmatrix} \tag{4.6}$$

$$\mathbf{C_m} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.7}$$

$$\mathbf{D_m} = \mathbf{0} \tag{4.8}$$

The rationale behind setting the top row of the $\mathbf{B_m}$ matrix to $\mathbf{0}$ is that q values are commanded, not $\theta$ values. The time response of the command model for a .035 rad/sec pitch rate command is given in Figure 4.1.

Because the design model is unstable, an open-loop CGT controller cannot be pursued. Therefore, the next step is to generate PI controller gains. Based on previous work [5,7,14,17], it was recognized that the PI controller would probably have to be both an explicit and an implicit model-follower. Explicit model-following is the standard form of a PI controller, while implicit model-following is an alternate approach of choosing the cost for LQG synthesis of the PI controller, aimed at enhancing the closed-loop stability robustness characteristics. An explicit-implicit controller combines the weighting matrices of Equations (2.50) and (2.81) into a

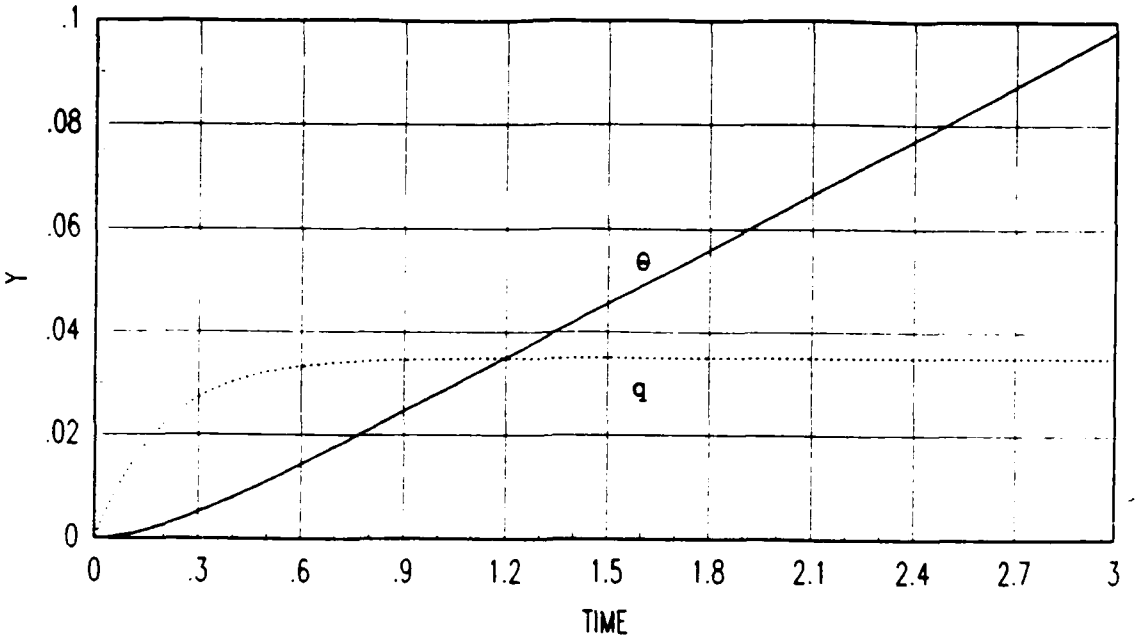



Figure 4.1. Command Model Response To .035 rad/sec Pitch Rate Command

cost weighting matrix

$$\mathbf{X_{IE}} = \begin{bmatrix} \mathbf{X_{11}} + \hat{\mathbf{Q}}_\mathbf{I} & \mathbf{X_{12}} + \hat{\mathbf{S}}_\mathbf{I} \\ \mathbf{X_{12}}^T + \hat{\mathbf{S}}_\mathbf{I}^T & \mathbf{X_{22}} + \hat{\mathbf{R}}_\mathbf{I} \end{bmatrix} \tag{4.9}$$

Both explicit and implicit weights (Equations (2.28) and (2.80)) were therefore generated from the beginning. The CGTPIKF software allows the user to pursue either an explicit or an implicit model-follower, or a combination of the two, so establishing implicit weighting matrices did not prohibit the generation and evaluation of a pure explicit model-following PI controller.

Initial quadratic cost weights were chosen to penalize perturbation deviations of the output from zero (for the explicit case) or perturbation dynamics deviations from those of the implicit model (for the implicit model-follower). Control magnitudes and rates were not considered to be the primary concern at this stage of the

design, so they received relatively light weighting. The initial costs were chosen as

$$\mathbf{Y} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \tag{4.10}$$

$$\mathbf{U_M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.11}$$

$$\mathbf{Q_I} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \tag{4.12}$$

$$\mathbf{R_I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.13}$$

$$\mathbf{U_R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.14}$$

The implicit model was arbitrarily chosen with orthogonal eigenvectors, and eigenvalues which would result in a fast time response. The initial implicit model was

$$\mathbf{A_{Im}} = \begin{bmatrix} -5 & 0 \\ 0 & -5 \end{bmatrix} \tag{4.15}$$

The eigenvalues of the above matrix correspond to the q entry of the command model, which had a .6 second rise time. With these cost weighting matrices and the implicit model in place, several initial controllers could be evaluated.

The CGT gains which result from the command model given in Equations (4.5) through (4.8) are

$$\mathbf{A_{11}} = \begin{bmatrix} -23.4527 & -14.8251 \\ -212.2382 & -118.2140 \\ 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{bmatrix} \tag{4.16}$$

$$\mathbf{A_{21}} = \begin{bmatrix} 34.2261 & 59.0917 \\ 39.2082 & 109.6057 \end{bmatrix} \tag{4.17}$$

$\mathbf{A_{13}}$ and $\mathbf{A_{23}}$ are 0 because disturbances are not being considered. $\mathbf{A_{12}}$ and $\mathbf{A_{22}}$ are not part of the closed-loop CGT/PI control law.

These CGT gains remain constant regardless of the weighting matrices or implicit model used in the PI controller. These are the CGT gains used throughout the rest of this chapter.

The pure explicit model-following CGT/PI controller with the above cost weighting matrices results in the PI gains

$$\mathbf{K_x} = \begin{bmatrix} 22.4065 & -0.2430 & 2.5580 & 138.8589 \\ 33.2951 & -0.2918 & 0.5398 & -8.4911 \end{bmatrix} \tag{4.18}$$

$$\mathbf{K_z} = \begin{bmatrix} 22.8702 & 33.9185 \\ 23.5549 & 25.8476 \end{bmatrix} \tag{4.19}$$

Plots of the output (q) and actuator response are given in Figures 4.2 and 4.3, and will be discussed shortly. In some of the figures of this chapter, the actuator response is titled 'UOPTIMAL', because with zero order actuators the actuator response is the optimal control u. The 'c' designation on the plot represents the canard, while 'f' represents the flaperon. Remember that actuator response is in degrees, while q is measured in rad/sec. Unless otherwise stated, the response is to a commanded q value of .035 rad/sec (2 deg/sec). Also, at this stage of the design, actuator limits were not included in the model.

The implicit model-following controller resulted in an $\mathbf{X_{IE}}$ matrix (Equation (4.9)) which was, to the numerical precision of MATRIX$_X$ , not positive semidefinite. Due to the structure of the implicit model (i.e., two rows less than the design model), this was a common problem of trying to use solely an implicit model-following PI controller. Therefore implicit model-following was no longer pursued, although implicit with explicit model-following was pursued.

The explicit-implicit model-following CGT/PI controller with the above cost

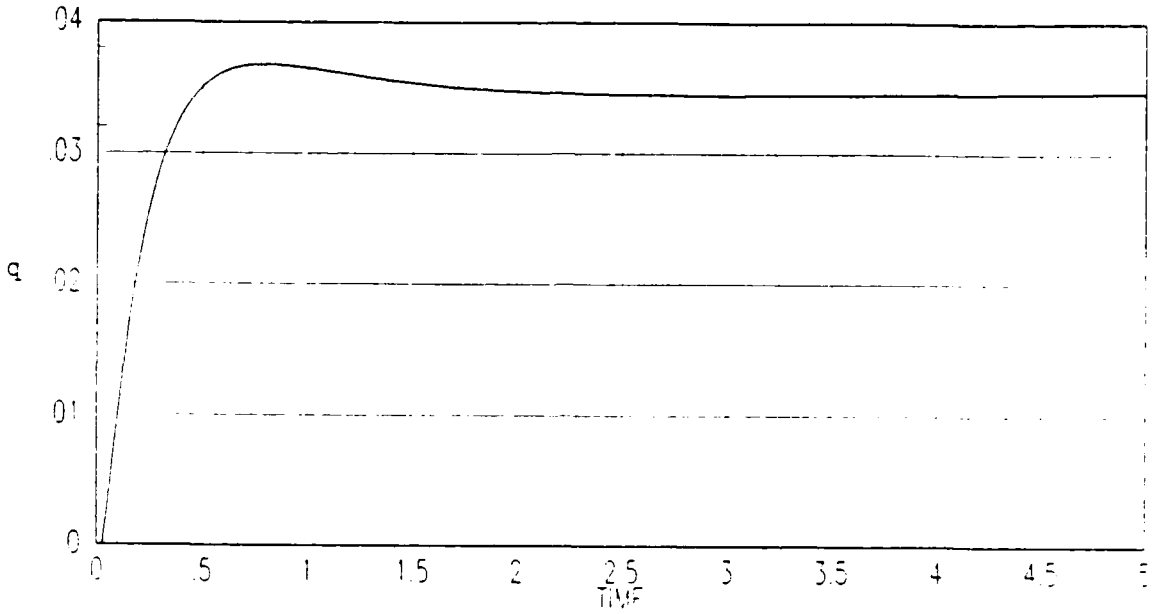



Figure 4.2. Explicit Model-Following Controller: Pitch Rate Response

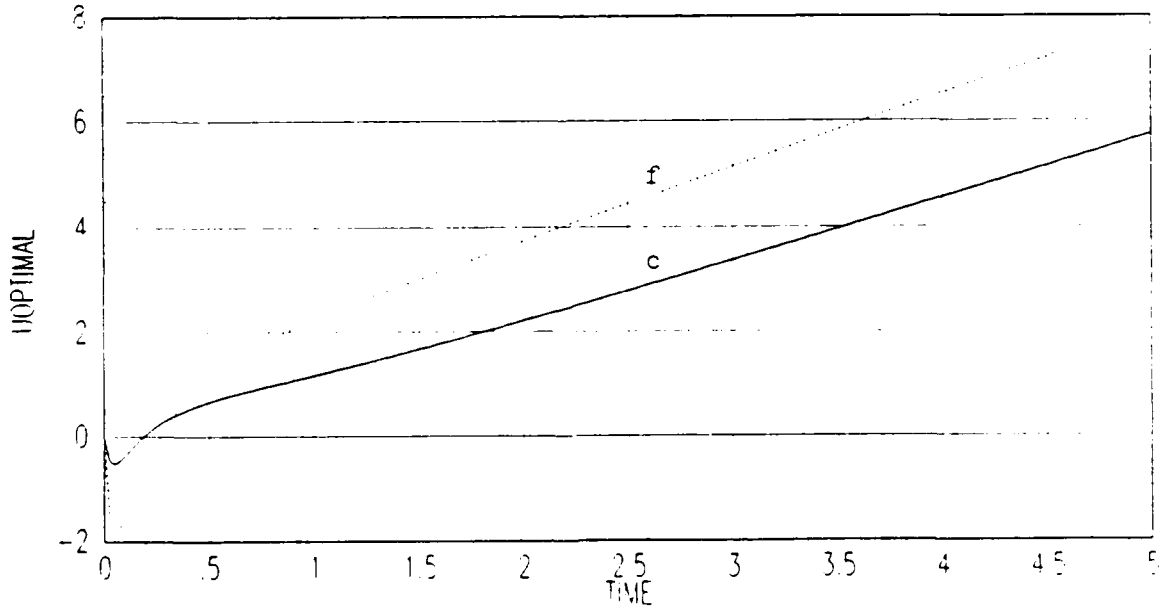



Figure 4.3. Explicit Model-Following Controller: Actuator Response

weighting matrices results in the PI gains

$$\mathbf{K_x} = \begin{bmatrix} 26.9681 & -0.2471 & 3.4545 & 187.2642 \\ 39.2530 & -0.3737 & 0.3794 & -25.3616 \end{bmatrix} \tag{4.20}$$

$$\mathbf{K_z} = \begin{bmatrix} 34.3830 & 44.3127 \\ 23.0372 & 27.8876 \end{bmatrix} \tag{4.21}$$

The response of the explicit-implicit controller is given in Figures 4.4 and 4.5. The response of this controller matches the command model quite well. The explicit-only controller has a small overshoot, but its rise time is slightly faster than that of the explicit-implicit controller, while its settling time is approximately .5 seconds slower. Both of these responses are considered quite satisfactory, although the explicit-implicit controller is better.

Looking at the actuator responses, the performance is questionable. The controllers exhibit nearly identical canard and flaperon response. Both aerodynamic surfaces have a positive ramp, which is undesirable. Also, referring to Figures 3.2 and 3.3 (which showed the aircraft response to canard and flaperon inputs), the two surfaces appear to be fighting each other, since positive canard results in positive pitch while positive flaperon results in negative pitch. While the magnitude of the surfaces' deflection is adequate, and the aircraft response is satisfactory, containment of the surfaces' deflection became a focus of the next few design iterations.

The next eight figures show the effect of changing the weighting matrices on actuator response. The q response is not presented because all of these figures closely matched Figures 4.3 (for the variations in explicit model-following controller weighting matrices) or 4.5 (for the variations in explicit-implicit model-following controller weighting matrices). All of these plots demonstrate the inability to control actuator ramping. Repeated iterations of decreasing $\mathbf{Y}$ and $\mathbf{Q_I}$, increasing the control magnitude and rate weights (sometimes by as much as three orders of magnitude), and altering the elements of the $\mathbf{A_{Im}}$ matrix, reflected negligible or adverse affects. However, increasing $\mathbf{U_R}$ alone, to a value of 100 on each of

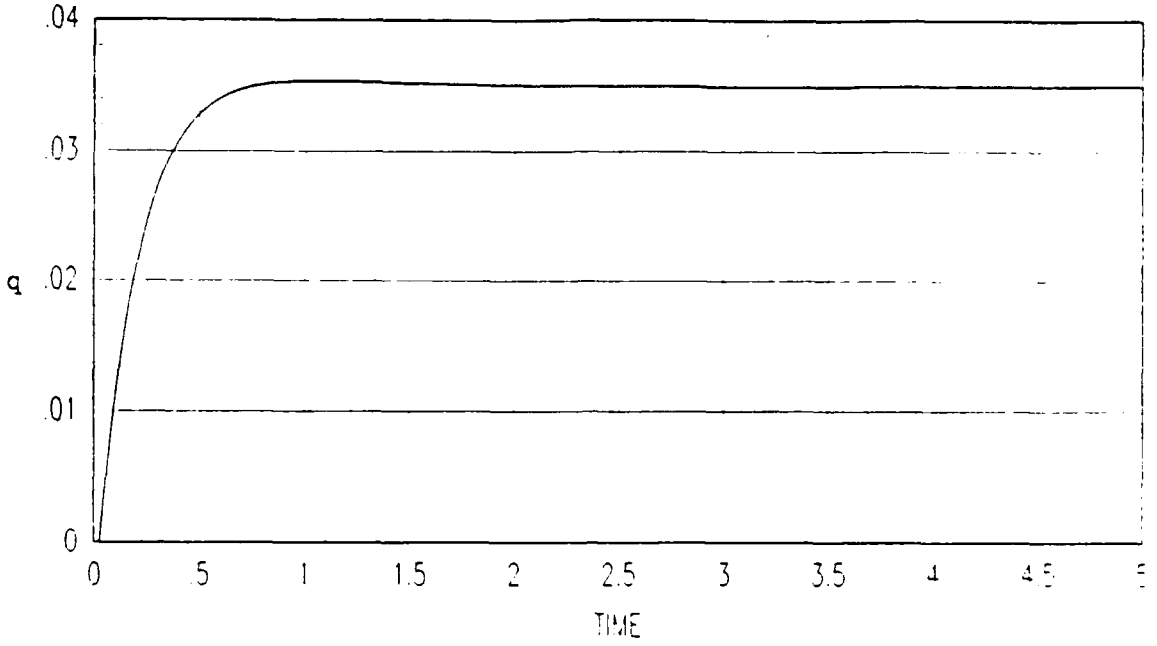



Figure 4.4. Explicit-Implicit Model-Following Controller: Pitch Rate Response

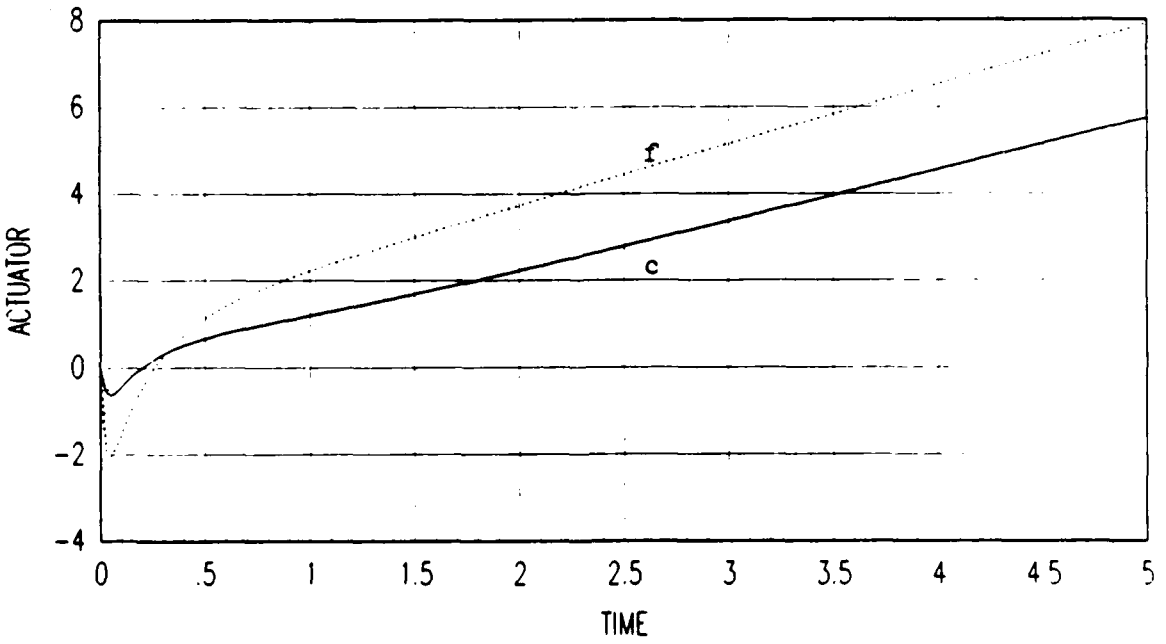



Figure 4.5. Explicit-Implicit Model-Following Controller: Actuator Response

the diagonal elements while maintaining the original weights of the other matrices, resulted in the initial negative response of both actuators to be contained, as shown in Figure 4.10. The ramp, however, remained for both surfaces, and the q response obtained a slight overshoot. As this controller will be evaluated against other truth models later, the q response is shown in Figure 4.11.

At this point it was obvious that the actuator ramping could not be controlled. The flaperon is responding in a manner opposite to what it should. The use of a transformation matrix [15] which would directly alter the sign of the commanded flaperon deflection might alleviate this problem, but that was not within the scope of this thesis. Attempts to lock either the canard or flaperon by imposing progressively larger weights proved futile. Therefore it was decided to continue with the design and evaluate the controller against failure cases and truth models incorporating first order actuator dynamics. The original controller with

$$U_R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.22}$$

was selected to be evaluated against failure cases without actuator dynamics included in the truth models. The results of this evaluation are presented in the next section.

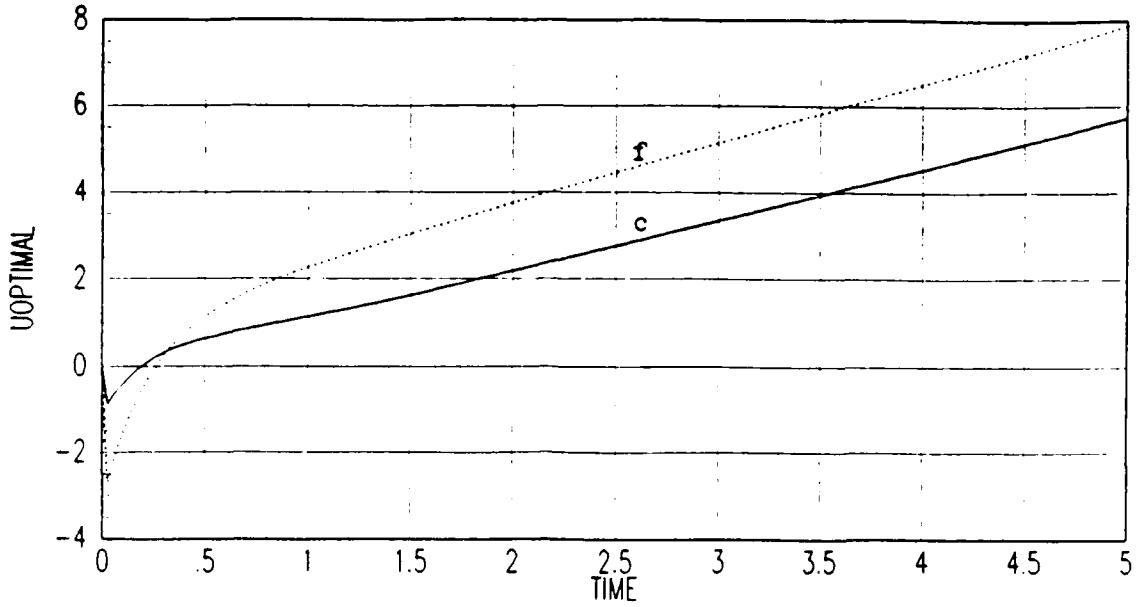



Figure 4.6. Effect Of Increasing $U_M$ To 100I on Explicit Controller; Actuator Response

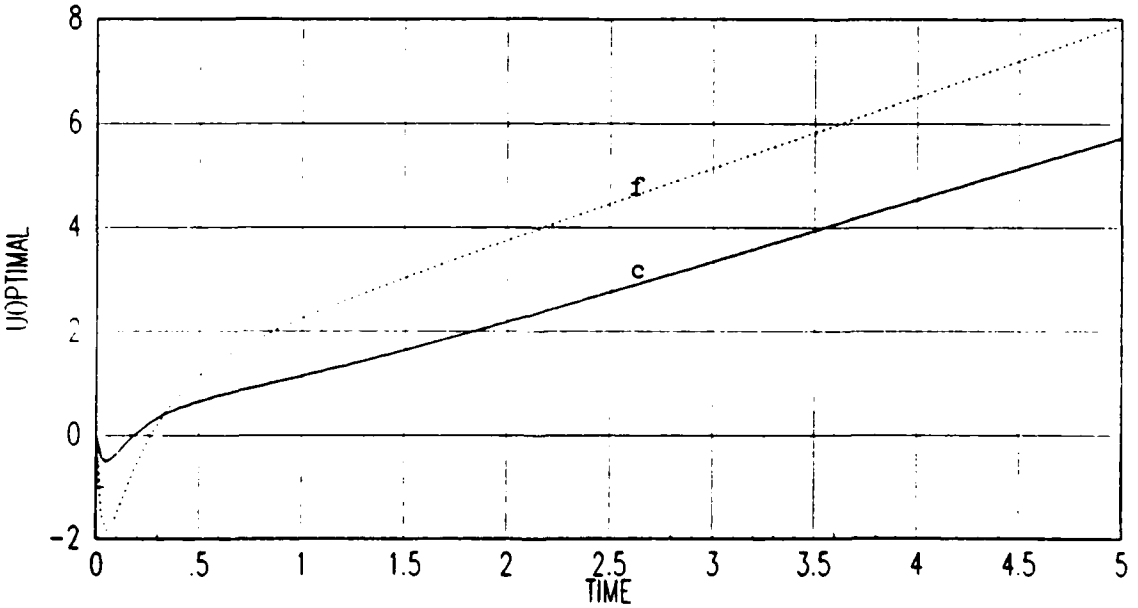



Figure 4.7. Effect Of Increasing $U_M$ and $U_R$ To 100I on Explicit Controller; Actuator Response

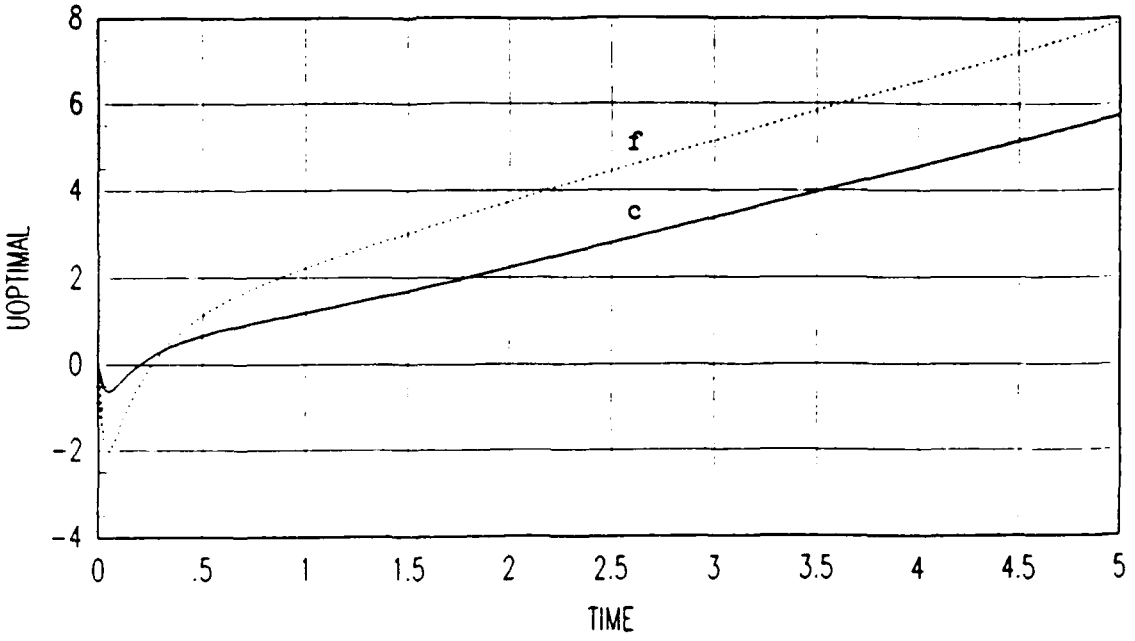



Figure 4.8. Effect Of Increasing $\mathbf{Q_I}$ To 100I on Explicit-Implicit Controller; Actuator Response

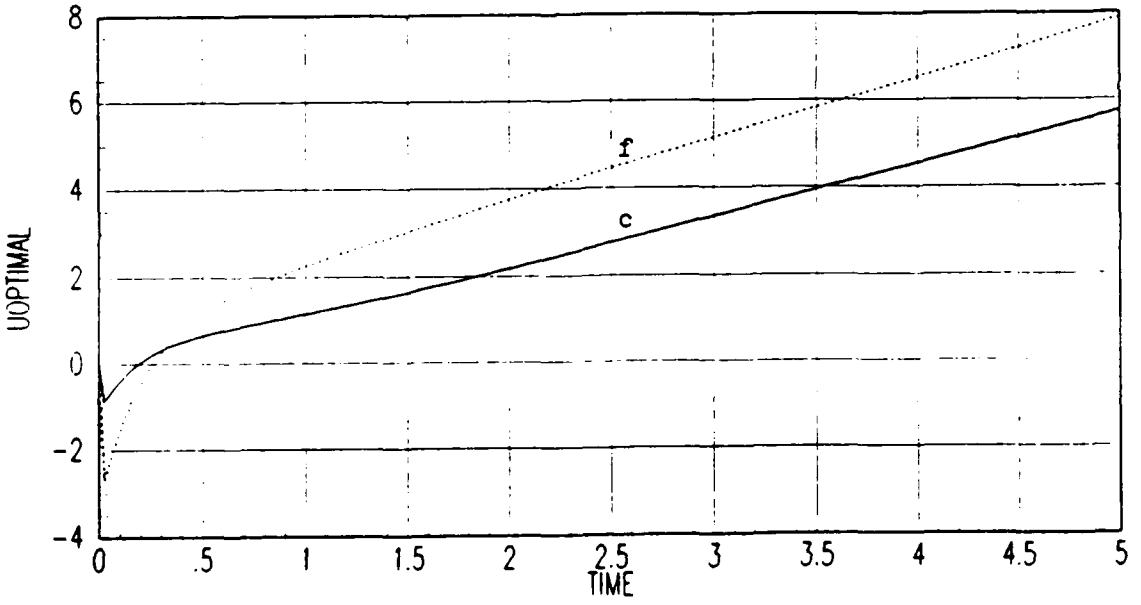



Figure 4.9. Effect Of Increasing $\mathbf{Q_I}$ and $\mathbf{R_I}$ To 100I on Explicit-Implicit Controller; Actuator Response

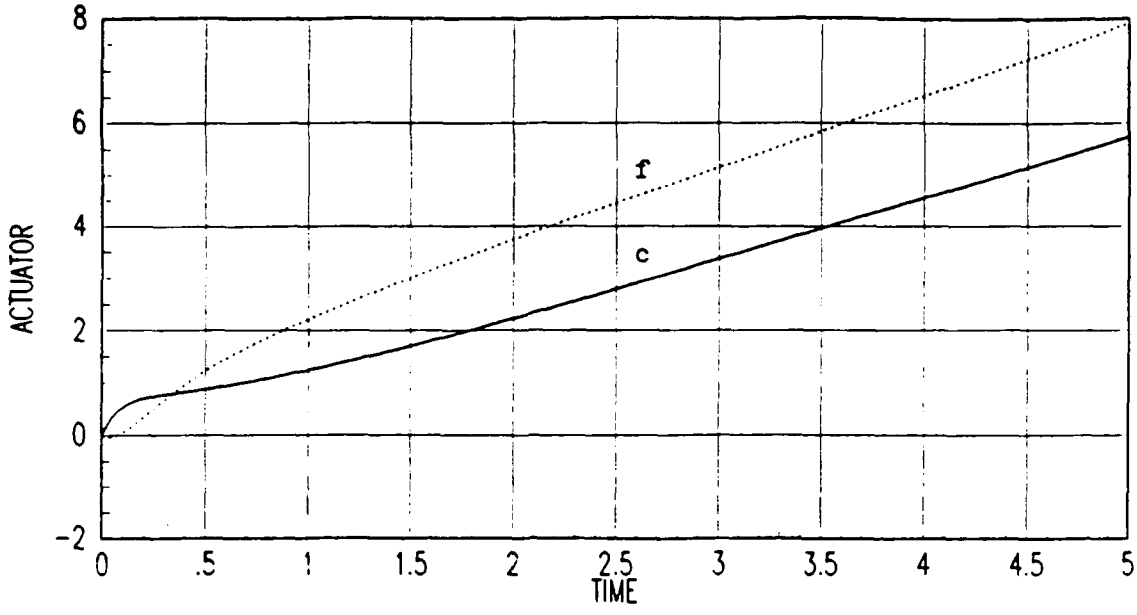



Figure 4.10. Effect Of Only Increasing $U_R$ To 100I on Explicit-Implicit Controller; Actuator Response

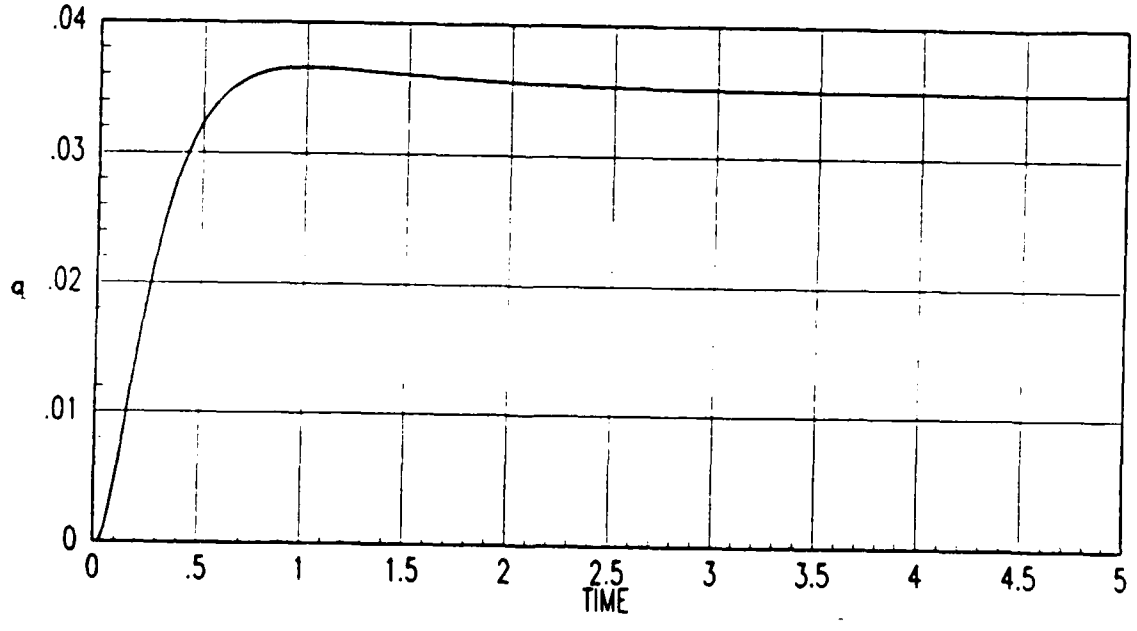



Figure 4.11. Effect Of Only Increasing $U_R$ To 100I on Explicit-Implicit Controller; Pitch Rate Response

### *4.3 Controller Evaluation Against Failure Cases: No Actuator Dynamics*

Failure cases which did not incorporate actuator dynamics were considered first to determine if the controller as designed was adequate, and to help identify potential problems which might occur when actuator dynamics were incorporated. The first failure case which was considered was TM03 (25 percent canard failure). As Figures 4.12 through 4.14 demonstrate, the controller is not robust to canard failures. (Figure 4.14 is included to show more detail of the actuator response.) The aircraft begins to diverge at 3.5 seconds, with excessive actuator displacements. Both actuators have negative deflections, as the canard tries to prevent the violent pitch up and the flaperon continues to exhibit contrary performance. The effect of adding actuator limits (Figure 4.15) is to decrease the divergence time to 2.5 seconds, as this is when position limits are exceeded. Adding antiwindup compensation (Figures 4.16 and 4.17) moderately improves performance (note the y axis scale change on Figure 4.16), but the aircraft is still divergent.

Evaluations of the controller against TM06, TM16, and TM19 (25 percent outer flap loss, 25 percent outer flap and elevator loss, and canard fail-to-trail, respectively) were much more satisfactory (Figures 4.18 through 4.23). Pitch rate response follows the command model perfectly. Actuator response continues to ramp, with some increase in flaperon deflection, but limits were not approached so these responses were considered satisfactory.

Evaluating the controller against TM23, outer flap and elevator fail-to-trail, is not satisfactory (Figures 4.24 and 4.25). With actuator limits and antiwindup compensation imposed, the aircraft diverges immediately after flaperon saturation at 3.3 seconds. This appears to be caused by the the canard, which is trying to counteract the flaperon's adverse affect, continuing its ramp. At 3.7 seconds the canard begins to respond favorably, but by this time the aircraft has diverged. It should be noted that all cases would result in actuator saturation and aircraft divergence if the simulation were performed for an appropriately long time duration,

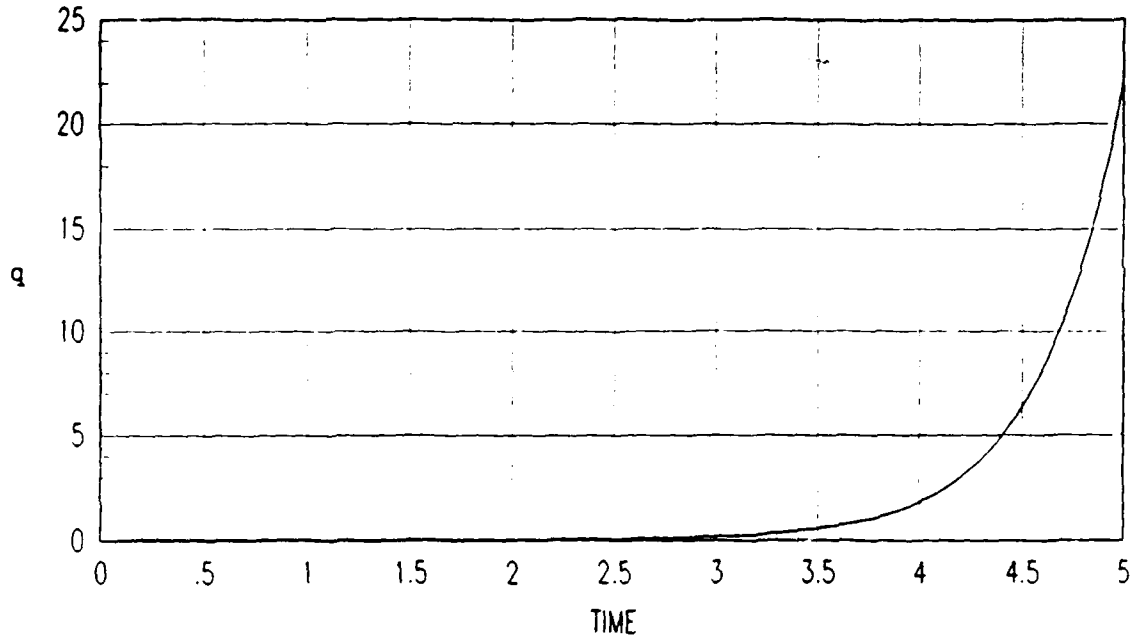



Figure 4.12. TM03 Pitch Rate Response

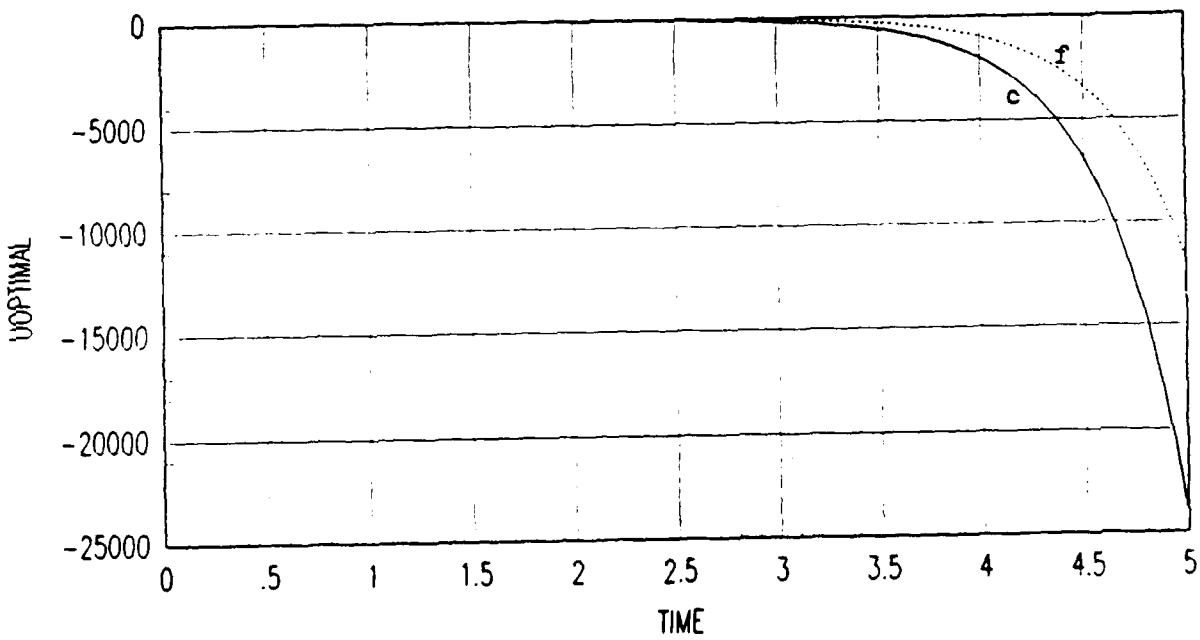



Figure 4.13. TM03 Actuator Response - 5 Seconds

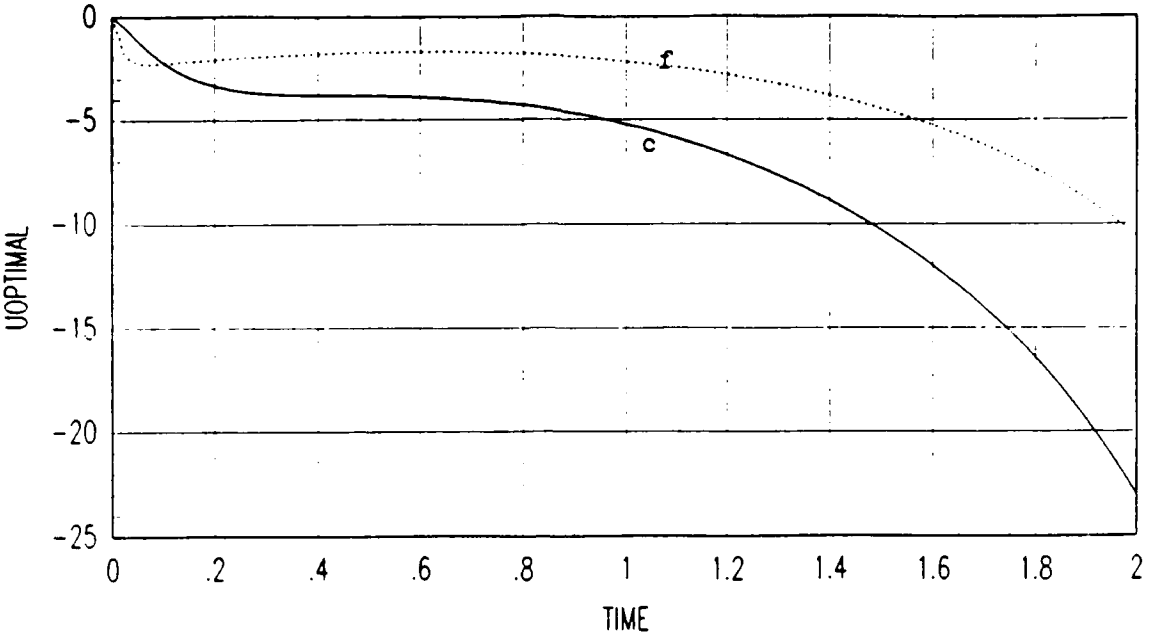



Figure 4.14. TM03 Actuator Response - 2 Seconds

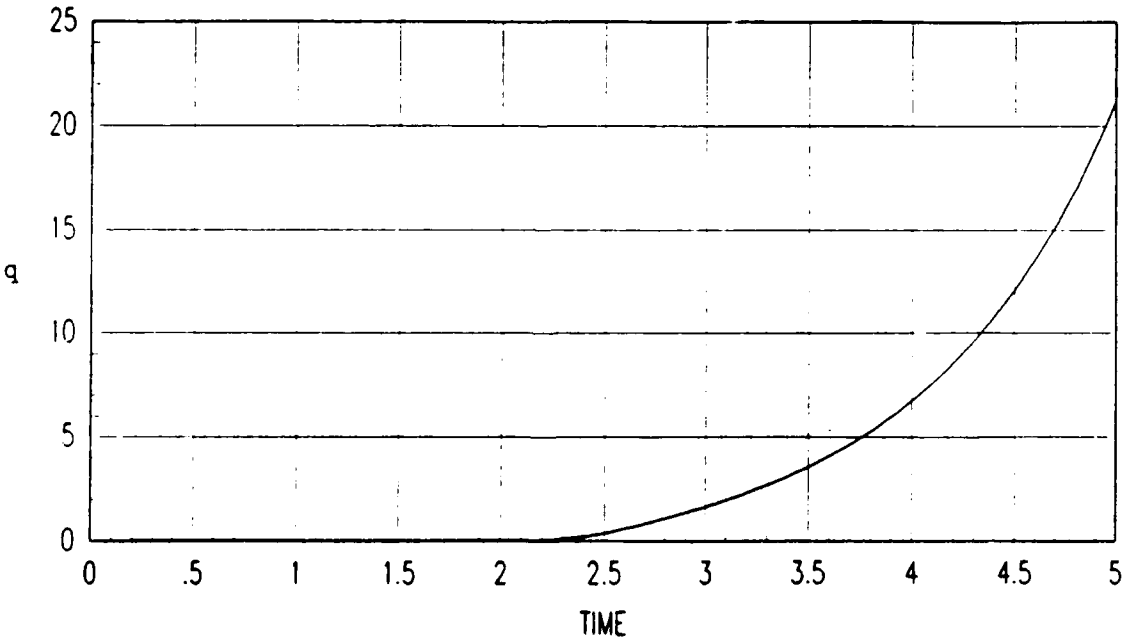



Figure 4.15. TM03 With Actuator Limits, Pitch Rate Response

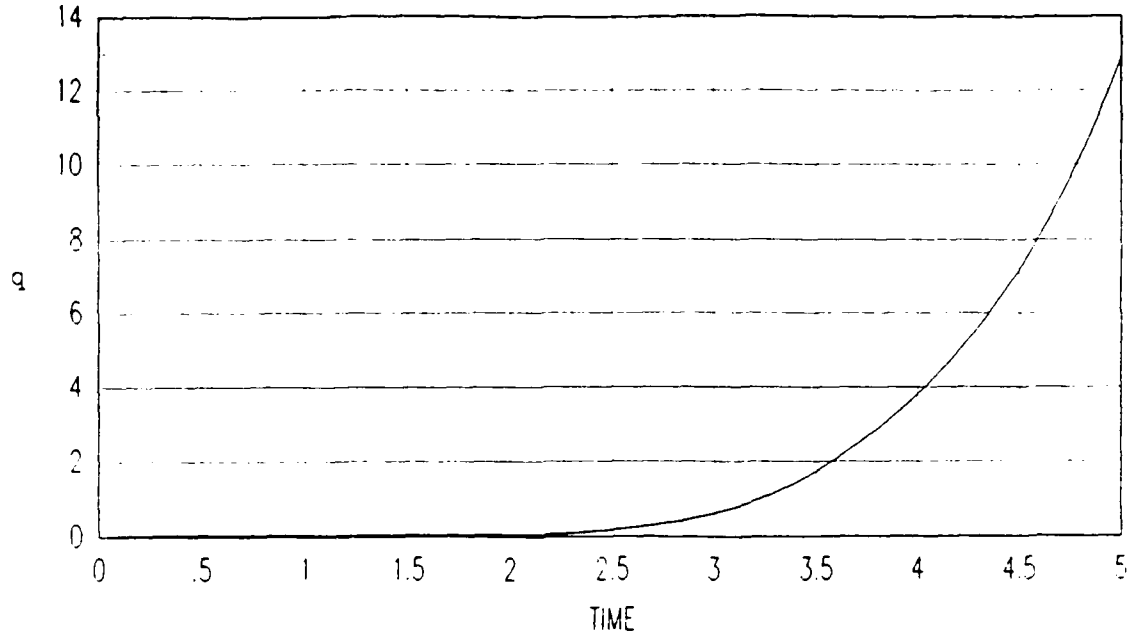



Figure 4.16. TM03 With Actuator Limits, Antiwindup Compensation On, Pitch Rate Response

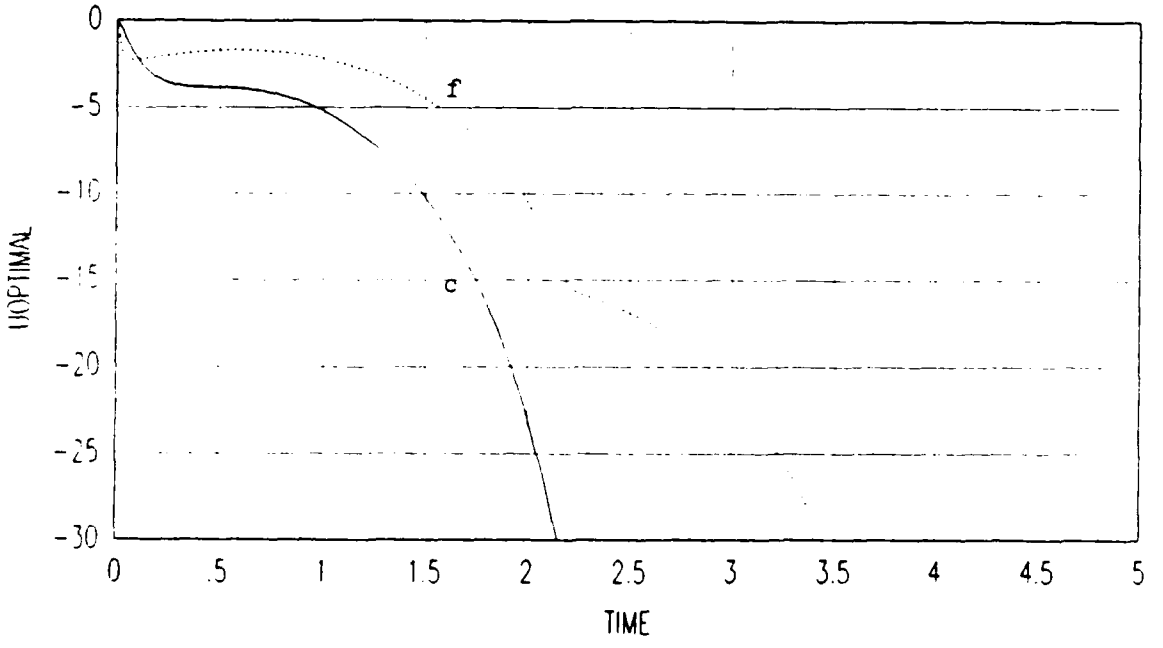



Figure 4.17. TM03 With Actuator Limits, Antiwindup Compensation On, Actuator Response

but it was judged that such prolonged step inputs would not be sustained while the aircraft was in air combat mode.

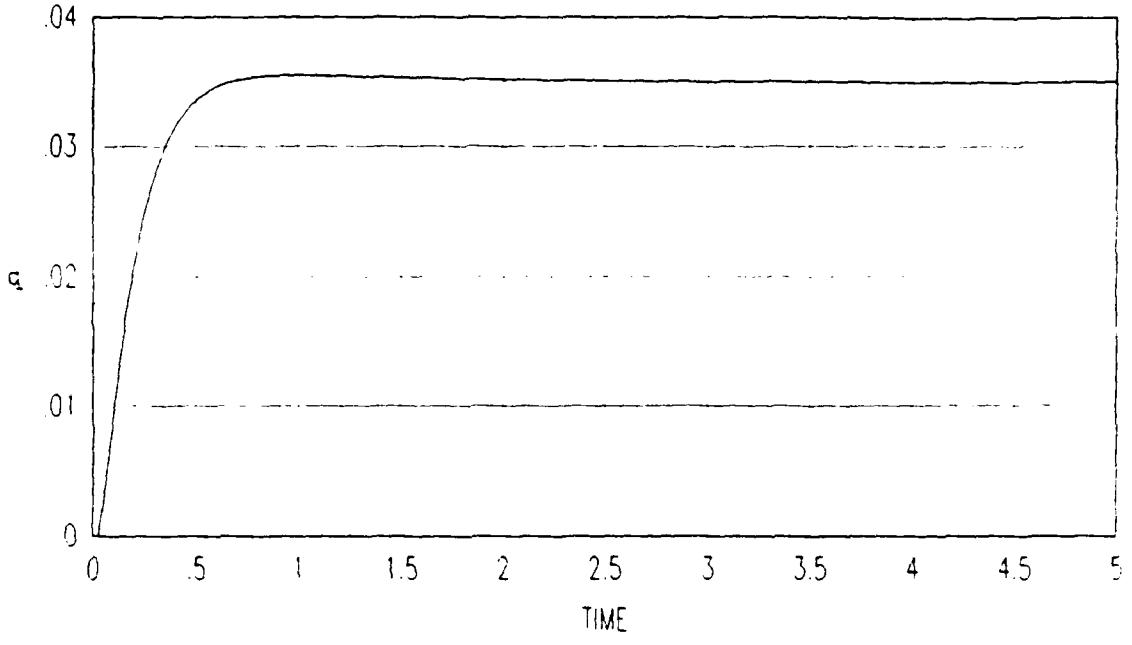



Figure 4.18. TM06 Pitch Rate Response

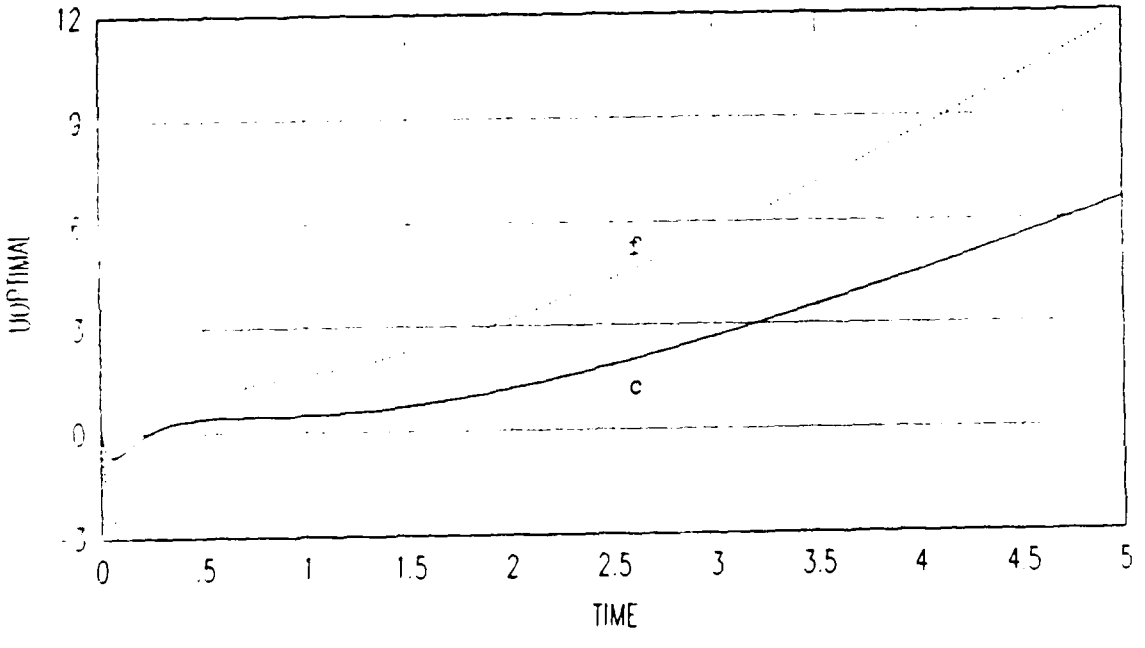



Figure 4.19. TM06 Actuator Response

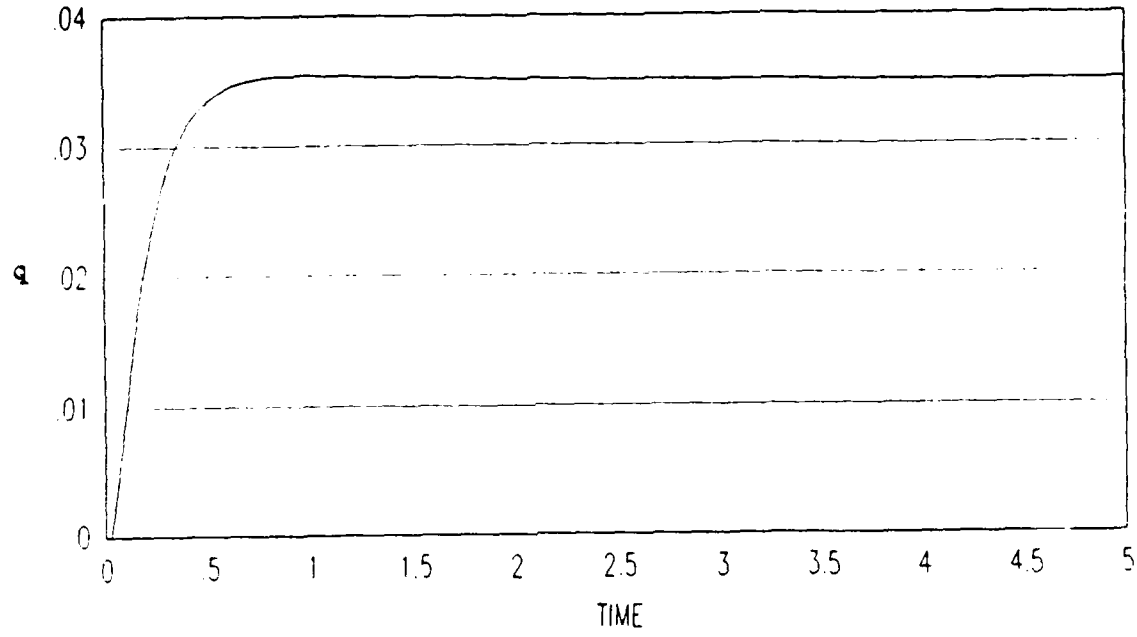



Figure 4.20. TM16 Pitch Rate Response

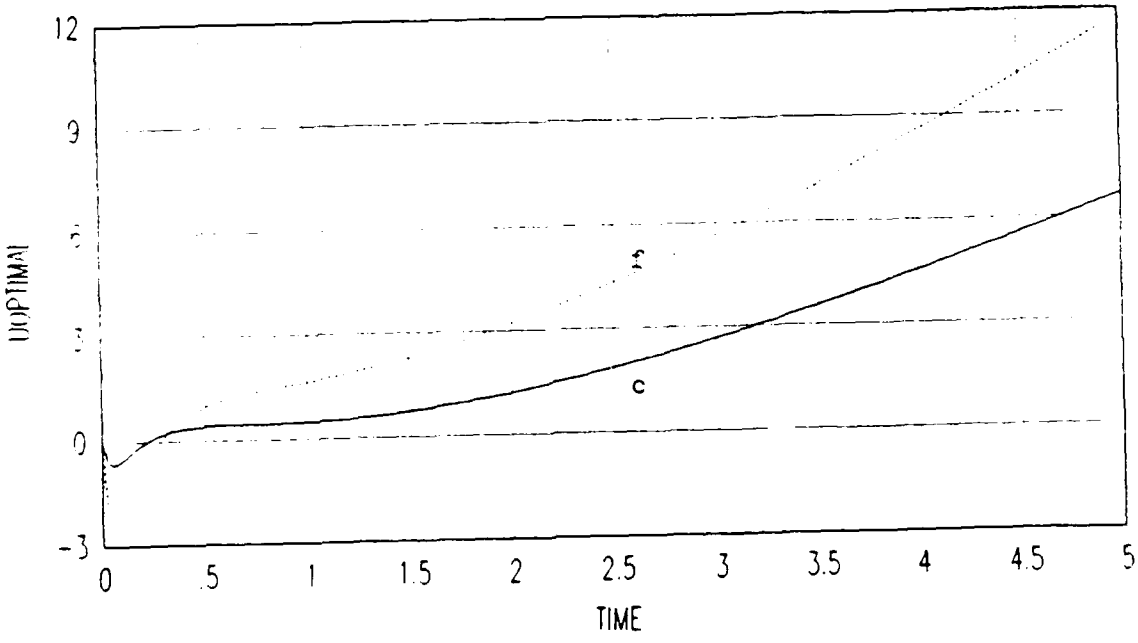



Figure 4.21. TM16 Actuator Response

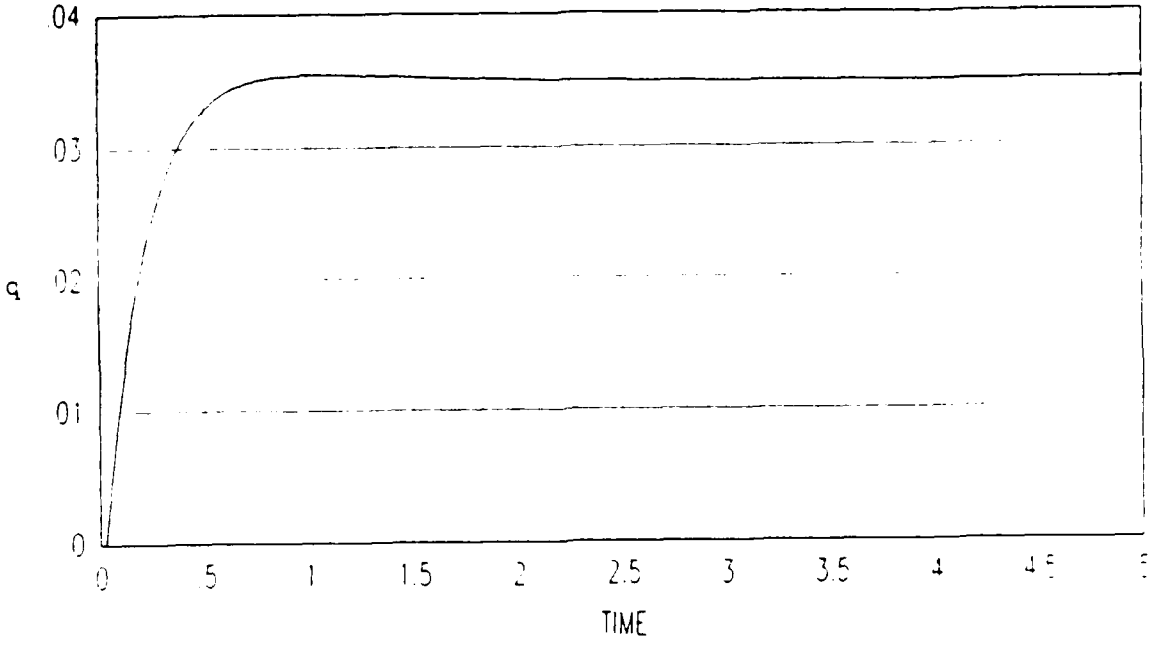



Figure 4.22. TM19 Pitch Rate Response

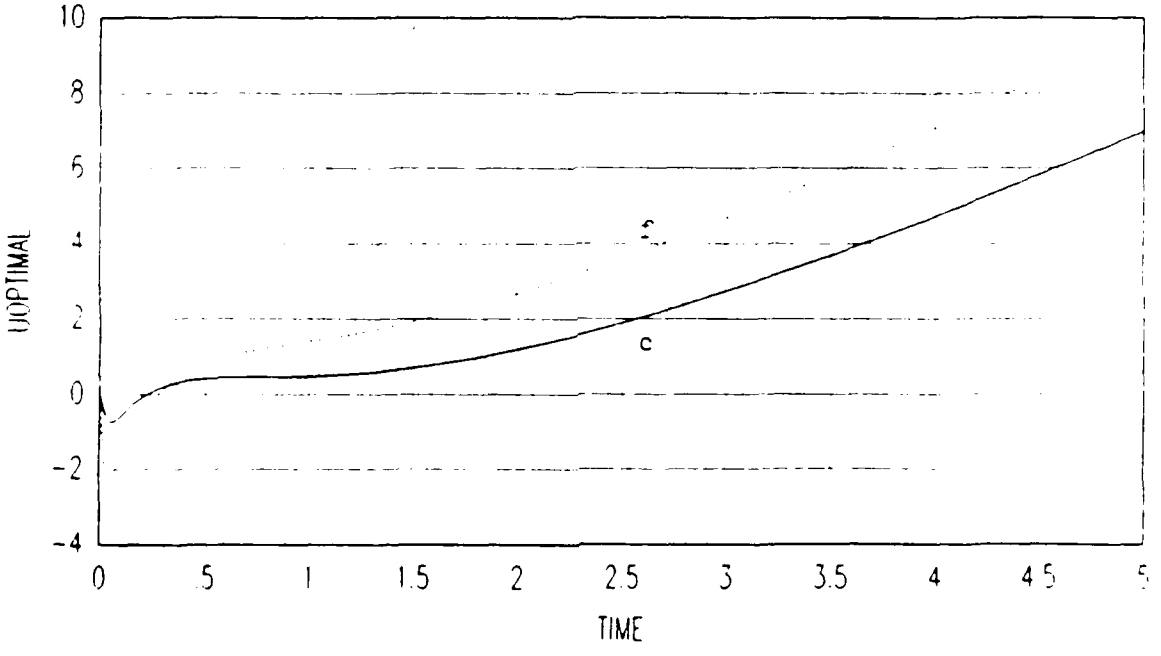



Figure 4.23. TM19 Actuator Response

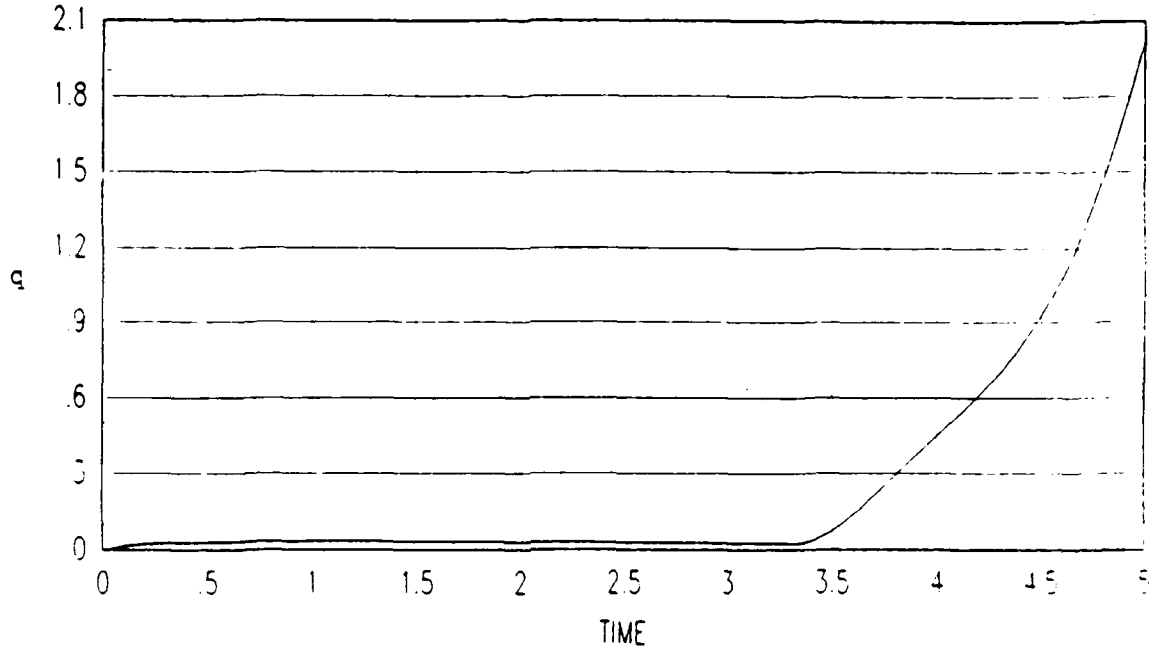



Figure 4.24. TM23 Pitch Rate Response

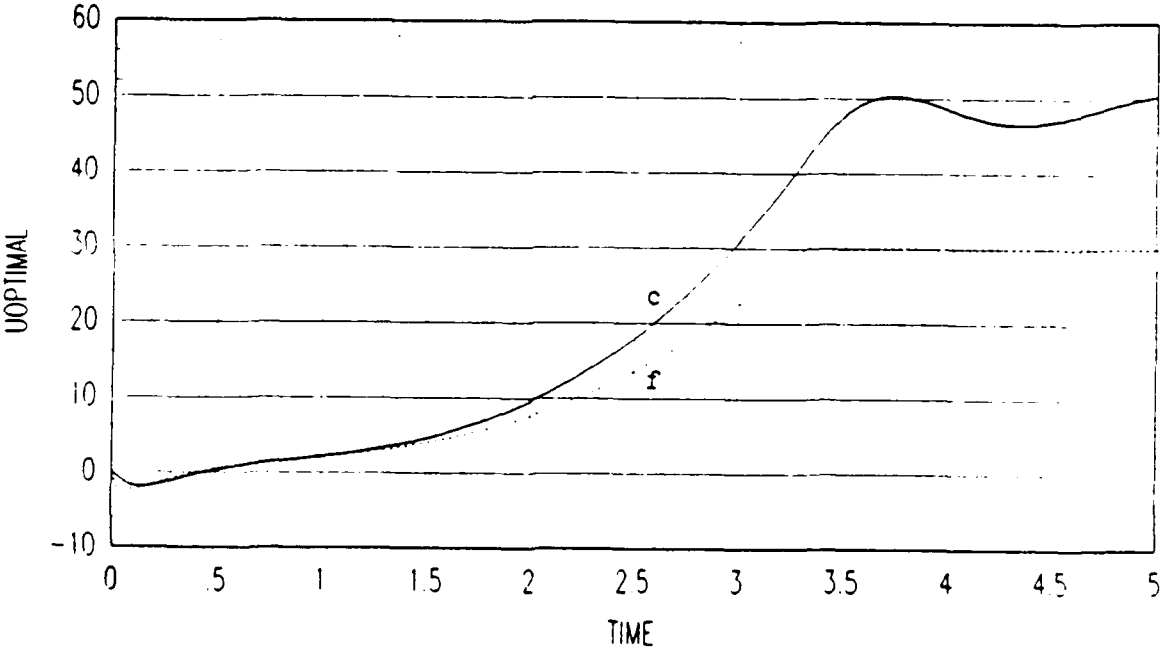



Figure 4.25. TM23 Actuator Response

### *4.4 Deterministic Controller Evaluation Against Higher Order Truth Models*

The next step of the design process was to evaluate the controller against the truth models which incorporated first order actuator dynamics. The controller evaluated in the last section, with the diagonal elements of $\mathbf{U_R}$ set to 1, proved to have poor performance when compared to the nominal truth model with actuator dynamics (TM101). As Figures 4.26 and 4.27 show, the q, canard, and flaperon responses exhibit higher order dynamics, which is deemed unsatisfactory. Adding actuator limits and antiwindup compensation did not affect the performance. The oscillations of the actuators leads one to suspect that increasing the control rate weighting matrix might provide improved performance. This turned out to be the case. When the controller with $\mathbf{U_R} = 100\mathbf{I}$ was evaluated against TM101, the performance approached that of the command model (Figures 4.28 and 4.29). Changing the other weighting matrices had negligible effect on performance, as noted in the lower order case. Adjusting the implicit model, however, did result in improved performance. As Figures 4.30 and 4.31 reveal, decreasing the implicit model matrix entries from -5 to -.5 (thereby slowing the implicit response characteristics) results in a smoother q response. The rise time is approximately .1 seconds longer, but this is deemed acceptable. This performance was consistent with that found when both controllers were evaluated against TM106, TM116, and TM119 (25 percent outerflap loss with first order actuators, 25 percent outerflap and elevator loss with first order actuators, canard fail-to-trail with first order actuators, respectively). These plots are not presented here due to their similarity with the response evaluated against TM101.

When the CGT/PI controller was evaluated against TM103 (25 percent canard loss with first order actuators), the response was once again unsatisfactory (Figures 4.32 and 4.33). Strangely enough, with actuator dynamics added to this failure case, the aircraft exhibited a negative pitch rate divergence, as opposed to the positive divergence without actuator dynamics (TM03). The actuator response

was also opposite of that exhibited in the TM03 case. Repeated attempts to control the divergence, by adjusting $\mathbf{U_R}$ and $\mathbf{A_{Im}}$, had an adverse effect. This was expected, as the incoporation of actuator dynamics in the truth model increased the severity of the control problem.

The final case to be considered, that of outer flap and elevator fail-to-trail (TM123), was also divergent, as in TM23. Again, this was expected. As shown in Figures 4.34 and 4.35, the aircraft exhibits a severe oscillation between 4 and 5 seconds, caused in part by the flaperon saturation. As with the previous case of canard failure, this failure condition proved to be uncontrollable. Varying the weighting matrices and the implicit model had negligible effect on the aircraft performance.

At this point it was decided to continue with the design, and evaluate the CGT/PI/KF controller against the failure conditions with actuator dynamics added. The flight conditions with lower order actuators were not to be evaluated further. The purpose of evaluating these conditions initially was to help identify the effect of adding actuator dynamics to the truth models. It has been demonstrated that the initial controller was not suitable for the higher order truth models, but that increasing $\mathbf{U_R}$ from $\mathbf{I}$ to $100\mathbf{I}$ improved the performance of the controller evaluated against the truth models which were previously controllable. The truth models which were divergent (TM03 and TM123), however, remained divergent.

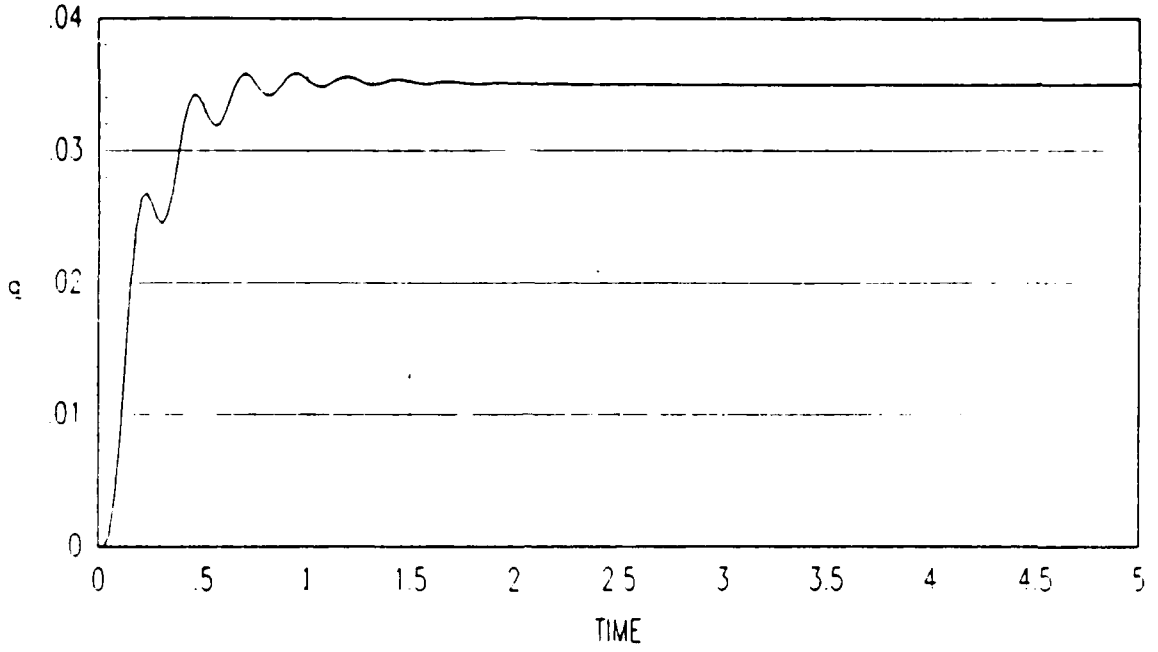



**Figure 4.26. TM101 Pitch Rate Response, $\mathbf{U_R = I}$**

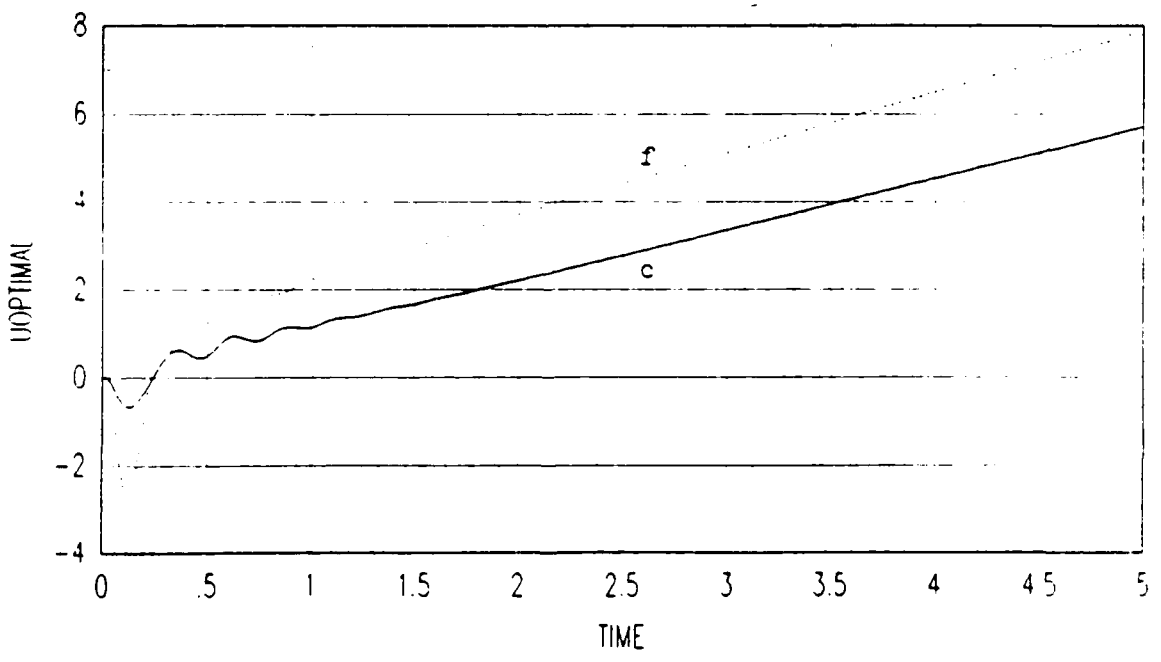



**Figure 4.27. TM101 Actuator Response, $\mathbf{U_R = I}$**

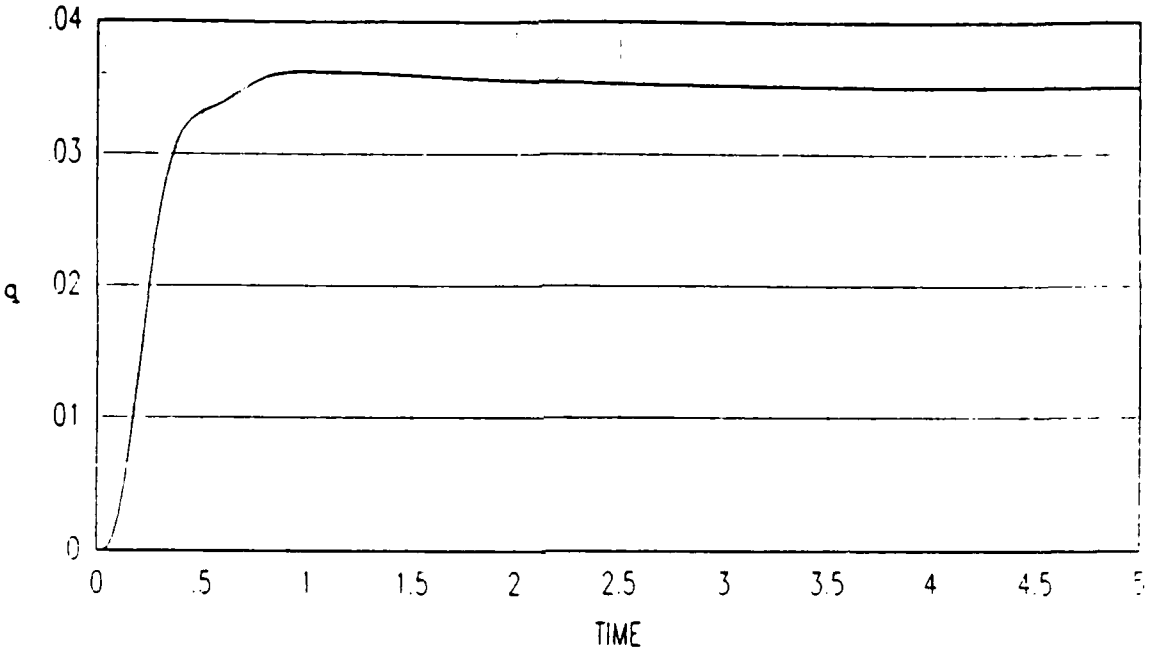



Figure 4.28. TM101 Pitch Rate Response, $\mathbf{U_R = 100I, A_{Im} = -5I}$

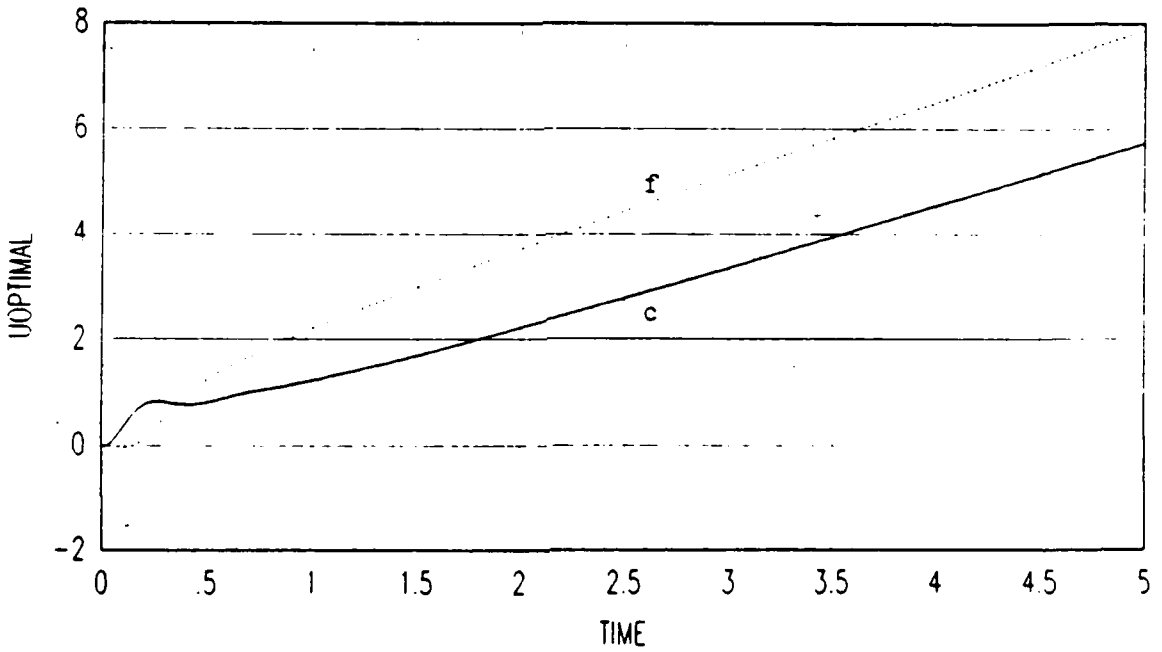



Figure 4.29. TM101 Pitch Rate Response, $\mathbf{U_R = 100I, A_{Im} = -5I}$

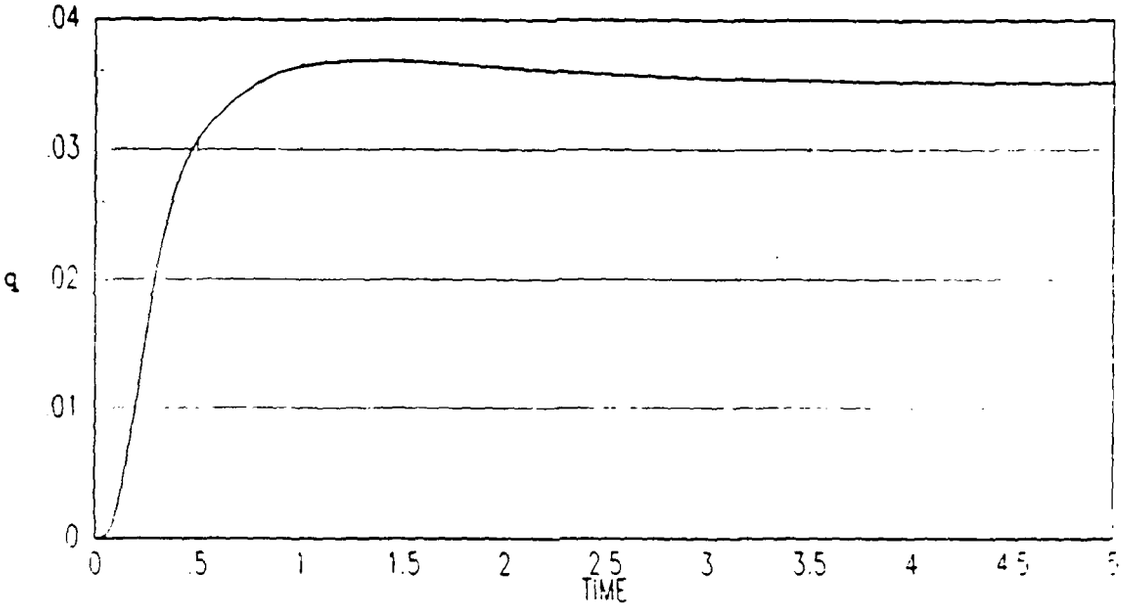



Figure 4.30. TM101 Pitch Rate Response, $\mathbf{U_R = 100I, A_{Im} = -.5I}$

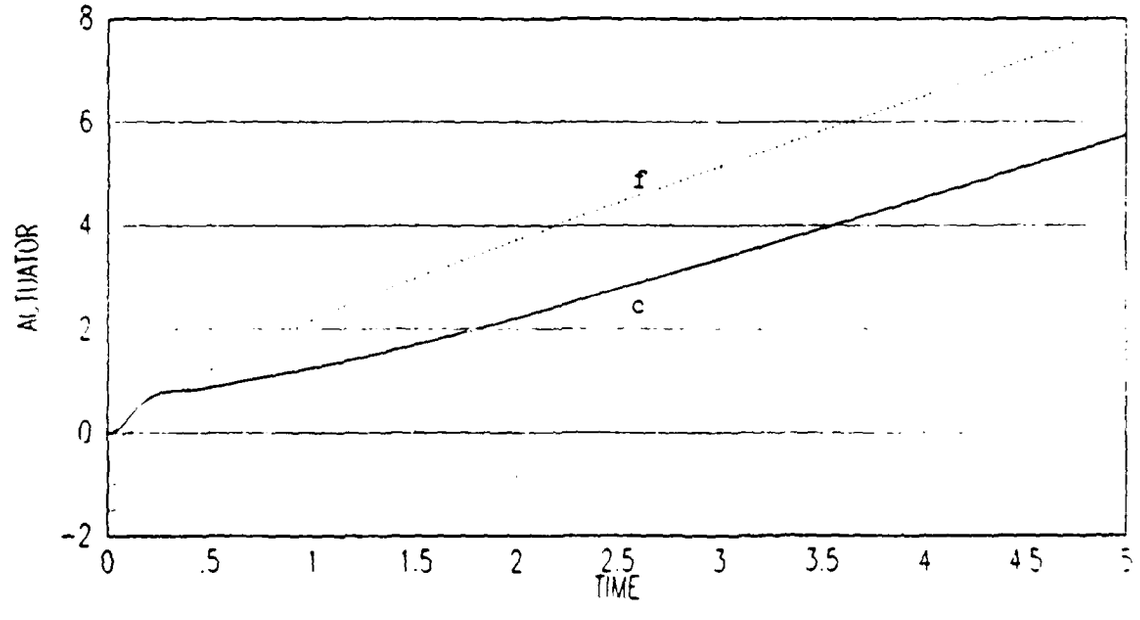



Figure 4.31. TM101 Pitch Rate Response, $\mathbf{U_R = 100I, A_{Im} = -.5I}$

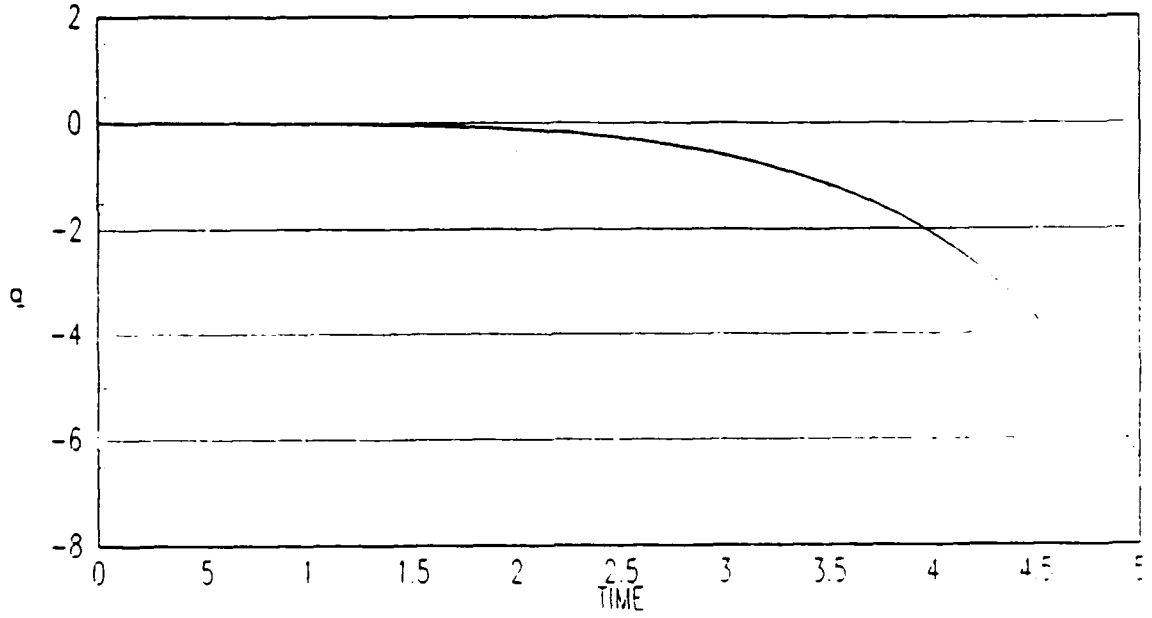



Figure 4.32. TM103 Pitch Rate Response, $\mathbf{U_R = 100I, A_{Im} = -.5I}$

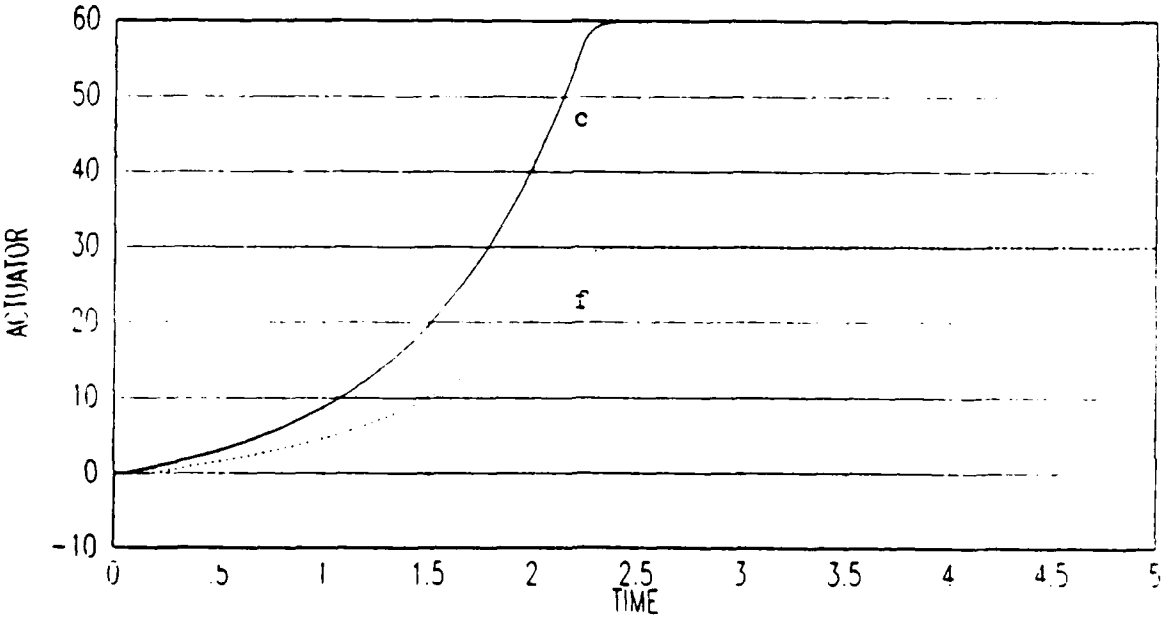



Figure 4.33. TM103 Actuator Response, $\mathbf{U_R = 100I, A_{Im} = -.5I}$

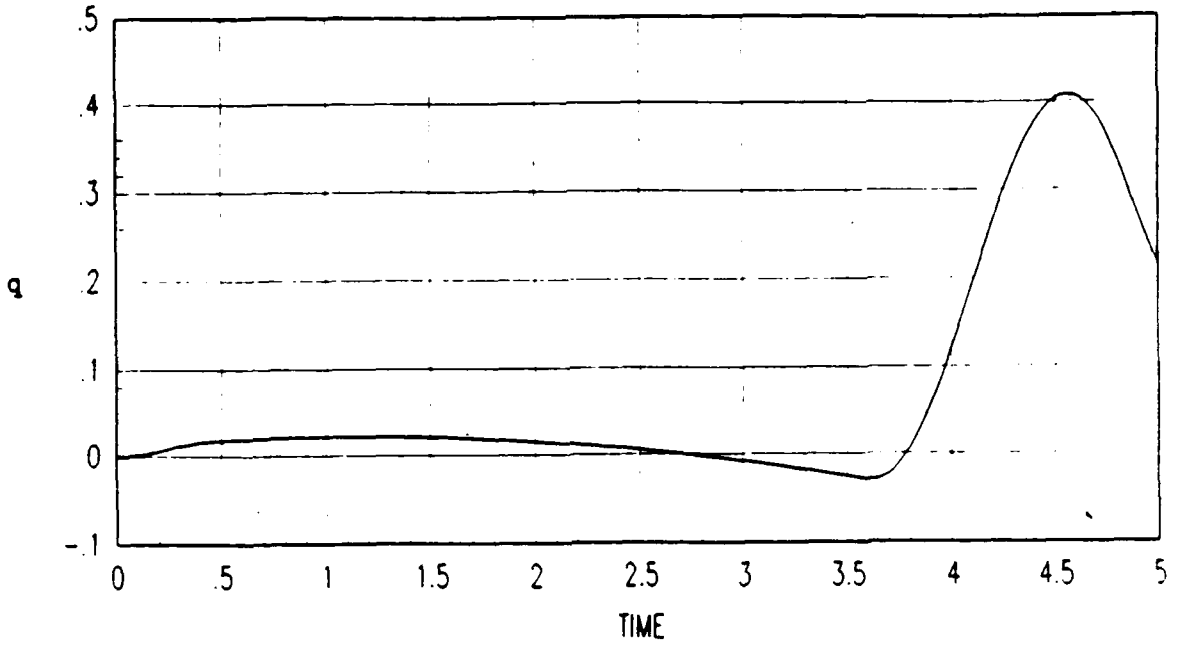



Figure 4.34. TM123 Pitch Rate Response, $\mathbf{U_R = 100I, A_{Im} = -.5I}$

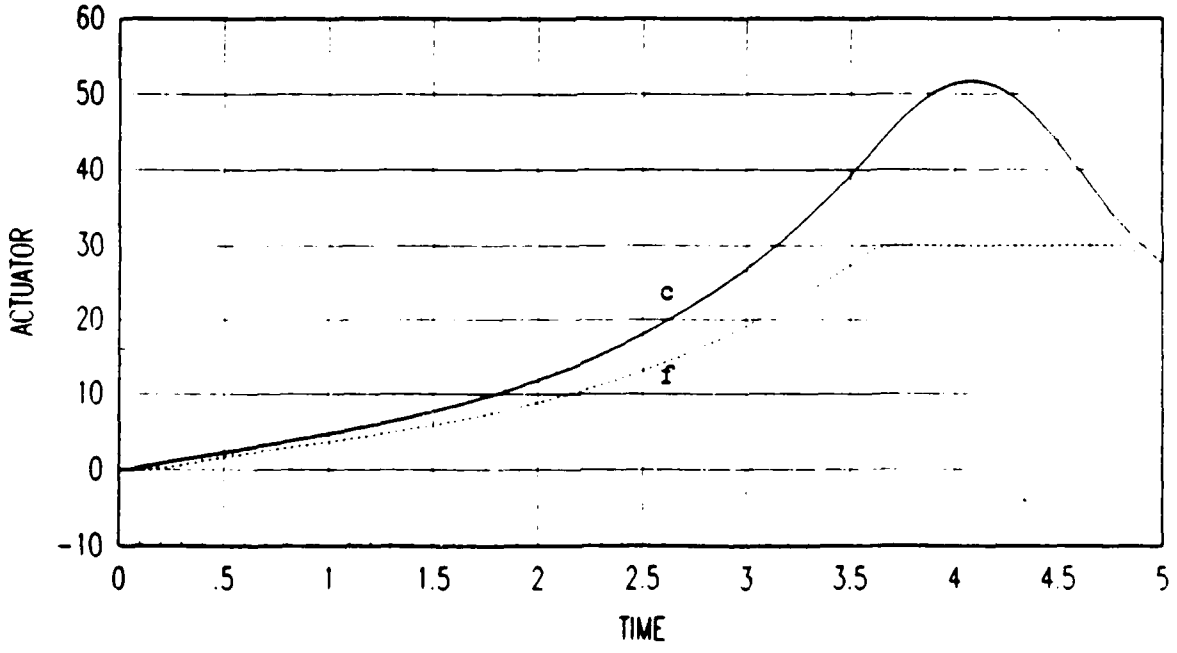



Figure 4.35. TM123 Actuator Response, $\mathbf{U_R = 100I, A_{Im} = -.5I}$

## 4.5 *CGT/PI/KF Controller Evaluation*

The final step in the flight controller design process was to design a Kalman filter and combine it with the previously designed CGT/PI deterministic controller. Certainty equivalence states that this is a valid design approach, due to the separation of the deterministic control problem from the stochastic nature of the problem. However, while the design evaluated at nominal conditions should perform in an acceptable manner, stability robustness of the system may be degraded due to the incorporation of the Kalman filter. It is therefore critical to evaluate the closed loop control system against the failure conditions as well as the nominal design conditions.

In designing the Kalman filter, the design model **A** and **B** matrices are the same as in Section 4.2. The measurement matrix **H** is identical to the **C** matrix, that is, measurements are taken of the controlled outputs, $\theta$ and q. The **G** matrix is a 4 by 4 identity matrix (to allow for the possibility of LTR tuning later), and the noise covariance matrices were initially chosen as

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 \end{bmatrix} \tag{4.23}$$

$$\mathbf{R} = \begin{bmatrix} .476\,E-5 & 0 \\ 0 & .322\,E-4 \end{bmatrix} \tag{4.24}$$

The initial **Q** value was chosen based on work performed on the STOL F-15, operating in a similar flight environment [9]. Reasonable **R** values for the CRCA were unavailable, so a value identical to that used for the AFTI F-16 [5] and the

STOL F-15 was used. The resulting Kalman filter gain matrix is

$$\mathbf{K_f} = \begin{bmatrix} -3.1561 & -0.5074 \\ 43.4047 & 14.5297 \\ 0.0672 & 0.0164 \\ 0.1106 & 0.5679 \end{bmatrix} \tag{4.25}$$

Since the truth models used in this thesis are deterministic, a covariance analysis was meaningless. An attempt to incorporate the Dryden wind model into the nominal truth model, via Pogoda's methodology [16], proved fruitless. Various parameters of the CRCA, particularly the body axis derivatives, were unavailable, which precluded modeling wind gusts and turbulence effects on the aircraft. Also, the use of parameters from other fighter aircraft operating in similar flight conditions was judged to be unrealistic. Therefore, evaluation of the filter independent of the CGT/PI controller was not pursued. Because the CGT/PI/KF controller will be evaluated against deterministic truth models, and the ability to change **Q** is available via LTR tuning, the lack of a covariance analysis will not be severely detrimental to the overall controller evaluation.

Incorporating the Kalman filter with the deterministic CGT/PI controller (where $\mathbf{U_R} = 100\mathbf{I}$ and $\mathbf{A_{Im}} = -.5\mathbf{I}$), and evaluating the resultant controller against the nominal truth model, TM101, shows the closed-loop system to have desirable performance characteristics (Figures 4.36 and 4.37). The rise time and settling time are somewhat longer than the deterministic controller's (as expected due to the additional dynamic lag of the filter), and there is a slight overshoot, but the response is still acceptable [1]. The actuator deflections remained essentially the same.

The first failure cases considered were the ones that were stable when evaluated against the CGT/PI controller. As shown in Figures 4.38 through 4.43, the responses approximate that of the command model. TM106 and TM116 (25 percent outer flap failure and 25 percent outer flap and elevator failure, respectively)

have a q response which exhibits a 17 percent overshoot, and a steady-state error of approximately .004 radians, but for a battle damaged aircraft this is considered acceptable. Also, the actuator deflections are still well within bounds, although the ramping effect has continued. TM119, canard fail-to-trail, has maintained reasonable performance (Figures 4.42 and 4.43), with a moderate overshoot and an increase in settling time of approximately .5 seconds.

Although the aircraft response for the previous cases was considered acceptable, Loop Transmission Recovery was pursued to try to decrease the overshoot of TM116. Referring to Equation (2.85), the **V** matrix was set to identity and the scalar q value (not to be confused with pitch rate) was set to .1, which is two orders of magnitude greater than the nominal Kalman filter's **Q** value. As Figure 4.44 shows, the pitch rate overshoot is moderately improved. The actuator response is nearly identical to that of Figure 4.41, and is not presented here. The effect of LTR tuning on the nominal plant will be shown later.

The effect of adding the Kalman filter to the controller, and evaluating it against TM103 and TM123, was quite surprising. In both cases the system response *improved*, as shown in Figures 4.45 through 4.48. This response was not expected, as the Kalman filter typically reduces the robustness of the system. However, in this case, both of the previously unstable flight conditions have become stable, albeit with severely degraded performance from that of the nominal flight condition. TM103, the 25 percent canard failure case, has a peak q of .022 rad/sec, with a final value of .019 rad/sec. TM123, outer flap and elevator fail-to-trail, is somewhat better, with a peak of .03 rad/sec and a steady-state value of .026. This unexpected response is discussed in the next section.

To try to improve the performance of these two failure conditions, LTR tuning was performed. Figure 4.49 shows that a scalar q factor of .1 results in a much smoother response for TM103, but the system still has a significant steady-state error. Increasing q another order of magnitude to 1.0 (Figure 4.50) results in the

system ramping up towards the desired value of pitch rate, but the plot does not show whether this value is obtained, or if the ramp continues. Applying q=1.0 to TM123 (Figure 4.51) results in a relatively small steady-state error and acceptable rise time. Applying this same q value to TM101 (to determine any adverse affects of LTR tuning on the nominal flight condition) shows an almost imperceptible increase in rise time, as shown in Figure 4.52. Therefore, in this case LTR tuning is an appropriate robustness enhancement technique.

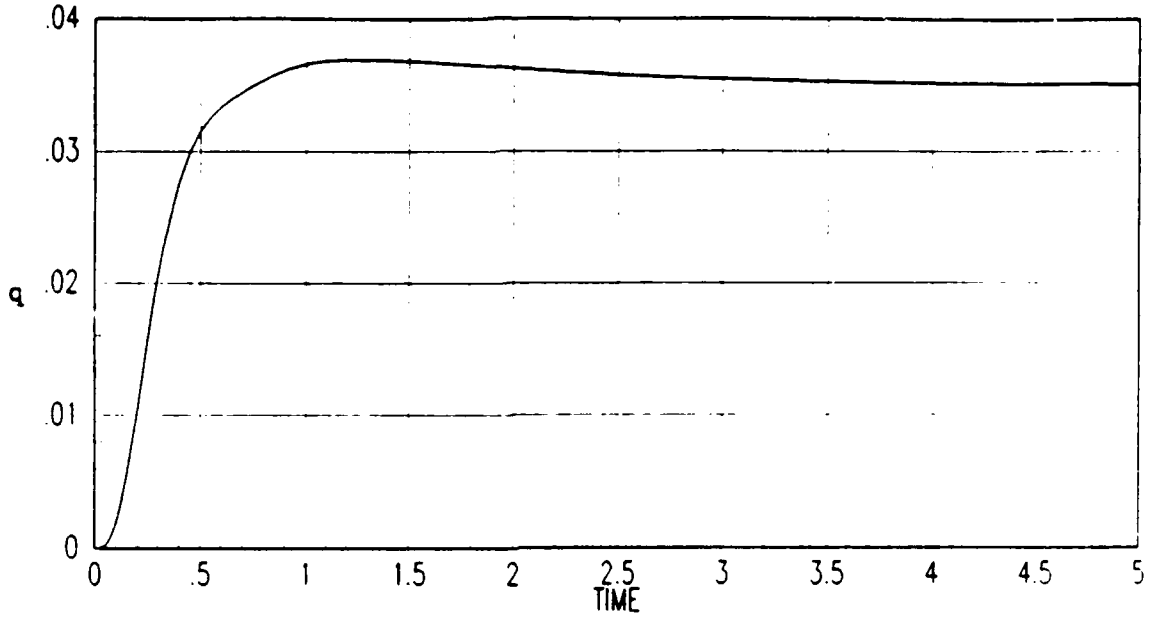



Figure 4.36. CGT/PI/KF Pitch Rate Response, TM101

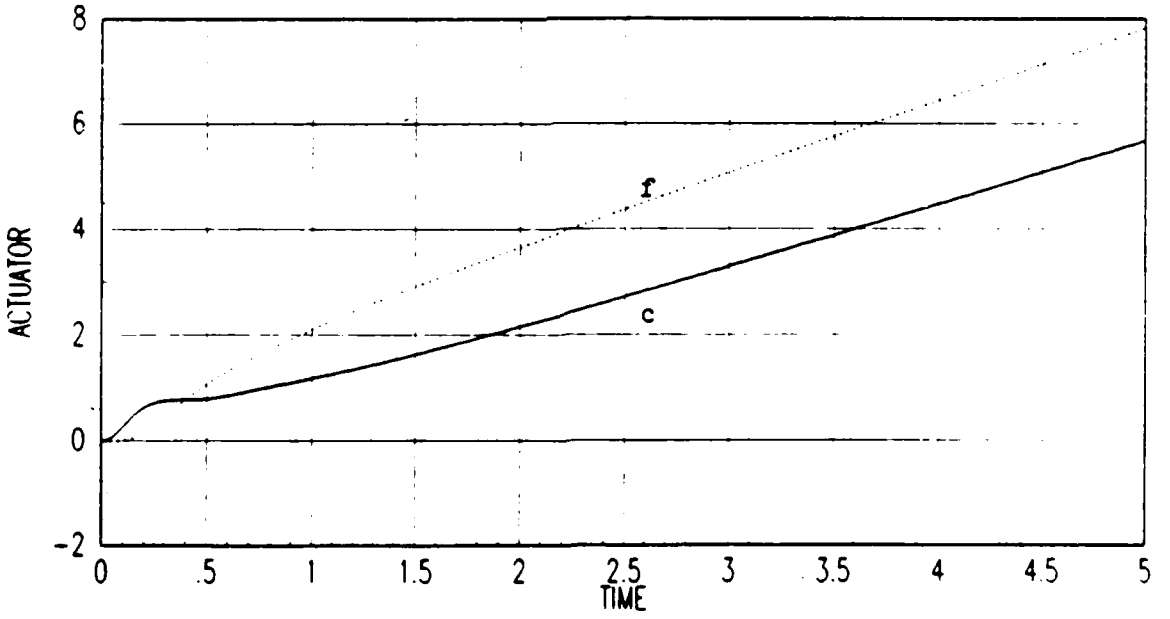



Figure 4.37. CGT/PI/KF Actuator Response, TM101

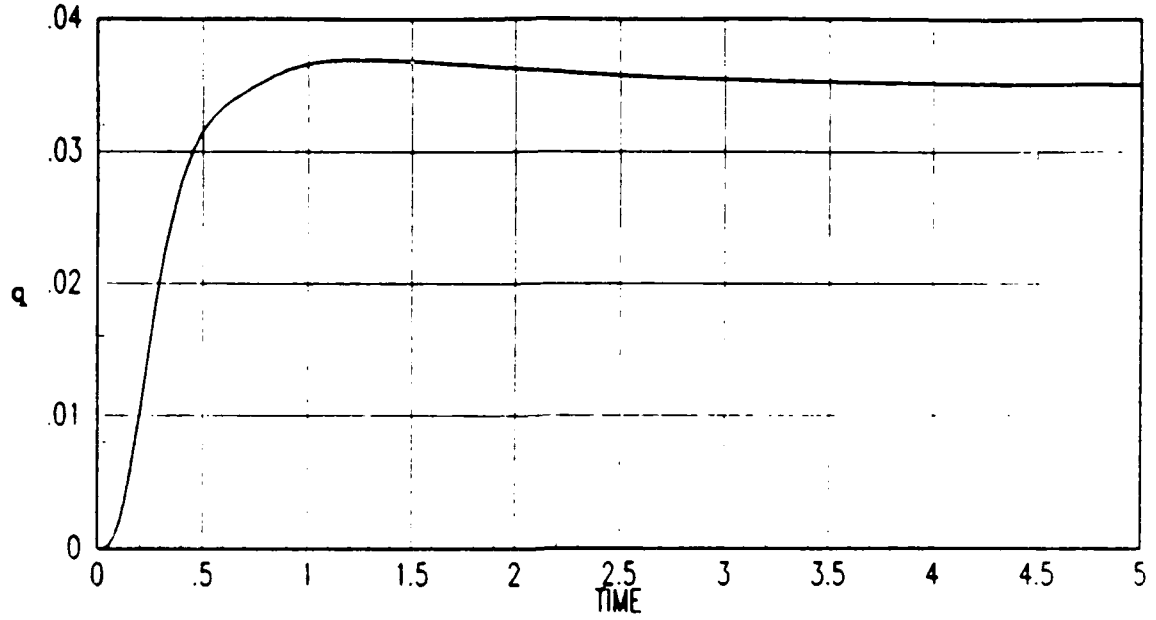



Figure 4.36. CGT/PI/KF Pitch Rate Response, TM101

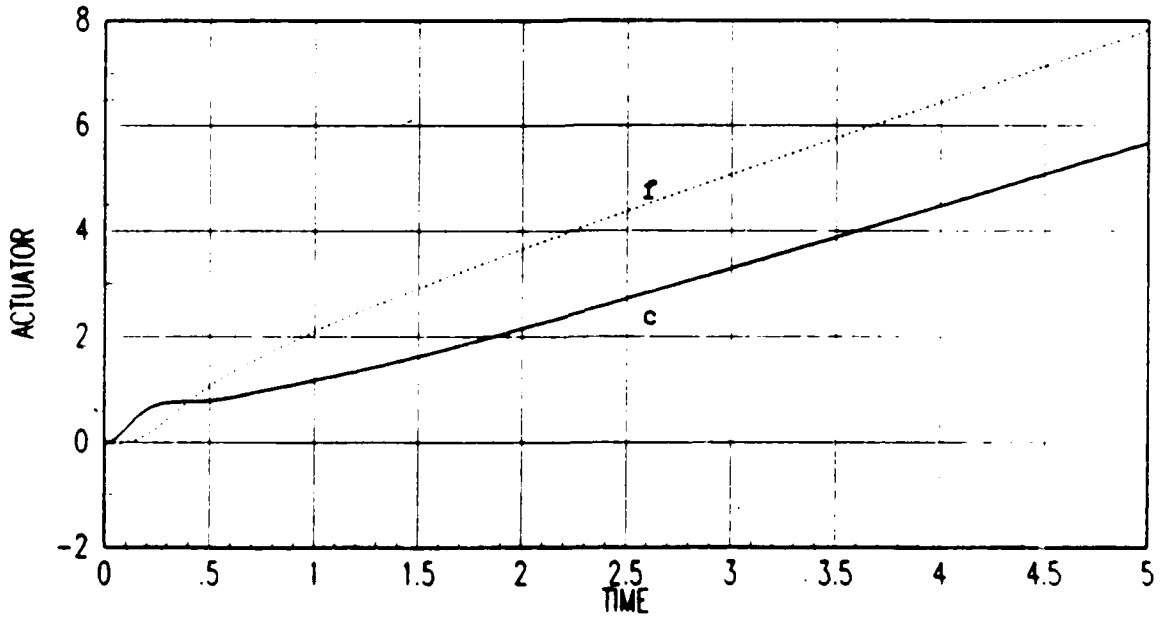



Figure 4.37. CGT/PI/KF Actuator Response, TM101

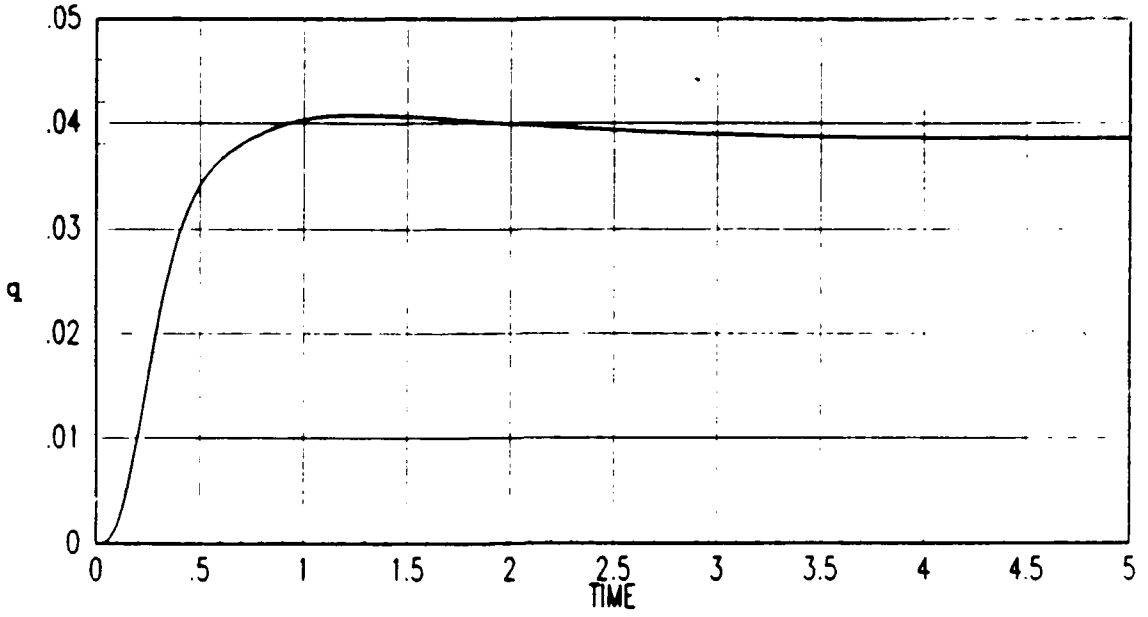



Figure 4.38. CGT/PI/KF Pitch Rate Response, TM106

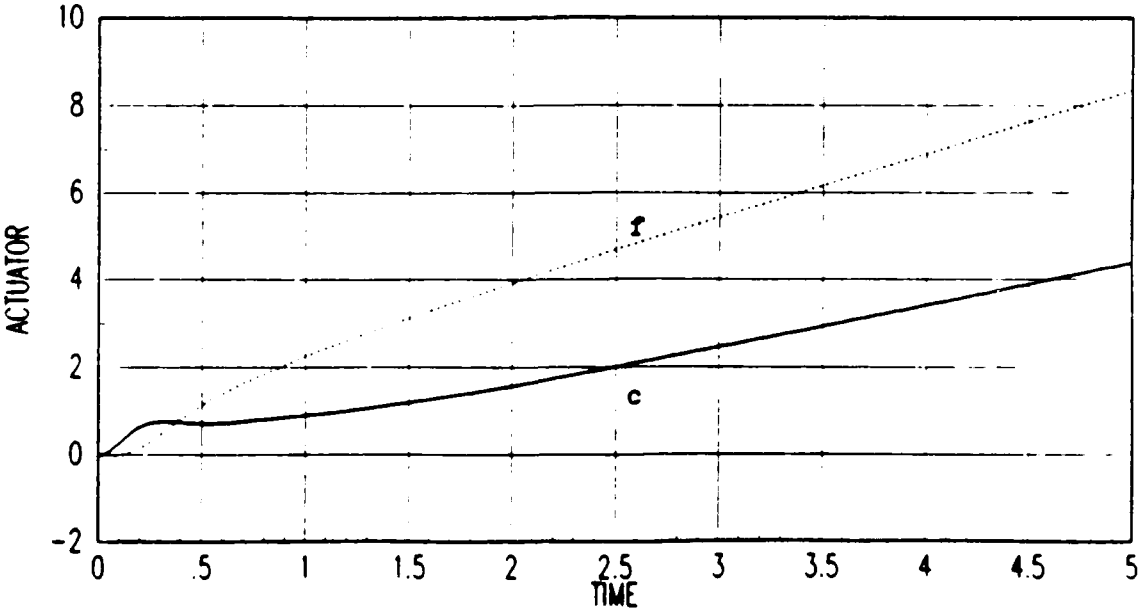



Figure 4.39. CGT/PI/KF Actuator Response, TM106

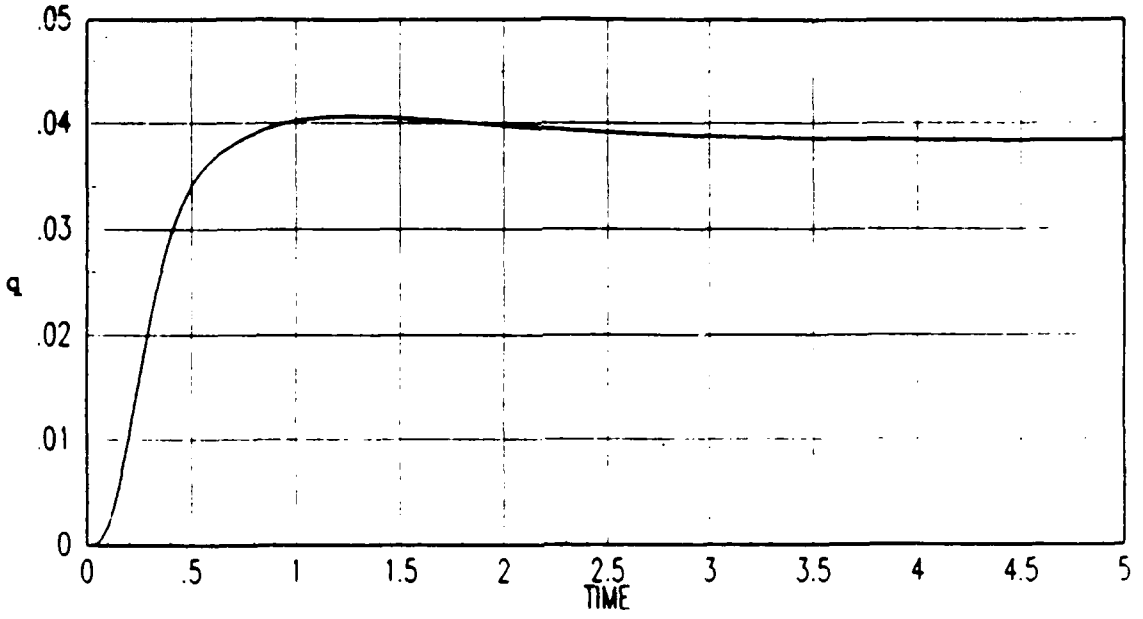



Figure 4.40. CGT/PI/KF Pitch Rate Response, TM116

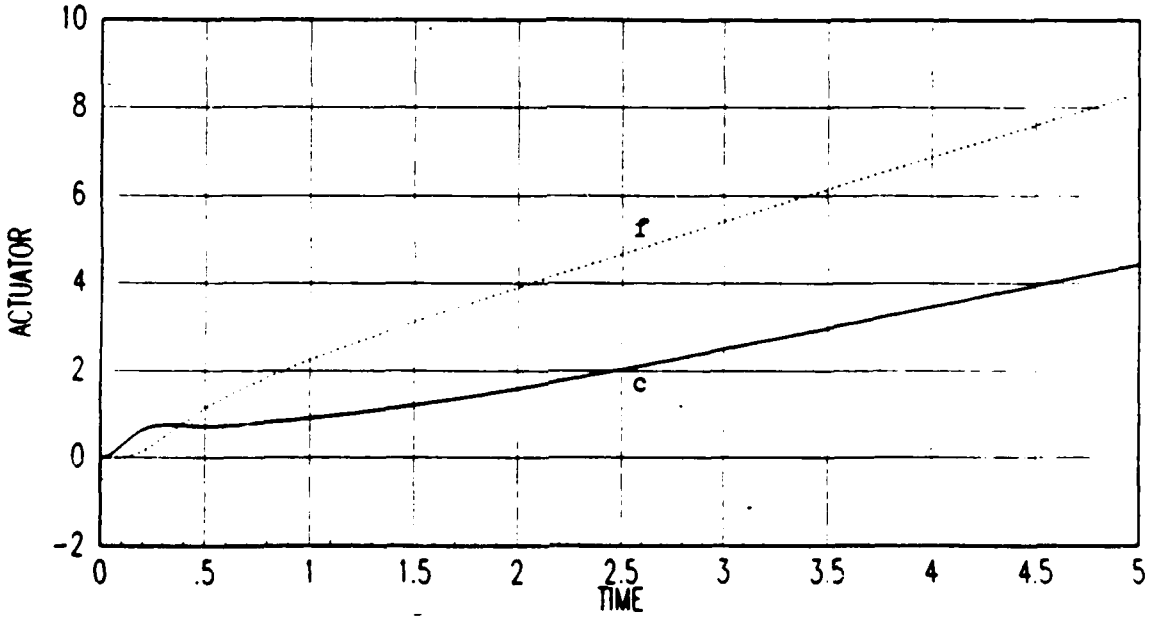



Figure 4.41. CGT/PI/KF Actuator Response, TM116

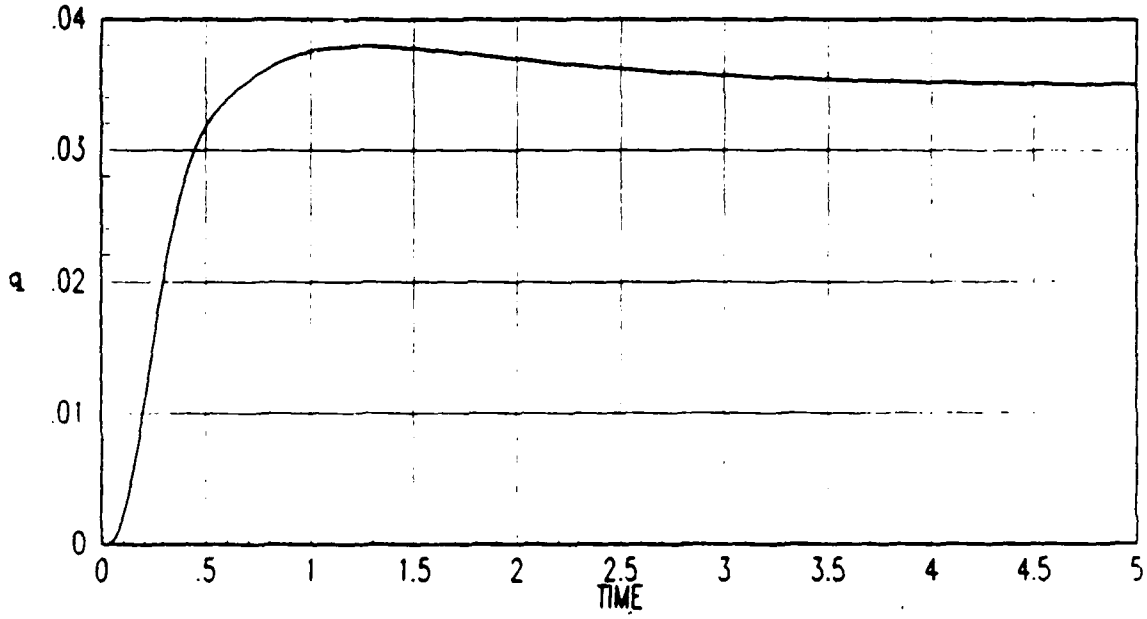



Figure 4.42. CGT/PI/KF Pitch Rate Response, TM119

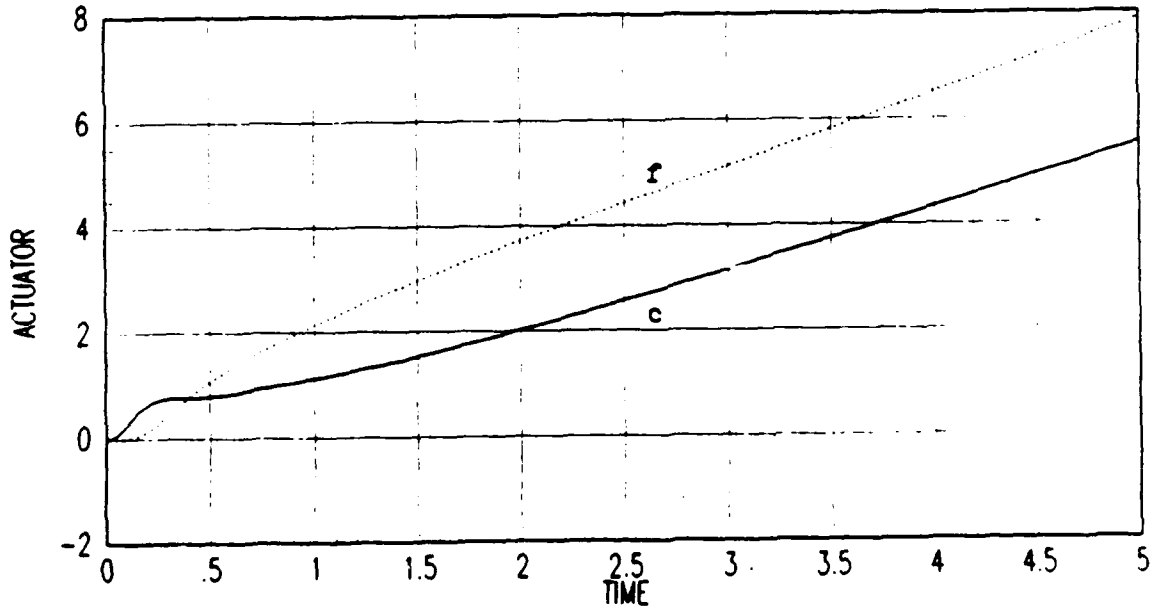



Figure 4.43. CGT/PI/KF Actuator Response, TM119

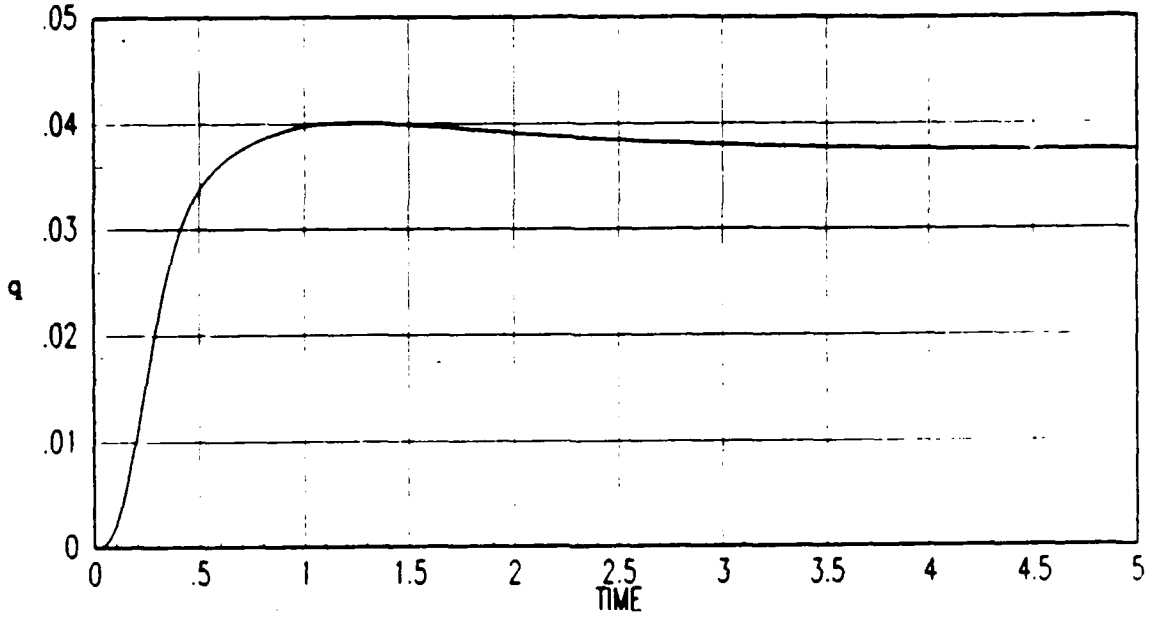



Figure 4.44. Effect of LTR Tuning, q=.1, Pitch Rate Response, TM116

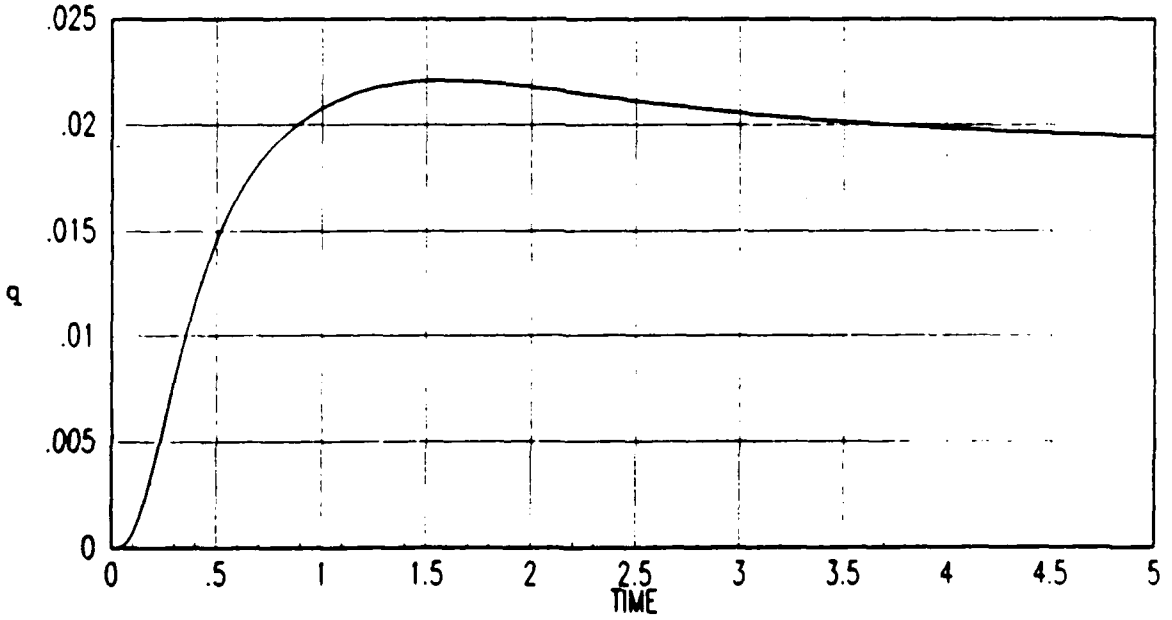



Figure 4.45. CGT/PI/KF Pitch Rate Response, TM103

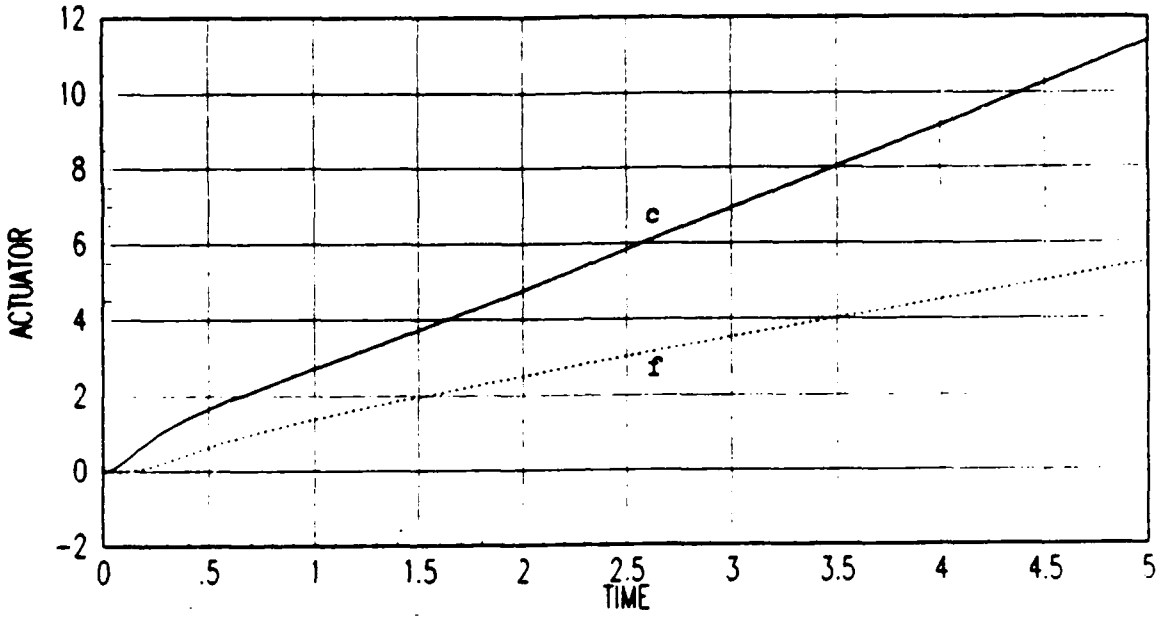



Figure 4.46. CGT/PI/KF Actuator Response, TM103

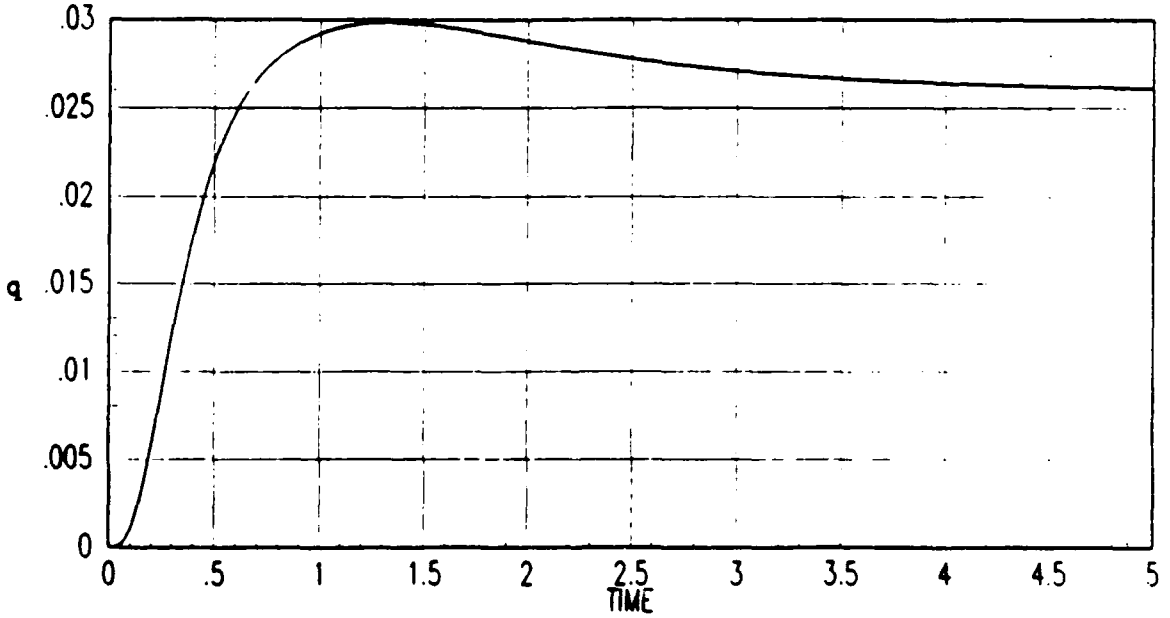



Figure 4.47. CGT/PI/KF Pitch Rate Response, TM123

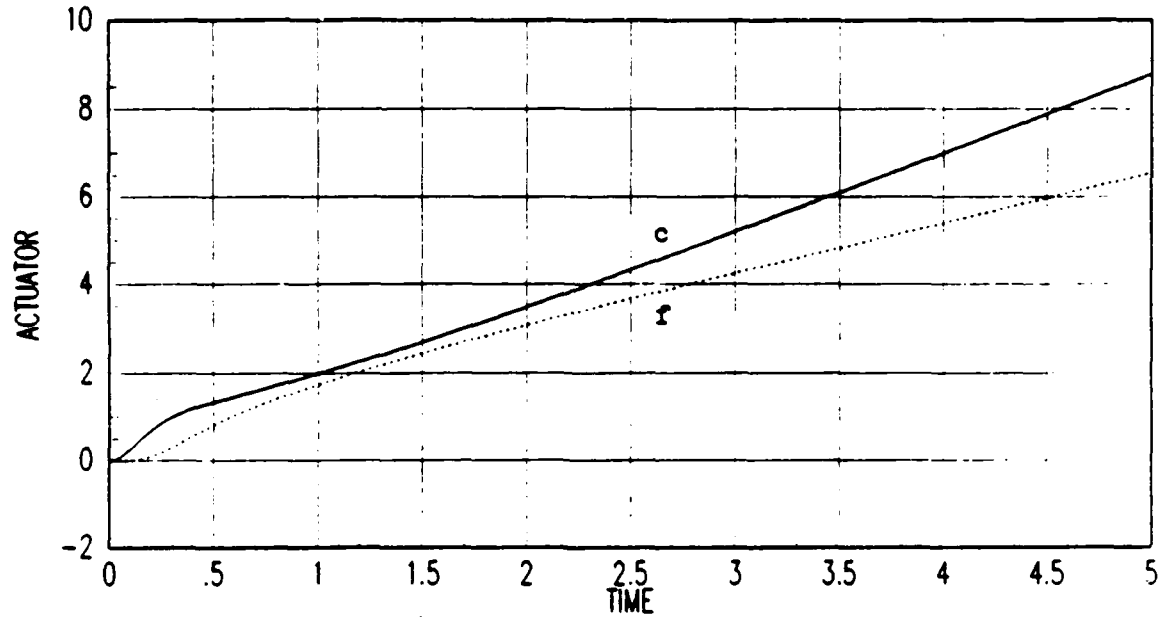



Figure 4.48. CGT/PI/KF Actuator Response, TM123

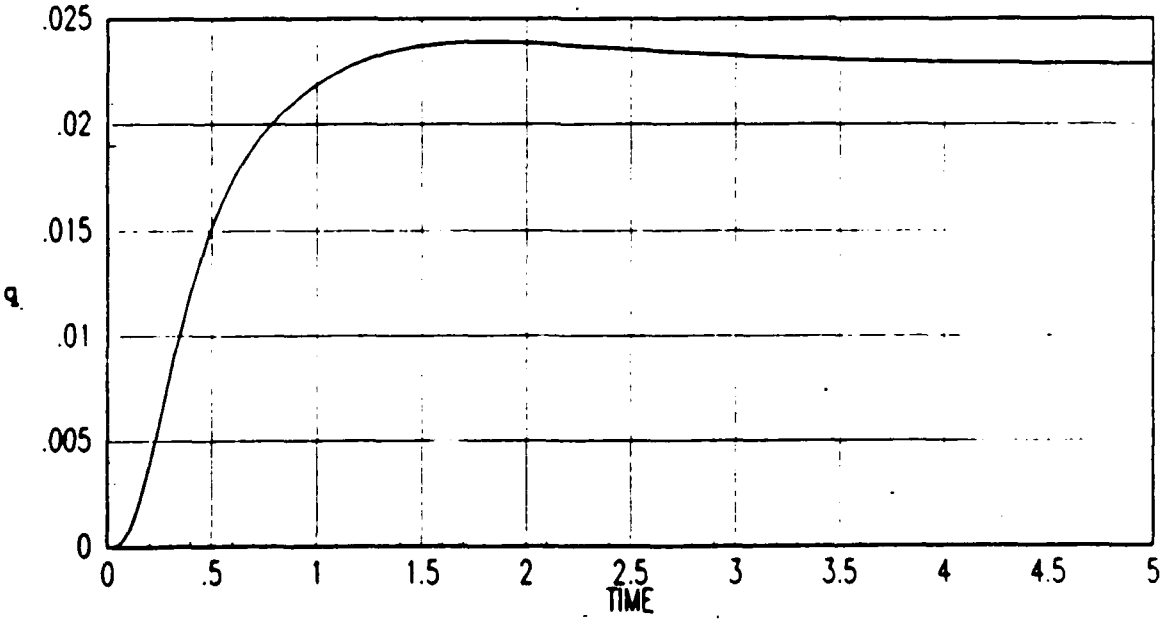



Figure 4.49. Effect of LTR Tuning, q=.1, Pitch Rate Response, TM103

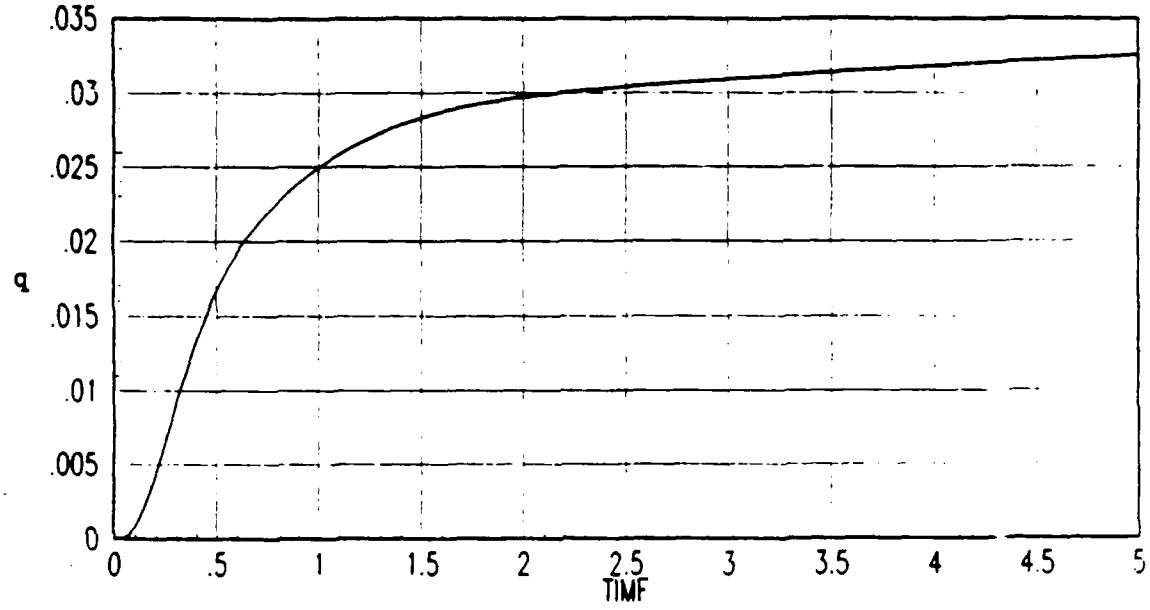



Figure 4.50. Effect of LTR Tuning, q=1.0, Pitch Rate Response, TM103

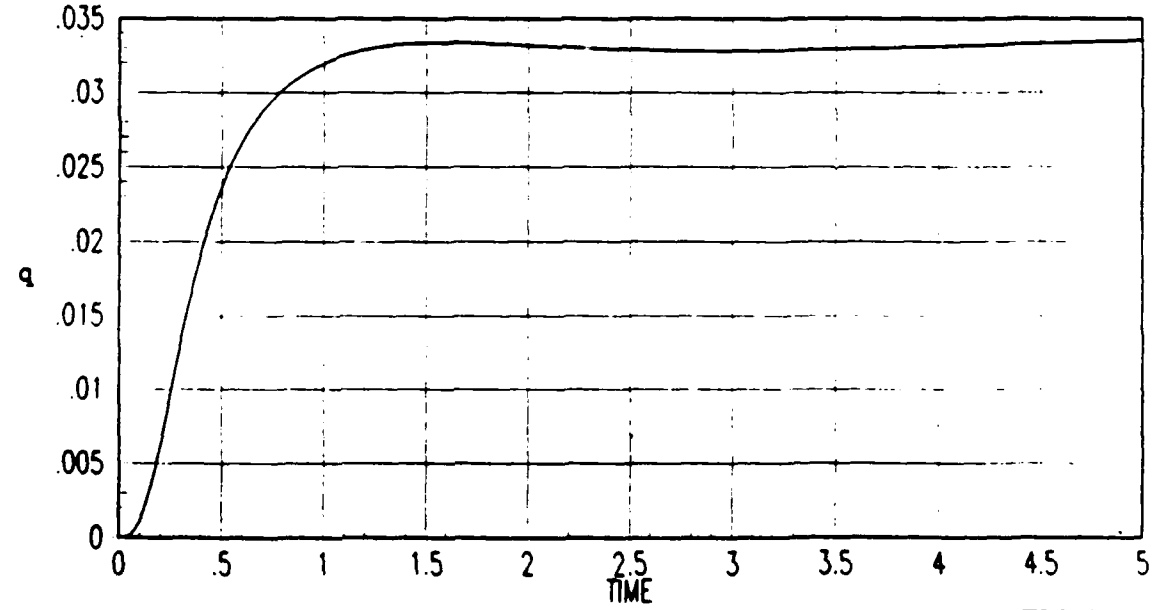



Figure 4.51. Effect of LTR Tuning, q=1.0, Pitch Rate Response, TM123

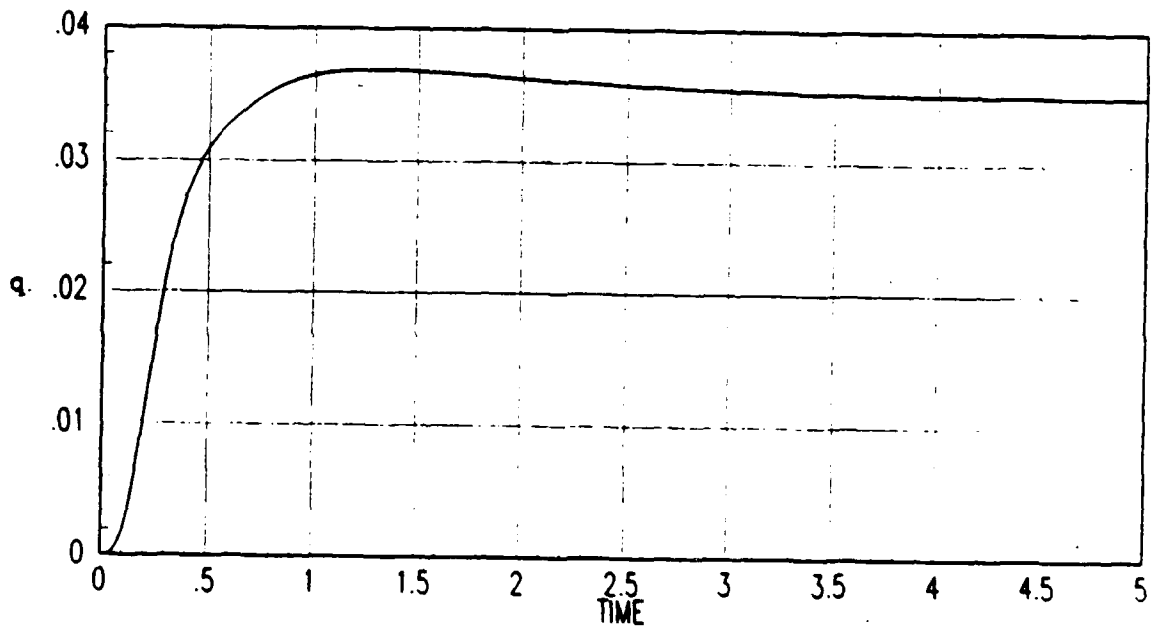



Figure 4.52. Effect of LTR Tuning, q=1.0, Pitch Rate Response, TM101

*4.6 Summary*

The CGT/PI deterministic controller was satisfactory for the nominal design condition, both with and without actuator dynamics included in the truth model. This controller proved to be robust to most failures, but canard failures and outer flap and elevator fail-to-trail were the most severe and led to instabilities. Varying $\mathbf{U_R}$ resulted in improved performance when actuator dynamics were incorporated into the truth models, successfully eliminating the oscillatory response of the actuators and of pitch rate. Decreasing the diagonal elements of the implicit model **A** matrix from -5 to -.5 resulted in further smoothing of the pitch rate response. The only problem with the controllers was the inability to contain the actuator ramping phenomenon. Increasing weights on control magnitudes and rates had negligible effect, and decreasing the weights on output deviations resulted in a degradation of performance. However, as the ramping did not cause actuator saturation for the time interval investigated (and by extrapolation not until at least 20 seconds), the controller was deemed acceptable for air combat mode applications.

Incorporation of the Kalman filter to form the CGT/PI/KF controller resulted in a slight degradation of performance for the nominal case, and a somewhat greater degradation for the three failure cases (with first order actuators simulated) which were stable when evaluated against the deterministic CGT/PI controller. The two failure conditions which were previously unstable, however, became stable with the incorporation of the Kalman filter. This was not expected. The improved performance is conjectured to be the result of the Kalman filter's band-limiting characteristic. It is possible that what was causing the instabilities in the system previously was a high frequency mismodelling characteristic which the filter fortunately cut off; i.e., the filter's effect on the controller loop shape was helpful rather than harmful for this particular phenomenon. This, however, is only a hypothesis, and further analysis would have to be performed to determine the actual cause of the improvement. Loop Transmission Recovery was shown to

improve performance significantly, without having to pay any significant penalty on performance at nominal conditions.

# V. *Conclusions And Recommendations*

## *5.1 Conclusions*

The CGTPIKF computer aided design package generated during this thesis effort is an effective and versatile tool when used in the design of CGT/PI/KF control systems. Because it is hosted on $MATRIX_X$ [10], a wide gamut of control design tools are available to the user. These tools, coupled with the time response plots generated by CGTPIKF itself, provide a thorough controller design and performance evaluation for MIMO systems. Also, the high degree of modularity and thorough documentation of the software allows a user the flexibility of modifying elements of the program, should the need arise.

The actual design of a control system for the CRCA attained both excellent performance at design conditions while remaining robust to many failures and to the effects of incorporating actuator dynamics into the truth model, which were not included in the design model. The full-state feedback CGT/PI controller performed adequately, with the nominal aircraft almost perfectly tracking the command model. Three of the failure conditions also exhibited this kind of performance, although two failure conditions did result in aircraft divergence. An anomaly was observed in that the control surfaces had a ramp, and they appeared to be fighting one another. This ramping of the control surfaces was both puzzling and unalterable by the design stategies pursued.

With the Kalman filter in the loop, performance at nominal flight conditions degraded slightly, but still conformed to the specification [1]. The three failure conditions that were previously stable also showed some degradation of performance, but the use of Loop Transmission Recovery (LTR) improved this performance. Surprisingly, the two failure cases which were unstable became stable when the Kalman filter was included in the loop. This was not expected, and it is hypothesized that

the band limiting characteristic of the Kalman filter may have affected the system loop shape so as to attenuate high frequency components which were the cause of the instability, but this was not verified. LTR tuning improved the performance at these failure conditions, with negligible loss of performance at nominal conditions.

In the course of designing the flight control system, several insights were gained as to how to adjust the controller to improve performance. Implicit model-following proved to be superior to the explicit, or standard, quadratic cost definition for synthesizing the PI controller, particularly with regard to the transient response of the closed-loop system in the face of failure conditions. Increasing the weight $\mathbf{U_R}$ on control rates proved to be the most effective way to enhance stability robustness. The effect of both $\mathbf{U_M}$ and $\mathbf{R_I}$ (the standard and implicit weighting matrices for control magnitudes) proved to be negligible. The validity of adding the implicit and standard weighting matrices together was demonstrated.

### *5.2 Recommendations For Future Research*

The possibility of incorporating a transformation matrix, which could either directly alter control input magnitudes or their signs, into a controller designed via LQG theory, should be investigated. Presently, no work has been done to prove the applicability of this type of weighting matrix to LQG, but work has been done in other fields [15], and the possibility exists that this could be extended to LQG to enhance performance of controllers.

The CGTPIKF software is, at the present time, totally user interactive. It would be beneficial if this software were able to be submitted as a batch job, with a range of weighting matrix values or LTR q factors being evaluated with the user off-line. This would save a great deal of time, as the present software is rather slow, requiring the user to wait while the program computes gains or builds the simulations. One approach to accomplishing this task would be to alter the appropriate CGTPIKF command files to remove all of the prompts and optional

design paths. This modified software could then be executed via another command file in which the specific variables of interest are varied.

The incorporation of time correlated noises into the software would also be beneficial. This could be readily accomplished by altering the CGT user-defined function. Also, an alternative approach of solving for the CGT gains which does not require the use of the HESSENBERG function would be beneficial, as this MATRIX$_X$ function does not always provide the correct result, depending on the version of MATRIX$_X$ used.

# Appendix A. *CGTPIKF User's Guide*

## *A.1 Introduction*

This Appendix is a user's guide to the CGTPIKF software. Experience using MATRIX$_X$ [10] is a prerequisite for using this software. Not only will the user know fundamentals such as how to enter matrices, but the whole gamut of MATRIX$_X$ tools will be available for the design process, including those aspects of the system which the author has not included in the program. The primary benefit of implementing a computer aided design (CAD) package which assists in the design and evaluation of CGT/PI/KF control systems on MATRIX$_X$ is that many different analysis tools exist, such as singular values, eigenvector evaluation, frequency analysis, etc. This software is intentionally designed to allow the user to apply such tools at various points throughout the iterative design process. Despite the limitations of the program, when considered in concert with the total MATRIX$_X$ CAD package, the program capabilities are enormous. Therefore, some knowledge of the capabilities of MATRIX$_X$ is essential to the control system designer.

This software has been qualified against results obtained from a previous CAD package which was written in FORTRAN IV [5]. The specific test case presented here is a duplication of Captain Gross's results (his Figure 5-1 and 5-2)[7]. Figure A.1 and A.2 were generated by the CGTPIKF software after inputting Gross's models and weighting matrices. These figures are extremely close to his results, and served as a verification of software functionality.

The files required to run CGTPIKF (or a subset of the program) are listed in Section A.5. Section A.6 provides a brief description of how the simulation is implemented. Actual use of the program is documented in Section A.2. Variable definitions are given in Section A.3, while some nuances of the software are listed

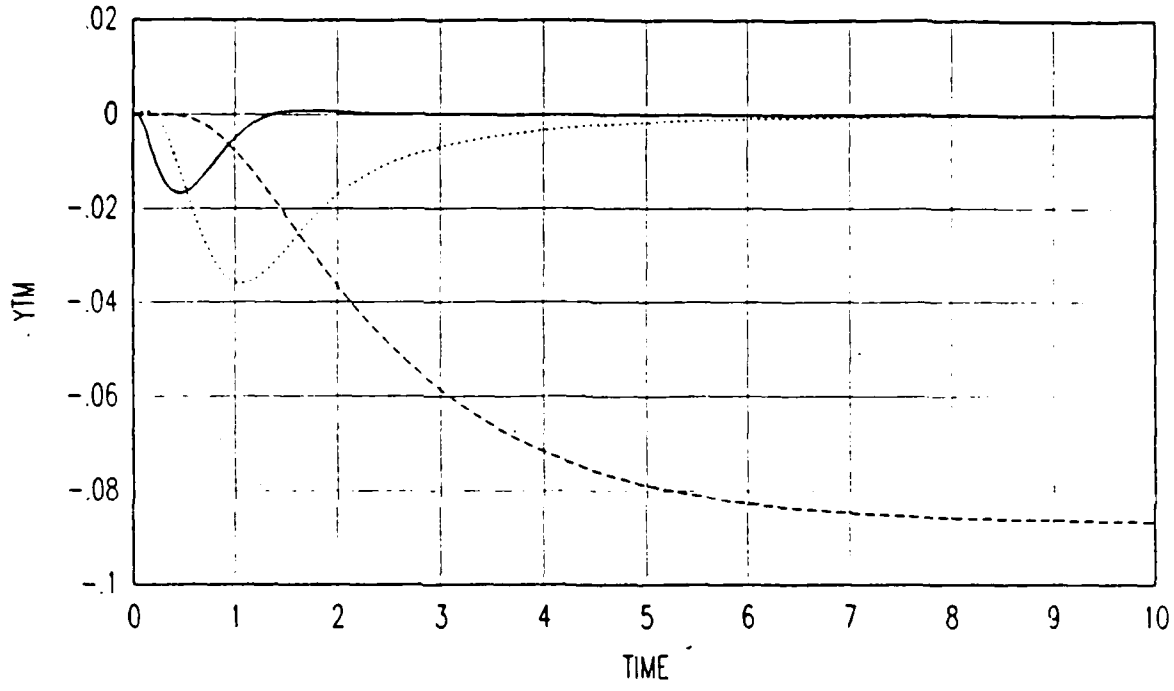



Figure A.1. Duplication of Gross's Output Response

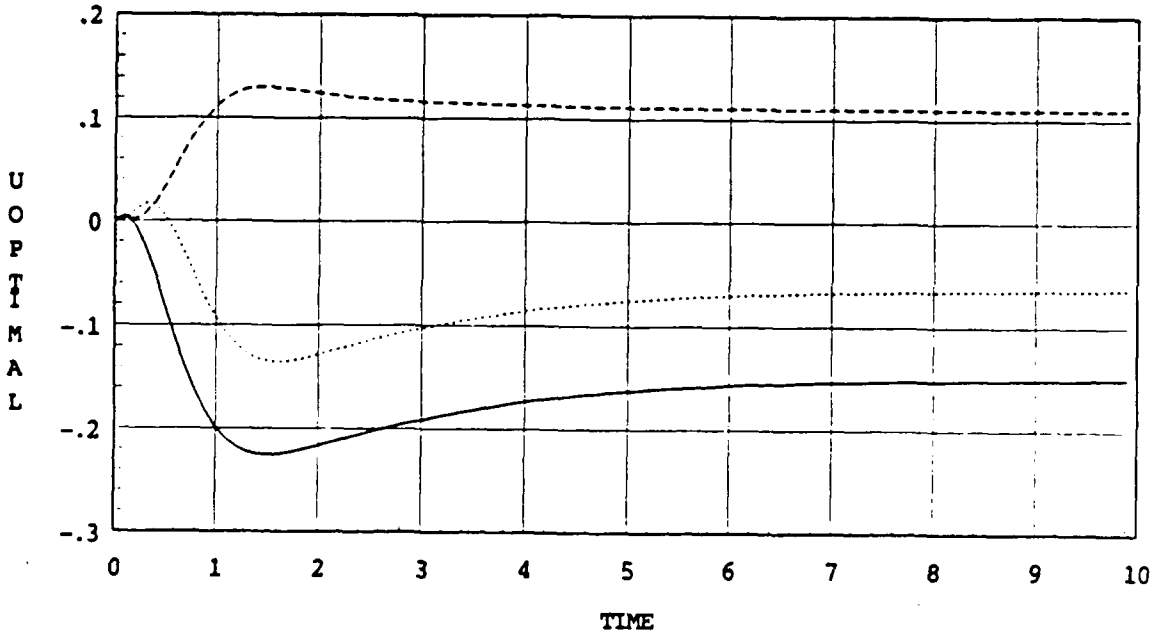



Figure A.2. Duplication of Gross's Actuator Response

in Section A.4. A sample design run is not provided, as it is of far too great a length to be included in a user's manual.

It should be noted that CGTPIKF was written on a VAX 11/780 using $MATRIX_X$ Version 6.0. Errors may occur when using other $MATRIX_X$ versions, as the format of commands is not always the same. The author encountered two errors while trying to implement this software on a Micro Vax 3 (VAX 3500) using $MATRIX_X$ VMS Version 1.2. The first error involved the format of the $MATRIX_X$ command DREGULATOR. In the present version of CGTPIKF, the argument SXU has to be transposed to be of the correct dimensions for this command, but on the VMS version it did not require transposition. Secondly, on the VMS version, the HESSENBERG function, used in the generation of CGT gains, does not produce the correct results. If this occurs, the user will see an error message generated by CGTPIKF which states THE BARRAUD FUNCTION DID NOT WORK CORRECTLY.

## *A.2 Using CGTPIKF*

*A.2.1 Overview* CGTPIKF is a user friendly CAD package which aids in the design of command generator trackers, proportional plus integral controllers, and Kalman filters. The program is based on the LQG assumptions, that is, that there is a Linear system model, that a Quadratic cost is to be optimized, and that the system is driven by Gaussian noises. Certainty equivalence states that the design of an optimal stochastic controller can be divided into the design of the deterministic full-state feedback control system and that of the Kalman filter. In CGTPIKF, the designer can pursue each of these designs separately, analyze each, and then combine the deterministic CGT/PI controller in cascade with the Kalman filter to form and analyze the performance potential of the optimal stochastic controller.

The primary analysis tool of CGTPIKF is time response plots. A common

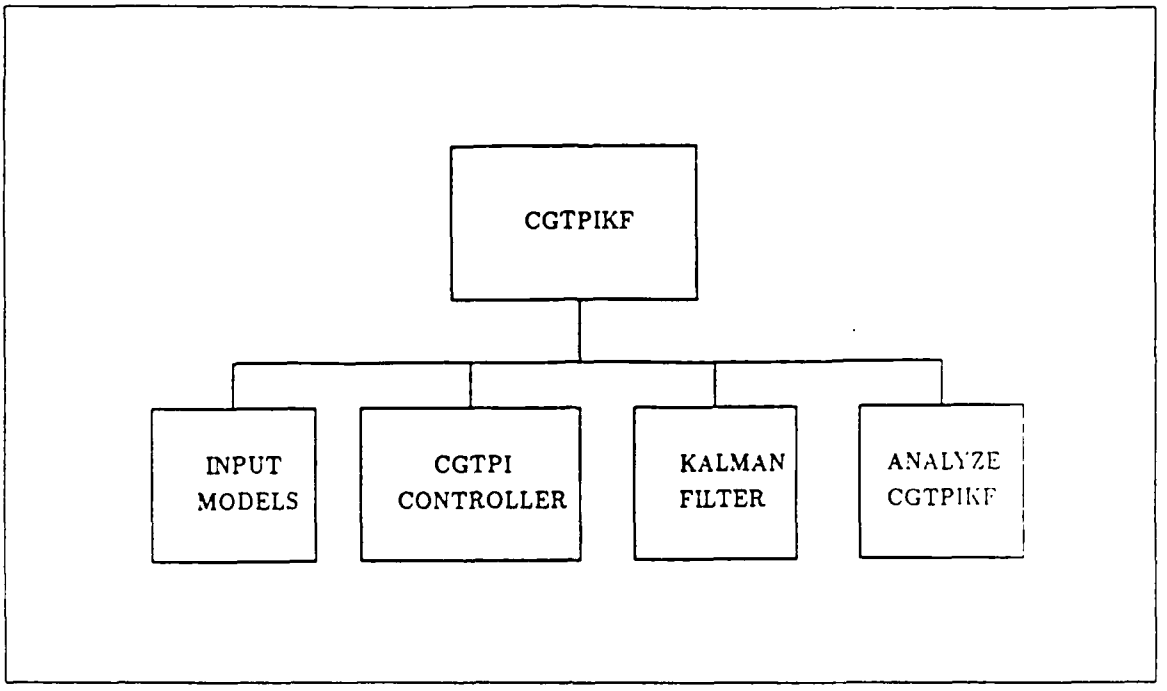



Figure A.3. CGTPIKF Top Level Program Flow

problem occurs while using modern control theory techniques in the design of control systems, that being how to relate classical performance specifications to the design. By using time response plots, the designer can evaluate the proposed design against actual time domain specifications. Frequency response and singular value analysis are not included in the program, but they are a part of MATRIX$_X$ , so the user may exit CGTPIKF and use these tools if desired.

Limitations which apply to CGTPIKF are that the models must be linear and time invariant. Also, the number of inputs and the number of outputs for the design, command, and truth models must be the same.

*A.2.2 Program Overview* CGTPIKF is menu driven. As such, the designer has the option of pursuing several design paths (Figure A.3). At the top level, the user may input or change system models, pursue a deterministic CGT/PI controller, or pursue a Kalman filter. The models required to pursue either of these last two options are listed in the appropriate section (A.2.5 or A.2.6). Finally, the

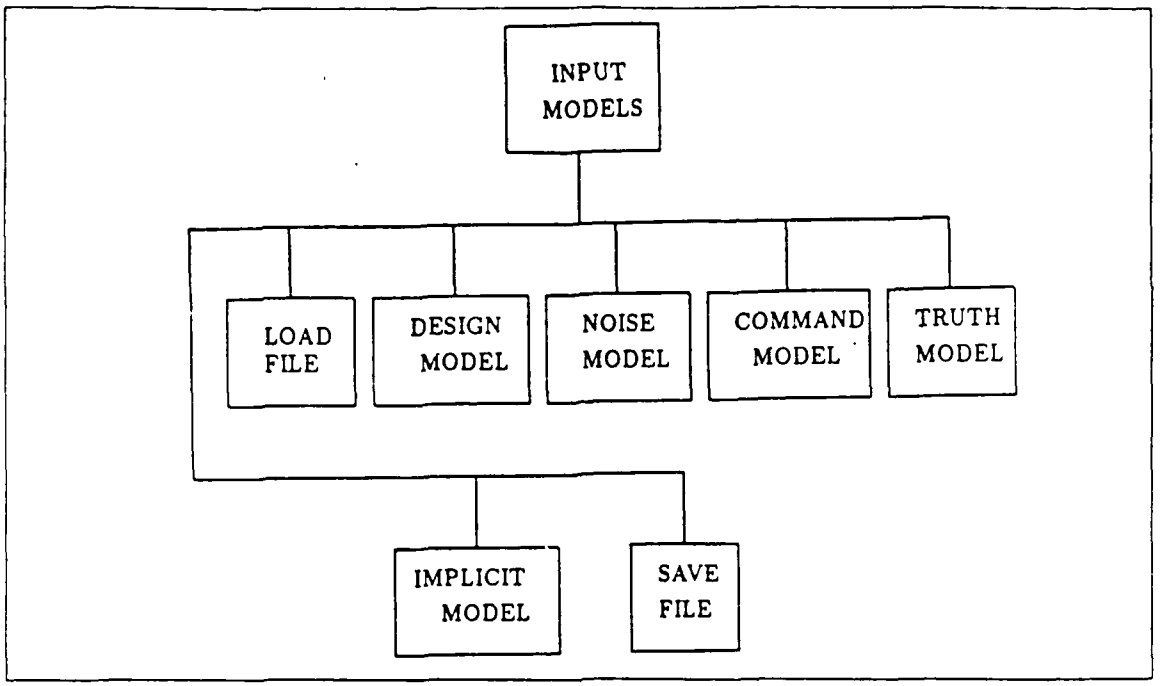



Figure A.4. INPUT MODELS Program Flow

user may exit the program. Before exiting the program, the user will be prompted for whether or not the variables on the stack (the variables presently defined in $MATRIX_X$ ) are to be saved. For all menus except the PI menu, the user has the option of exiting that menu (and its set of options) and returning to the previous menu. The limitation of not exiting from the PI menu is to ensure that new PI gains are generated if the cost weighting matrices, implicit model, or choice of explicit vs implicit model-following controller is changed.

When the designer selects INPUT MODELS, a menu appears allowing the option of entering the design model (upon which both the deterministic CGT or CGT/PI controller and the Kalman filter are based), the command and noise models (for CGT generation), the truth model to compare the controller or filter against, the implicit model for PI implicit model following, and the sample time (Figure A.4).

When the designer decides to pursue the CGT/PI controller (Figure A.5),

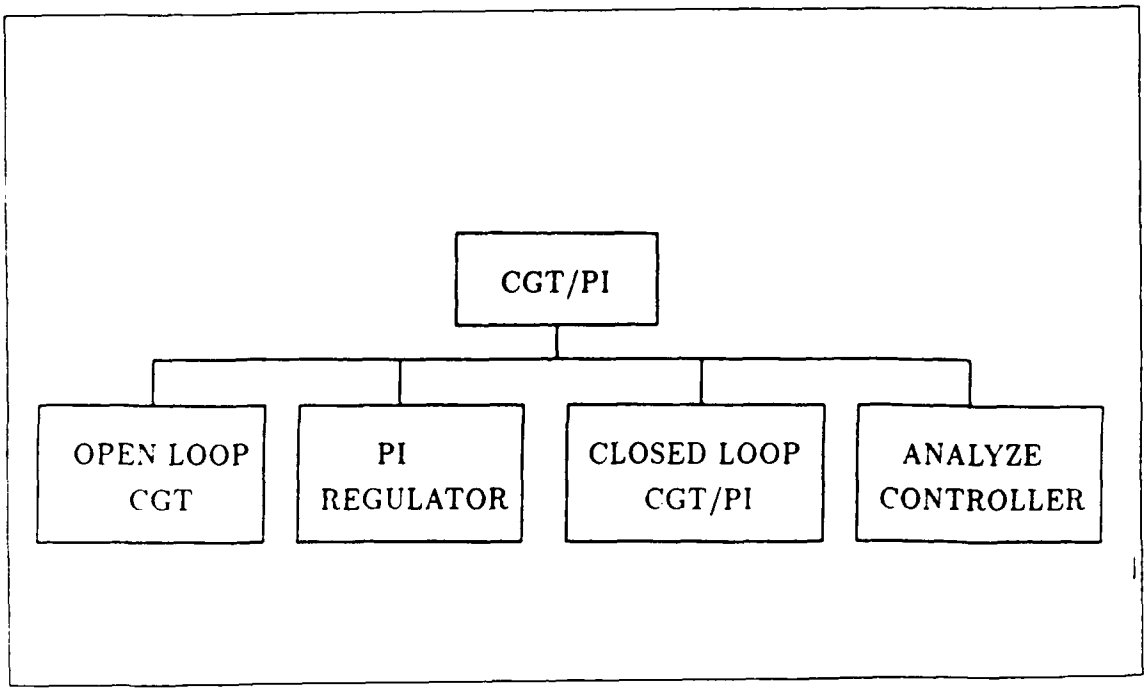



Figure A.5. Pursuing A CGT/PI Controller

he is given the option of pursuing an open-loop CGT (if the system is stable), a closed-loop CGT/PI, or a PI controller. Also, the ANALYZE CGTPI option allows the user to evaluate the controller design and get time response plots. When a PI or CGT/PI controller is being pursued, the user may enter cost weighting matrices and pursue either explicit, implicit, or a combination of implicit and explicit model following. Explicit model following is a standard PI controller which tries to minimize perturbation deviations from zero. A PI controller based on implicit model following tries to make the closed loop system dynamics characteristics emulate an implicit model, which is defined by the designer. Pursuing both simply adds the weighting matrices of an explicit controller to those of an implicit controller.

If the user pursues the KALMAN FILTER option (Figure A.6), he is given the opportunity of modifying the G matrix, which determines how noises are fed into the system, or modifying the noise covariance matrices **Q**, **R**, or **QR**, which is the cross correlation between the discrete $\mathbf{w_d}(t_i)$ and $\mathbf{v}(t_i)$. He can also generate

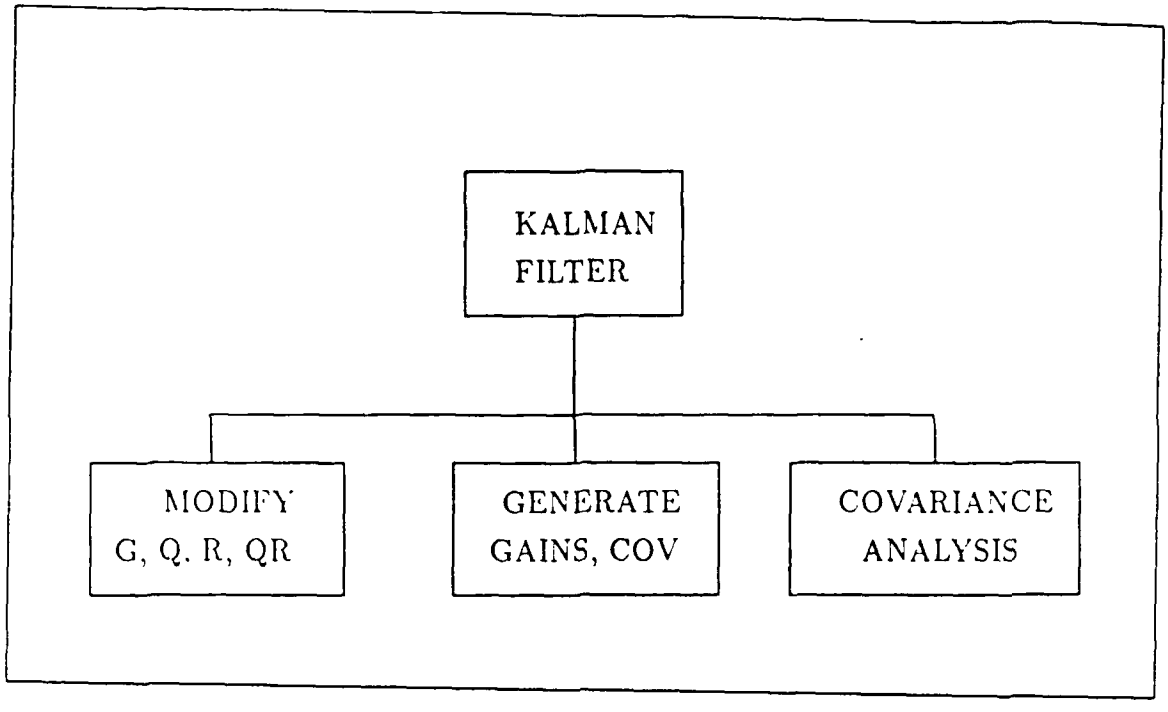



Figure A.6. Kalman Filter Program Flow

the Kalman filter gains and covariances. Finally, a covariance analysis of the filter (alone) against the truth model can be performed to allow for proper filter design and tuning. Note that when Loop Transmission Recovery (LTR) is being pursued, **G** must be the identity matrix.

Finally, analyzing the closed-loop CGT/PI/KF controller (Figure A.7) provides the final verification of the design process. This option also allows LTR tuning to be pursued.

*A.2.3 Getting Started* To run CGTPIKF, the files listed in Section A.5 must exist in the user's directory. Once that is accomplished, enter $\text{MATRIX}_X$ . At the prompt, (<>) type:

```
EXECUTE('CGTPIKF.')
```

and hit return. $\text{MATRIX}_X$ will now execute the CGTPIKF command file. The program will respond with

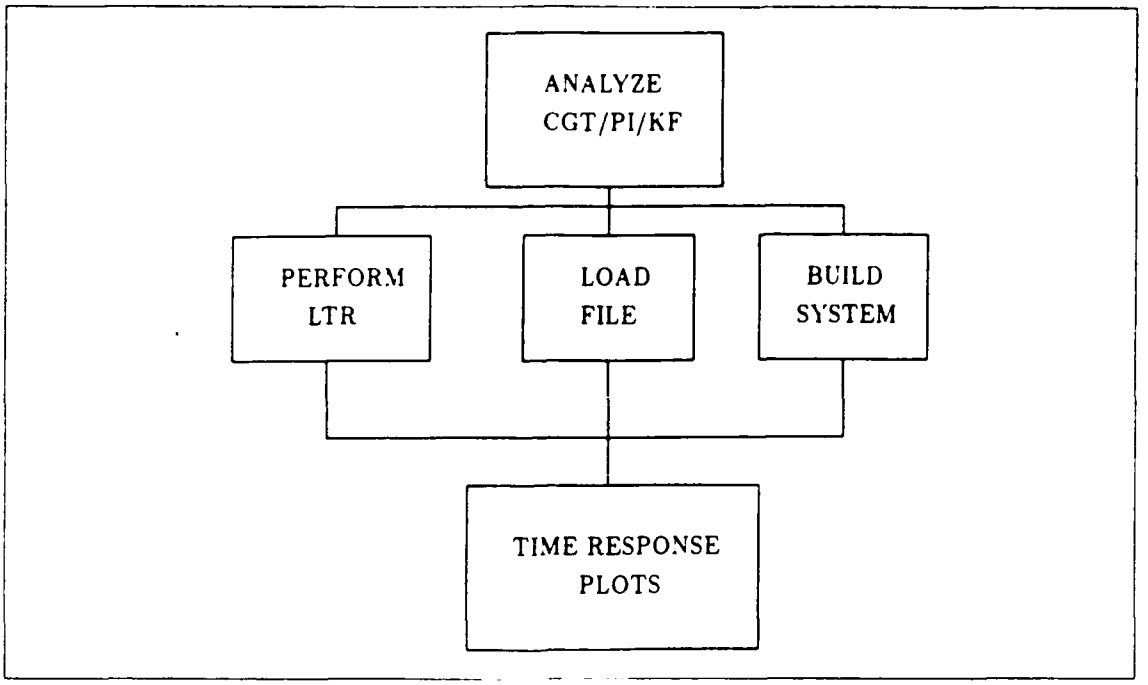



Figure A.7. Analysis Of Closed-Loop CGT/PI/KF Program Flow

```
   CGTPIKF TOP MENU
1    INPUT MODELS
2  CGT/PI CONTROLLER
3    KALMAN FILTER
4  ANALYZE CGT/PI/KF
5        QUIT
```

These are the five options described in Section A.2.2. The next five sections will deal with these options.

*A.2.4 INPUT MODELS* When INPUT MODELS is selected from the main CGTPIKF menu, the user is presented with a banner telling him what to do. This banner is

```
THE DESIGN, COMMAND, AND NOISE MODELS, AND SAMPLE TIME DT,

MUST BE INPUT, OR THE PROGRAM WILL ABORT.  THE DESIGN MODEL MUST BE

INPUT BEFORE THE COMMAND, NOISE, TRUTH, OR IMPLICIT MODELS.

THE TRUTH MODEL DOES NOT HAVE TO BE INPUT UNTIL YOU

EVALUATE YOUR CONTROLLER, AND THE IMPLICIT MODEL MAY BE INPUT

AT A LATER TIME. THIS PROGRAM DOES NOT PRESENTLY USE THE NOISE

MODEL, BUT IT STILL MUST BE INPUT.

PAUSE>


FOR ALL MODELS, IF A MATRIX ALREADY EXISTS AND YOU DO NOT WANT

TO CHANGE IT, JUST TYPE IN THE APPROPRIATE MATRIX NAME (EG, ADM).



PAUSE>
```

When a $PAUSE >$ appears anywhere in this program, hitting the RETURN key will allow the program to proceed. ADM refers to the design model's **A** matrix. This notation will be explained later in this section. A complete list of variable

names and their definitions is provided in Section A.3.

The program then displays the various options available to him.

```
     MODEL MENU
1     LOAD FILE
2    DESIGN MODEL
3    NOISE MODEL
4   COMMAND MODEL
5    TRUTH MODEL
6   IMPLICIT MODEL
7    SAMPLE TIME
8    SAVE MODELS
9       QUIT
```

Each of these options allows the user to input the desired model. Options 2 through 7 result in the program presenting a banner to the user which tells him what he is expected to do. For each of the models, the user will input the continuous-time matrices which define the model. (Banners which are self explanatory will no longer be presented.) An option to look at variable values, which is included in other menus, is not included here because all of the matrices defining a model are displayed to the user after the model has been entered.

If LOAD FILE or SAVE MODELS is chosen, the user will be prompted for the file name. To load or save the stack to a file, simply type the filename inside of single quotation marks, that is, 'filename.dat'. The dat extension is not required, but if no extension is chosen, that is the default imposed by $MATRIX_X$ . Care must be taken when loading a file, as inputting a filename which does not exist will result in CGTPIKF aborting. (This is a $MATRIX_X$ limitation, not a limitation

of the CGTPIKF software.) All of the variables which have been defined, however, will remain on the stack.

When inputting matrices to define the models, the equations are of the general form for states **x**, controlled outputs **y**, and measurements **z**:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{w} \tag{A.1}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{A.2}$$

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v} \tag{A.3}$$

Not all of these matrices will be displayed for each model (e.g., the command model only requires **A**, **B**, **C** and **D**). The program will display a banner at the beginning of each model input cycle informing him of what the form of the model is. He will then be prompted for each of the necessary matrices. THE DESIGN MODEL MUST BE INPUT (or exist on the stack) BEFORE ANY OTHER MODEL IS INPUT OR THE PROGRAM WILL ABORT! The truth model must exist before any analysis is performed, as in analyzing the CGT/PI or CGT/PI/KF controllers, or performing a Kalman filter covariance analysis. The **G** matrix of the design model must be the identity matrix if LTR tuning will be performed, but this limitation is not imposed when it is entered.

The **A** matrix dimensions for each model defines the number of states of that model. Variable names are a mnemonic consisting of the matrix name and the model (DM = design model, NM = noise model, CM = command model, TM = truth model, or IM = implicit model) with which they are associated. For example, the design model **A** matrix is ADM. Later, these matrices will be discretized, and the resulting matrices will have a 'D' added at the end (e.g., ADMD). A complete list of variable names which exist on the stack after a typical run is given in Section A.3.

When including actuator dynamics in the truth model, a specific format is required. Suppose the designer wanted to add actuator dynamics states to a truth

model described by

$$\dot{\mathbf{x}} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} \tag{A.4}$$

and inputs $\delta_1$ and $\delta_2$. The truth model as entered by the user would have to be

$$\dot{\mathbf{x}} = \begin{bmatrix} a_1 & a_2 & b_1 & b_2 \\ a_3 & a_4 & b_3 & b_4 \\ 0 & 0 & -20 & 0 \\ 0 & 0 & 0 & -20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \delta_1 \\ \delta_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 20 & 0 \\ 0 & 20 \end{bmatrix} \begin{bmatrix} \delta_{1-command} \\ \delta_{2-command} \end{bmatrix} \tag{A.5}$$

where the numbers entered are assumed to represent the desired actuator dynamics. In general, the actuator states must be entered in order from lowest derivative to highest derivative, alternating between the actuators of the system. For the above example, if second order actuator dynamics were desired, the state vector would be

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \delta_1 \\ \delta_2 \\ \dot{\delta}_1 \\ \dot{\delta}_2 \end{bmatrix} \tag{A.6}$$

The user will be prompted for the number of states of the actuator (NSACT), and for the above example would enter 4. The order of each actuator model must be the same for all actuators. If a greater order of actuator states is desired for one actuator vs another, the actuator represented by the lower order of states must be augmented by the appropriate number of "dummy" states which will yield an augmented vector of dimension equal to the highest order term. These dummy states will have a 0 entry in the A matrix and a 1 entry in the **B** matrix. NSACT will therefore be a multiple of the number of inputs.

When the number of truth model states (NSTM) is greater than the number of design model states (NSDM), the program will prompt the user for a transformation matrix (TTM) which transforms the truth model states into a matrix with the same number of rows and columns as the design model. This is required to ensure dimensional compatibility during the simulation.

*A.2.5 CGT/PI CONTROLLER* When CGT/PI CONTROLLER is selected from the CGTPIKF TOP MENU, the following menu is displayed:

```
        CGTPI MENU
 1     OPEN LOOP CGT
 2     PI REGULATOR
 3  CLOSED LOOP CGT/PI
 4    ANALYZE CGT/PI
 5    SAVE VARIABLES
 6  LOOK AT VARIABLES
 7         QUIT
```

If a pseudoinverse is used to invert the $\mathbf{\Pi}$ matrix of Equation (2.17), the banner

```
A PSEUDOINVERSE WAS USED TO GENERATE THE PIMAT MATRIX
```

appears, to alert the user that the design will not be based on a 'real' inverse. Two functions are defined at the beginning of this part of the program, and they will scroll by on the screen. This occurs at other points in the program also, and is not important to the user.

The design, noise, and command models must exist to generate CGT gains. If only a PI regulator is being pursued, only the design model is required. The implicit model must exist whenever implicit model-following is being pursued.

If Option 1 is selected, the program will pursue an open-loop CGT controller if the design model **A** matrix has eigenvalues with strictly negative real parts. Selecting this option generates in the gains A11, A21, A22, A13, and A23 of Equation (2.9). (A12 is not generated because it is not used in either the open-loop CGT or closed-loop CGT controller.) If any of the eigenvalues are in the right half s-plane, the user will be told that the system is unstable and to pursue a closed-loop CGT. If the error message "THE BARRAUD FUNCTION DID NOT WORK CORRECTLY" appears, then the values of A11 through A23 are incorrect. This may be caused by the $MATRIX_X$ HESSENBERG function producing inaccurate results. Refer to the $MATRIX_X$ User's Manual [10] for details.

Option 2 allows the user to pursue a PI controller (regulator). The execution of this option is the same as pursuing a closed loop CGT/PI controller, because the user inputs the same data and makes the same choices. The only difference is that Option 3 calls the CGT gain generation function, and forms the gain $KXM = KX * A11 + A21$ as well as KX, KZ, and A11 through A23. Option 2 only forms KX and KZ. The analysis tools of Option 4 have NOT been qualified against designs pursued under Options 1 or 2. Option 4 has only been used in conjunction with closed-loop CGT/PI control systems, as generated by Option 3. The gains generated have been verified against other software packages [5], but the simulation using SYSTEM BUILD commands has not been verified for the designs of Options 1 or 2.

Option 3 results in the menu

```
        PI MENU
1  INPUT COST MATRICES
2  INPUT IMPLICIT MODEL
3    PURSUE EXPLICIT
4    PURSUE IMPLICIT
```

```
5       PURSUE BOTH
6    ALTER XIE ELEMENT
7    LOOK AT VARIABLES
8  CONTINUE WITH DESIGN
```

Option 1 allows the user to input the continuous cost weighting matrices corresponding to Equations (2.28) and (2.80), that is, $\mathbf{Y}, \mathbf{U_M}, \mathbf{Q_I}, \mathbf{R_I}$ and $\mathbf{U_R}$. Positive definiteness and positive semidefiniteness are checked. Option 2 allows the user to define (or redefine) an implicit model if implicit model following will be pursued. Option 3, PURSUE EXPLICIT, allows the user to design a standard PI controller. This option implements Equations (2.51)-(2.53) to generate the upper partition of the cost weighting matrix of Equation (2.50), consisting of the $\mathbf{X_{ii}}$ terms. This matrix is called XIE, which symbolizes a weighting matrix for implicit or explicit model following. If Option 4 is pursued, XIE becomes the upper left partition of the weighting matrix of Equation (2.81), that is, implicit model following is pursued. Option 5 results in XIE being the sum of the previous two results. Options 3 and 4 do not have to be pursued before selecting Option 5. After each of these three options have been pursued, XIE is checked for positive semidefiniteness.

After the cost weighting matrix XIE has been formulated via Options 3, 4, or 5, individual elements may be changed by the user in Option 6. This may be necessary to ensure XIE is positive semidefinite (which is checked once again after the elements have been changed). Option 7, LOOK AT VARIABLES, allows the user to see the value of any variable on the stack. This option may also be used as a scratch pad, as for determining the eigenvalues of a matrix (EIG(A)).

As previously mentioned, the PI MENU is the only menu from which the user cannot exit. The only way the program allows the user to exit this part of the program is to pursue Option 8, CONTINUE WITH DESIGN. After entering this option, the XIE matrix will be displayed and the user will be prompted for any

changes. Any changes will have to entail retyping the entire matrix at this point. If no changes are desired, typing the matrix name XIE will allow the program to proceed. If XIE is positive semidefinite, the user will then be prompted for off diagonal terms, SXU, of Equation (2.50) or (2.81). The program then goes about generating the gains KX and KZ of Equation (2.61). The weighting matrices XIE, UR, and SXU are discretized (and remain on the stack as XIED, URD and SXUD), and the CGT option is pursued internal to the program. The system eigenvalues and the gains KX, KZ, and KXM (which equals $\mathbf{K_x A_{11}} + \mathbf{A_{21}}$) are displayed to the user. The program then returns to the CGTPI MENU.

If Option 4 of the CGTPI MENU is pursued, the user enters the portion of the program which allows for the evaluation of the deterministic controller. The truth model must exist before trying to generate any time response plots. As a first iteration, it is useful to define the truth model to be the same as the design model. This part of the program is broken up into two menus, the first being

```
    CGTPI.ANALYZE MENU1
1    MODIFY THE GAINS
2      LOAD A FILE
3    BUILD THE SYSTEM
4       NEXT MENU
```

Option 1 allows the user to modify the gains KX and KZ directly, if that is desirable. It is an optional step that need not be exercised, but may allow additional flexibility in the design proce . This menu also allows the user to load a file, which is useful when comparing a controller against several truth models. The controller design is not dependent on the truth model, so this option is a time saving tool. It is a good idea, however, that the truth models have been run through the MODEL MENU

option of inputting the truth model, to ensure all variables (such as NSACT) have been generated. Note that, when using any of the SAVE FILE options in CGTPIKF, ALL of the variables on the stack are saved, so be careful to ensure that when saving truth models, only the truth model defining matrices are saved. Loading what you think is just a truth model file, but is in reality a file containing design model parameters which may be different from the ones you are presently using, would result in erroneous results and a great deal of frustration.

Before generating time response plots, the system must be constructed for simulation via Option 3 of CGTPI.ANALYZE MENU1, BUILD THE SYSTEM. This option prompts the user for initial values of the truth model and command model states. It also inquires whether or not actuator limits are to be included, and if so, what the lower and upper values are. If actuator limits are to be considered, the program then inquires whether antiwindup compensation is to be turned on. The program then uses MATRIX$_X$ SYSTEM BUILD [10] commands to build a simulation model. This process usually takes several minutes, so the user is cautioned to be patient.

After the system has been built, time response plots may be generated. Selecting Option 4 of MENU1 allows the user to proceed to CGTPI.ANALYZE MENU2, which is

```
    CGTPI.ANALYZE MENU2
1   PLOT DES MOD OUTPUT
2   PLOT COM MOD OUTPUT
3   TRUTH MODEL OUTPUTS
4   TRUTH MODEL STATES
5    LOOK AT VARIABLES
6    ACTUATOR RESPONSE
7         QUIT
```

A banner is displayed prior to this menu which explains each option. It is repeated here.

```
NOW YOU CAN GET TIME RESPONSE PLOTS.  DES MOD IS THE UNCONTROLLED

DESIGN MODEL.  COM MOD IS THE COMMAND MODEL.  TRUTH MODEL OUTPUTS

ARE THE TRUTH MODELS OUTPUTS WITH THE CONTROLLER IN THE LOOP.

ACTUATOR RESPONSE IS THE RESPONSE OF THE ACTUATORS.  IF NO ACTUATOR

DYNAMICS EXIST, THIS IS THE OPTIMAL CONTROL GENERATED BY THE

CONTROLLER. IT IS ALSO THE INPUT TO THE TRUTH MODEL BEING EVALUATED.
```

Option 7 in this menu, QUIT, results in returning to the CGTPI MENU. The remaining two options in that menu, SAVE VARIABLES and LOOK AT VARIABLES, have already been defined. Option 7, QUIT, returns the user to the CGTPIKF TOP MENU.

*A.2.6 KALMAN FILTER* Option 3 of the CGTPIKF TOP MENU allows the user to build and evaluate the Kalman filter. Selecting this option results in the menu

```
     KALMAN FILTER MENU
  1  MODIFY G,Q,R,OR QR
```

```
2    GENERATE GAINS,COV
3    COVARIANCE ANALYSIS
4          QUIT
```

to be displayed. The design model must exist to pursue Options 1 or 2, and both the design model and truth model must exist to pursue Option 3. Option 1 allows the design model's continuous-time GDM matrix (Equation (A.1)) to be modified. Also, the user may input the continuous-time noise strength matrix QDM and the discrete-time noise covariance matrices RDM and QRDM, where (dropping the DM notation)

$$\mathbf{Q}\delta(\tau) = \mathbf{E}\left\{\mathbf{w}(t)\mathbf{w}^T(t+\tau)\right\} \tag{A.7}$$

and $\mathbf{R}$ is the covariance of $\mathbf{v}$. QRDM is the discrete correlation between $\mathbf{w_d}$ and $\mathbf{v}$. When LTR tuning will be performed (to be discussed later), QDM must be square with the dimensions of the design model, and GDM should be the identity matrix.

Option 2 generates the Kalman filter gain $\mathbf{K}$ of Equation (2.74), which is called KFGAIN in this program, and the $\mathbf{P}(t_i^-)$ and $\mathbf{P}(t_i^+)$ matrices of Equations (2.73) and (2.76), appropriately called PFMINUS and PFPLUS. Also, the resulting eigenvalues, KFEVAL, are also displayed.

Option 3 performs a covariance analysis of the Kalman filter against the truth models. Option 4 returns the user to the CGTPIKF TOP MENU.

*A.2.7 Analyzing The Closed Loop CGT/PI/KF Controller* Option 4 of the CGTPIKF TOP MENU, ANALYZE CGT/PI/KF, allows the user to evaluate the performance of the closed loop system with the Kalman filter in the loop. This part of the program has almost the identical structure of the ANALYZE CGT/PI option of the CGT/PI CONTROLLER part of the program, with the exception that LTR tuning may be performed. As before, the truth model must exist before pursuing this part of the program. The first of the two menus for this option is

```
   CGTPIKF.ANLYZ MENU1
1  PERFORM LTR TUNING
2     LOAD A FILE
3   BUILD THE SYSTEM
4      NEXT MENU
```

Option 1 allows LTR tuning to be performed. When this option is pursued, the QDM matrix (the continuous-time dynamics driving noise strength matrix of the design model) must be square with dimensions of the number of design model states. The user will be prompted for a positive definite $\mathbf{V}$ matrix (an identity matrix is a reasonable initial choice), and for the scalar quantity q of the equation

$$\mathbf{Q}' = \mathbf{Q} + q^2\mathbf{B}\mathbf{V}\mathbf{B}^T \tag{A.8}$$

The remaining options are identical to those defined previously, with the exception that the user is also prompted for design model initial conditions when the system is built for the simulation. It should be noted that the control law is based on the Kalman filter estimates of $\hat{\mathbf{x}}(t_i^+)$ and $\hat{\mathbf{x}}(t_{i-1}^+)$.

The second menu of this option is

```
   CGTPIKF.ANLYZ MENU2
1  PLOT DES MOD OUTPUT
2  PLOT COM MOD OUTPUT
3  TRUTH MODEL OUTPUTS
4  TRUTH MODEL STATES
5   LOOK AT VARIABLES
6   ACTUATOR RESPONSE
7        QUIT
```

These options are also identical to those defined in Section A.2.5.

*A.2.8 Exiting CGTPIKF* When Option 5 of the CGTPIKF TOP MENU is selected, the following banner is displayed to the user:

```
IF YOU WISH TO SAVE THE CONTENTS OF THE STACK INTO A FILE, ENTER

A  0 <ZERO>. OTHERWISE, ENTER 1. REGARDLESS OF WHAT YOU ENTER, ALL

VARIABLES WILL REMAIN ON THE STACK FOR YOU TO DO WITH AS YOU

PLEASE IN THE MAIN MATRIXX PROGRAM ENVIRONMENT.

ENTER A  0 OR 1
```

Selecting '1' exits the program, '0' prompts the user for a filename, saves all variables on the stack to that file, and exits CGTPIKF. All variables listed in Section A.3 remain on the stack, for the user to evaluate or change.

## *A.3 Variable Definitions*

This section provides definitions for the variables that are typically on the stack at the end of a CGTPIKF session. Due to a MATRIX$_X$ restriction of only 144 variables existing at any given time, care has been taken throughout this program development to eliminate unnecessary variables while retaining those variables which would be useful to the control designer when using other MATRIX$_X$ options.

The variables remaining on the stack at the end of a sample run are:

```
User-defined variables ...
 A11          A13          A21          A22          A23          ACM          ACMD
 ACTUATOR     ADM          ADMD         AIM          ANM          ATM          ATMD
 BCM          BCMD         BDM          BDMD         BTM          BTMD         CCM
 CDM          CTM          DCM          DDM          DT           DTM          EVAL
 EX           EY           GDM          GNM          GTM          HDM          HTM
 KFEVAL       KFGAIN       KX           KXM          KXN          KZ           NINPUTS
 NMEAS        NSACT        NSCM         NSDM         NSNM         NSTM         PFMINUS
 PFPLUS       PIMAT        PLOWLIM      PTO          PUPLIM       QDM          QDMD
 QI           QIHAT        QLTR         QPRIME       QRDM         QTM          RDM
 RI           RIHAT        RLOWLIM      RTM          RUPLIM       SCM          SCMD
 SDM          SDMD         SIHAT        STM          STMD         SXU          SXUD
 TIME         TMNINPUTS    TMNMEAS      TMNOUPUTS    TTM          UCM          UM
 UR           URD          V            XC11         XC12         XC22         XCMO
 XDMO         XIE          XIED         XTM          XTMO         Y            YCMD
 YDMD         YTM
Permanent variables...
 EPS          EYE          FLOP         FTOMT        JAY          LBTOKG       PI
 RAND
using 3248 out of 100000 elements.
```

The permanent variables are inherent to MATRIX$_X$ , and the number of elements used depends on the size of the user-defined matrices. What follows is a definition of all of the above user-defined variables. Many of these matrices are generated internal to CGTPIKF. Those variables with a star (*) at the beginning of the variable name are input by the user.

A11: A CGT gain (Equation (2.18))

A13: A CGT gain (Equation (2.20))

A21: A CGT gain (Equation (2.21))

A22: A CGT gain (Equation (2.22))

A23: A CGT gain (Equation (2.23))

ACM*: The command model's continuous time **A** matrix

ACMD: The command model's discrete **A** matrix ($\mathbf{\Phi_m}$)

ACTUATOR: The matrix generated in the CGT/PI of CGT/PI/KF Analysis routines, Option 6. This matrix contains the response of the actuators, and can be plotted against the TIME matrix, if they have the same number of rows.

ADM*: The design model's continuous time **A** matrix

ADMD: The design model's discrete **A** matrix ($\mathbf{\Phi}$)

AIM*: The implicit model's continuous time **A** matrix

ANM*: The noise model's **A** matrix. Not presently used, but must be input.

ATM*: The truth model's continuous time **A** matrix

ATMD: The truth model's discrete **A** matrix ($\mathbf{\Phi_{tm}}$)

BCM*: The command model's continuous time **B** matrix

BCMD: The command model's discrete time **B** matrix

BDM*: The design model's continuous time **B** matrix

BDMD: The design model's discrete time **B** matrix

BTM*: The truth model's continuous time **B** matrix

BTMD: The truth model's discrete time **B** matrix

CCM*: The command model's C matrix

CDM*: The design model's **C** matrix

CTM*: The truth model's **C** matrix

DCM*: The command model's **D** matrix

DDM*: The design model's **D** matrix

DT*: The sampling time

DTM*: The truth model's **D** matrix

EVAL: Discrete closed-loop eigenvalues with PI gains in the full state feedback loop.

EX*: The matrix describing how time-correlated noise affects the states (Equation (2.1))

EY*: The matrix describing how time-correlated noise affects the system outputs (Equation (2.2))

GDM*: The design model's **G** matrix

GNM*: The noise model's **G** matrix

GTM*: The truth model's **G** matrix

HDM*: The design model's **H** matrix

HTM*: The truth model's **H** matrix

KFEVAL: Kalman filter's eigenvalues

KFGAIN: Kalman filter's gain matrix

KX : PI gain (Equation (2.61))

KXM: PI gain (Equation (2.61))

KXN: PI gain (Equation (2.61)). Not presently used.

KZ: PI gain (Equation (2.61))

NINPUTS: Number of inputs

NMEAS: Number of measurements

NSACT: Number of states of the actuators

NSCM: Number of states of the command model

NSDM: Number of states of the design model

NSNM: Number of states of the noise model

NSTM: Number of states of the truth model

PFMINUS: State covariance matrix prior to measurement update

PFPLUS: State covariance matrix after measurement update

PIMAT: $\Pi$ matrix (Equation (2.42))

PLOWLIM*: Actuator position lower limits

PT0*: Truth model's state covariance matrix initial conditions

PUPLIM*: Actuator position upper limits

QDM*: Design model's continuous noise strength matrix

QDMD: Design model's discrete noise covariance matrix

QI*: PI cost weighting matrix (Equation (2.80)).

QIHAT: PI cost weighting matrix (Equation (2.82))

QLTR*: Scalar value q used for LTR tuning (Equation (2.85))

QPRIME: Continuous noise strength as generated by LTR tuning (Equation (2.85))

QRDM*: Cross correlation term between $\mathbf{w_d}$ and $\mathbf{v}$ of design model

QTM*: Truth model's continuous noise strength matrix

RDM*: Design model's discrete-time measurement noise covariance matrix

RI*: PI cost weighting matrix (Equation (2.80)).

RIHAT: PI cost weighting matrix (Equation (2.84))

RLOWLIM*: Actuator rate lower limits

RTM*: Truth model measurement noise covariance matrix

RUPLIM*: Actuator rate upper limits

SCM: State space matrix consisting of command model **A**, **B**, **C**, and **D** matrices

SCMD: State space matrix consisting of discrete command model **A**, **B**, **C**, and **D** matrices

SDM: State space matrix consisting of design model **A**, **B**, **C**, and **D** matrices

SDMD: State space matrix consisting of discrete design model **A**, **B**, **C**, and **D** matrices

SIHAT: PI cost weighting matrix (Equation (2.83))

STM: State space matrix consisting of truth model **A**, **B**, **C**, and **D** matrices

STMD: State space matrix consisting of discrete truth model **A**, **B**, **C**, and **D** matrices

SXU*: PI cost weighting matrix (Equation (2.50) and (2.81)) off diagonal terms

SXUD: Discrete version of SXU

TIME: A matrix generated in all the CGT/PI of CGT/PI/KF Analysis routines. This matrix is the time vector, and can be used to plot actuator and plant re ponses.

TMNINPUTS: Number of truth model inputs. Currently set to NINPUTS

TMNMEAS: Number of truth model measurements. Currently set to NINPUTS

TMNOUTPUTS: Number of truth model outputs. Currently set to NINPUTS

TTM*: Matrix which transforms the truth model states to the

design model states

UCM: A matrix generated in all the CGT/PI of CGT/PI/KF ANALYSIS routines. This matrix is the command input to the command generator tracker, and hence the forcing function for all simulations.

UM*: PI cost weighting matrix (Equation (2.28)).

UR*: PI cost weighting matrix (Equation (2.50)).

URD: Discrete version of UR

V*: Used in LTR tuning (Equation (2.85))

XC11: PI cost weighting matrix (Equation (2.50))

XC12: PI cost weighting matrix (Equation (2.50))

XC22: PI cost weighting matrix (Equation (2.50))

XCM0: Command model state initial conditions

XDM0: Design model state initial conditions

XIE: PI cost weighting matrix (Equation (2.50) or (2.81))

XIED: Discrete version of XIE

XTM: The matrix generated in the CGT/PI of CGT/PI/KF Analysis routines, Option 4. This matrix contains the response of the truth model states, and can be plotted against the TIME matrix, if they have the same number of rows.

XTM0: Truth model state initial conditions

Y*: PI cost weighting matrix (Equation (2.28)).

YCMD: The matrix generated in the CGT/PI of CGT/PI/KF Analysis routines, Option 2. This matrix contains the response of the command model outputs, and can be plotted against the TIME matrix, if they have the same number of rows.

YDMD: The matrix generated in the CGT/PI of CGT/PI/KF Analysis

routines, Option 1. This matrix contains the response of the design model outputs, and can be plotted against the TIME matrix, if they have the same number of rows.

YTM: The matrix generated in the CGT/PI of CGT/PI/KF Analysis routines, Option 3. This matrix contains the response of the truth model outputs, and can be plotted against the TIME matrix, if they have the same number of rows.

### *A.4 Nuances of CGTPIKF*

1. When you see the *PAUSE* > prompt, the RETURN key will allow you to continue with the program.

2. You can change a matrix used by the program while in the main MATRIX$_X$ environme but should run through the part of the program that normally defines it anyhow, to ensure that all necessary variables used later in the program have been defined. For example, the truth model A matrix (ATM) can be entered or changed without going through the INPUT MODELS part of the program, but the variables NSTM, NSACT, and TTM will not be generated unless you do go through the TRUTH MODEL subroutine.

3. If a matrix already exists and you don't want to retype the whole matrix, simply type the matrix name. This is especially useful in conjunction with the previous hint.

4. For instant exit from the program, 'CONTROL C' has worked for the author. All variables will remain on the stack provided this mode of exit is not done in any of the following points in the program:

1. While pursuing an OPEN LOOP CGT
2. While pursuing a CLOSED LOOP CGT/PI, after the program has displayed

the comment 'THIS WILL TAKE A WHILE. PLEASE BE PATIENT.'

3. While pursuing the COVARIANCE ANALYSIS option.

If 'CONTROL C' is used at any of these points, the stack will not contain any user defined variables. All of the variables will, however, be in a file called 'GARBAGE.DAT'.

5. The MATRIX$_X$ error message "TOO MANY NAMES" means that too many user-defined names are on the stack. This may occur if the user has defined a large number (greater than ten) variables before running CGTPIKF. The solution to this problem is to clear the variables which are not necessary for execution of the program.

6. Every time a user is given a yes-or-no option to perform a task, he will be prompted for either a 1 or a 0. Neither of these numbers consistently represents either the "yes" or the "no". The author arbitrarily decided to make the 1 option result in what he considered to be the most likely design path, or the least detrimental option (that is, the option that is easiest from which to recover).

### *A.5 Required Files For Running This Software*

CGTPIKF is a highly modularized program. As a result, many files comprised of MATRIX$_X$ command files and functions are required to run it. The overall flow of the program was outlined in Section A.2, and this section tells which files are necessary to perform each function. This is not meant as a programming guide; rather, it is intended to allow the user to identify which files are required for execution of CGTPIKF (or a subset of the program), and where to look in the code if questions about the specific implementation of LQG theory arise. The file names are presented in a hierarchical fashion. The name of the file which calls subsequent files is listed, the file which calls it is identified in parenthesis, and the files it calls are listed under it. How a sub-file is called is listed next to the file

name. Most of the files are $MATRIX_X$ command files, and are executed with the $MATRIX_X$ command EXECUTE('filename'). Some files are $MATRIX_X$ user-defined functions, rather than command files, and these are specifically identified by (function) next to the file name. $MATRIX_X$ user-defined functions are implemented in the following manner: first they are defined by DEFINE 'filename', then executed by typing the output variables of the function in square brackets, an = sign, followed by the function name with input variables in parenthesis (e.g., [output variables]=FUNCTION(input variables)). Refer to the $MATRIX_X$ reference manual [10], Chapter 7, for details on both command files and user-defined functions.

CGTPIKF. (top level program)

| | |
|---|---|
| MODELINPUTS. | (INPUT MODELS menu option) Allows user to input models. |
| CGTPI. | (CGTPI CONTROLLER menu option) Design and evaluation of open-loop CGT and closed-loop CGT/PI deterministic controllers, and PI regulators. |
| KF. | (KALMAN FILTER menu option) Design and evaluation of Kalman filters |

MODEL INPUTS. (called by CGTPIKF)

| | |
|---|---|
| INPUTFILE. | (LOAD FILE menu option) Allows user to load a file from memory. |
| DESMODEL. | (DESIGN MODEL menu option) Handles input of design model matrices. |
| NOISEMOD. | (NOISE MODEL menu option) Handles input of noise model matrices. |
| COMMODEL. | (COMMAND MODEL menu option) Handles input |

of command model matrices.

TRUMODEL. (TRUTH MODEL menu option) Handles input of truth model matrices.

IMPLICIT.MODEL (IMPLICIT MODEL menu option) Handles input of implicit model matrix.

SAVEFILE. (SAVE FILE menu option) Allows user to save a file to memory.

CGTPI. (called by CGTPIKF.)

CGT (Called by OPEN LOOP CGT menu option and by the PI command file) Generates the CGT gains $\mathbf{A_{11}}, \mathbf{A_{21}}, \mathbf{A_{13}}, \mathbf{A_{22}}$ and $\mathbf{A_{23}}$.

PI. (PI REGULATOR menu option) Generates the PI gains $\mathbf{K_x}$ and $\mathbf{K_z}$, and optionally $\mathbf{K_{xm}}$

CGTPI.ANALYZE (ANALYZE CGT/PI menu option) Uses SYSTEM BUILD commands to build the simulation, and generates time response plots.

SAVEFILE. (SAVE FILE menu option) Allows user to save a file to memory.

CGT (called by CGTPI. or PI.) (Function)

BARRAUD (Function) Generates the solution to $\mathbf{X} = \mathbf{AXB} + \mathbf{C}$ [2]

PI. (called by CGTPI. )

INPUTCOST. Allows user to input the cost weighting matrices **Y, UM. QI, RI, UM.** Calls INPUTFILE.

IMPLICIT.MODEL (IMPLICIT MODEL menu option) Handles input of implicit model matrix.

EXPLICIT (Function) Generates explicit continuous time cost weighting matrices.

IMPLICIT (Function) Generates implicit continuous time cost weighting matrices.

DISCOST (Function) Discretizes the augmented A and B matrices, and the cost weighting matrices.

CGT (Called by OPEN LOOP CGT menu option and by the PI command file) Generates the CGT gains $\mathbf{A_{11}}, \mathbf{A_{21}}, \mathbf{A_{13}}, \mathbf{A_{22}}$ and $\mathbf{A_{23}}$.

CGTPI. ANALYZE (called by CGTPI.)

INPUTFILE. (LOAD A FILE menu option) Allows user to load a file from memory.

CONTROL. (BUILD THE SYSTEM menu option) Builds the simulation via SYSTEM BUILD commands

PLOTHELP (Function) Used in the plotting options. Generates the time and command matrices required to do simulations using the SYSTEM BUILD SIM command.

PLOTLOOP (Function) Used in the plotting options. Generates plots and allows the user to reduce the number of variables being plotted.

ANALYZE.YCMD (PLOT COM MOD OUTPUT menu option) SYSTEM BUILD commands which allow for later command model simulation.

ANALYZE.YTM (TRUTH MODEL OUTPUTS menu option)

| | |
|---|---|
| | SYSTEM BUILD commands which allow for later truth model simulation. |
| ANALYZE.XTM | (TRUTH MODEL STATES menu option) SYSTEM BUILD commands which allow for later truth model simulation. |
| ANALYZE.ACT | (OPTIMAL CONTROL U menu option) SYSTEM BUILD commands which allow for later actuator simulation. |

CONTROL. (called by CGTPI.ANALYZE)

| | |
|---|---|
| KZYCM.BLOCKS | This command file, as well as the next two, generate the SYSTEM BUILD super-block which represents the term $\mathbf{K_z}[[\mathbf{C_m x_m}(t_{i-1})] + [\mathbf{D_m u_m}(t_i)]]$ of Equation (2.61) for the simulation. |
| KZYCM.CONNECT2 | |
| KZYCM.CONNECT1 | |
| KZYTM.BLOCKS | This command file, as well as the next two, generate the super-block which represents the term $\mathbf{K_z}[[\mathbf{C}\mathbf{x}(t_{i-1})] + [\mathbf{D}\mathbf{u}(t_i)]]$ of Equation (2.61) for the simulation. |
| KZYTM.CONNECT2 | |
| KZYTM.CONNECT1 | |
| GAIN1.BLOCKS | This command file, as well as the next five, generate the term $\mathbf{K_x}[\mathbf{x}(t_i) - \mathbf{x}(t_{i-1})]$ of Equation (2.61) |
| GAIN1.SUM | |

| | |
|---|---|
| GAIN1.INTCON2 | |
| GAIN1.INTCON1 | |
| GAIN1.EXTCON2 | |
| GAIN1.EXTCON1 | |
| GAIN3.BLOCKS | This command file, as well as the next five, generate the term $[\mathbf{K_x A_{11}} + \mathbf{A_{21}}]$ $[\mathbf{x_m}(t_i) - \mathbf{x_m}(t_{i-1})]$ of Equation (2.61) |
| GAIN3.SUM | |
| GAIN3.INTCON2 | |
| GAIN3.INTCON1 | |
| GAIN3.EXTCON2 | |
| GAIN3.EXTCON1 | |
| BLOCK1.BLOCKS | This command file, as well as the next three, generate the super-block BLOCK1, which incorporates super-blocks KZYCM and GAIN3. |
| BLOCK1.CONNECT2 | |
| BLOCK1.CONNECT1 | |
| BLOCK1.SUM | |
| LAW.BLOCKS0 | This command file, as well as the next four, put together all of the above super-blocks into a super-block called LAW, which represents the control law of Equation (2.61) |
| LAW.BLOCKS1 | Incorporates antiwindup compensation in LAW |
| LAW.SUM | |
| LAW.CONNECT2 | |

| | |
|---|---|
| LAW.CONNECT1 | |
| ACT.BLOCKS0 | This command file, as well as the next three, generate the super-block ACT, which incorporates actuator dynamics and limits. |
| ACT.BLOCKS1 | |
| ACT.BLOCKS2 | |
| ACT.NOLIMITS | |
| YTM.2 | This command file, as well as the next one, creates a super-block which generates the truth model output response |
| YTM.1 | |
| XTM.2 | This command file, as well as the next one, creates a super-block which generates the truth model state response |
| XTM.1 | |
| YCMD.2 | This command file, as well as the next one, creates a super-block which generates the command model output response |
| YCMD.1 | |

KF. (called by CGTPIKF.)

| | |
|---|---|
| INPUT.KF | (MODIFY G,Q,R,OR QR menu option) Allows user to input noise matrices. |
| COV.ANALYSIS | (COVARIANCE ANALYSIS menu option) Performs covariance analysis. |

CGTPIKF. ANALYZE (called by CGTPIKF.)

| | |
|---|---|
| INPUTFILE. | (LOAD A FILE menu option) Allows user to load a file from memory. |
| LTR. | (PERFORM LTR TUNING menu option) Allows user to specify V and QLTR, and forms QPRIME |
| CONTROL.WITHKF | (BUILD THE SYSTEM menu option) Builds the simulation, with Kalman filter, via SYSTEM BUILD commands |
| PLOTHELP | (Function) Used in the plotting options. Generates the time and command matrices required to do simulations using the SYSTEM BUILD SIM command. |
| PLOTLOOP | (Function) Used in the plotting options. Generates plots and allows the user to reduce the number of variables being plotted. |
| ANALYZE.YCMD | (PLOT COM MOD OUTPUT menu option) SYSTEM BUILD commands which allow for later command model simulation. |
| ANALYZE.YTMF | (TRUTH MODEL OUTPUTS menu option) SYSTEM BUILD commands which allow for later truth model simulation with the Kalman filter in the loop. |
| ANALYZE.XTMF | (TRUTH MODEL STATES menu option) SYSTEM BUILD commands which allow for later truth model simulation with the Kalman filter in the loop. |
| ANALYZE.ACTF | (OPTIMAL CONTROL U menu option) |

SYSTEM BUILD commands which allow for later actuator simulation with the Kalman filter in the loop.

CONTROL. WITHKF (called by CGTPI.ANALYZE)

| | |
|---|---|
| KF.BUILD | This block generates the Kalman filter. |
| KZUDM.BLOCKS | This command file, as well as the next two, generate the SYSTEM BUILD super-block which represents the term $\mathbf{K_z}[\mathbf{D_m u_m}(t_i)]$ of Equation (2.61) for the simulation. |
| KZUDM.CONNECT2 | |
| KZUDM.CONNECT1 | |
| GAIN1F.BLOCKS | This command file, as well as the next five, generate the term $\mathbf{K_x}[\hat{\mathbf{x}}(t_i^+) - \hat{\mathbf{x}}(t_{i-1}^+)]$ of Equation (2.61) |
| GAIN1F.SUM | |
| GAIN1F.CONNECT2 | |
| GAIN1F.CONNECT1 | |
| BLOCK5.BLOCKS | This command file, as well as the next three, generate the super-block BLOCK5, which incorporates GAIN1F and KZUDM, and the term $\mathbf{K_z}[\mathbf{C}\hat{\mathbf{x}}(t_{i-1}^+)]$ |
| BLOCK5.CONNECT2 | |
| BLOCK5.CONNECT1 | |
| BLOCK5.INTCON | |
| BLOCK5.SUM | |
| LAWF.BLOCKS0 | This command file, as well as the next |

| | |
|---|---|
| | five, put together all of the above super-blocks into the control law of Equation (2.61) |
| LAWF.BLOCKS1 | Incorporates antiwindup compensation in LAW |
| LAWF.SUM1 | |
| LAWF.CONNECT2 | |
| LAWF.CONNECT1 | |
| LAWF.INTCON | |
| ACTF.BLOCKS0 | This command file, as well as the next three, generate the super-block ACT, which incorporates actuator dynamics and limits. |
| ACTF.BLOCKS1 | |
| ACTF.BLOCKS2 | |
| ACTF.NOLIMITS | |
| YTMF.2 | This command file, as well as the next one, creates a super-block which generates the truth model output response |
| YTMF.1 | |
| XTMF.2 | This command file, as well as the next one, creates a super-block which generates the truth model state response |
| XTMF.1 | |

KF.BUILD (Called by CGTPIKF.ANALYZE)

| | |
|---|---|
| XHATMINUS.BLOCKS | This command file, as well as the next two, generate the super-block XHATMINUS |
| XHATMINUS.SUM1 | |

| | |
|---|---|
| XHATMINUS.SUM2 | |
| KF.BLOCKS | This command file, as well as the next five, generate the super block FILTER |
| KF.CONNECT2 | |
| KF.CONNECT1 | |
| KF.EXTCON | |
| KF.SUM1 | |
| KF.SUM2 | |

### *A.6 Simulation Description*

What follows is an explanation of how the simulation is performed. Some knowledge of MATRIX$_X$ SYSTEM BUILD [10] will be useful, but not required. The super-blocks which formulate the simulation are nested, and this discussion will work from the top-most level down. The deterministic CGT/PI control law formulation and evaluation will be addressed first, followed by the CGT/PI/KF. For all super-blocks, internal blocks are numbered clockwise from 1 in the upper left hand corner to 6 in the lower left hand corner.

*A.6.1 Full-State Feedback Simulation* Figure A.8 shows the top level super-block of the deterministic simulation. ACT is a super-block which generates the control law and actuator dynamics and nonlinearities. The input to ACT is the command model input UCM, $\mathbf{u_m}$ of Equation (2.4). The output drives the truth model system matrix, STM. Note that this matrix only represents the plant; actuator states are separately identified in the ACT block. The outputs of super-block YTM are the truth model outputs. Figure A.9 is almost identical to super-block YTM, except that the C matrix is the identity matrix, resulting in the outputs being the truth model state response.

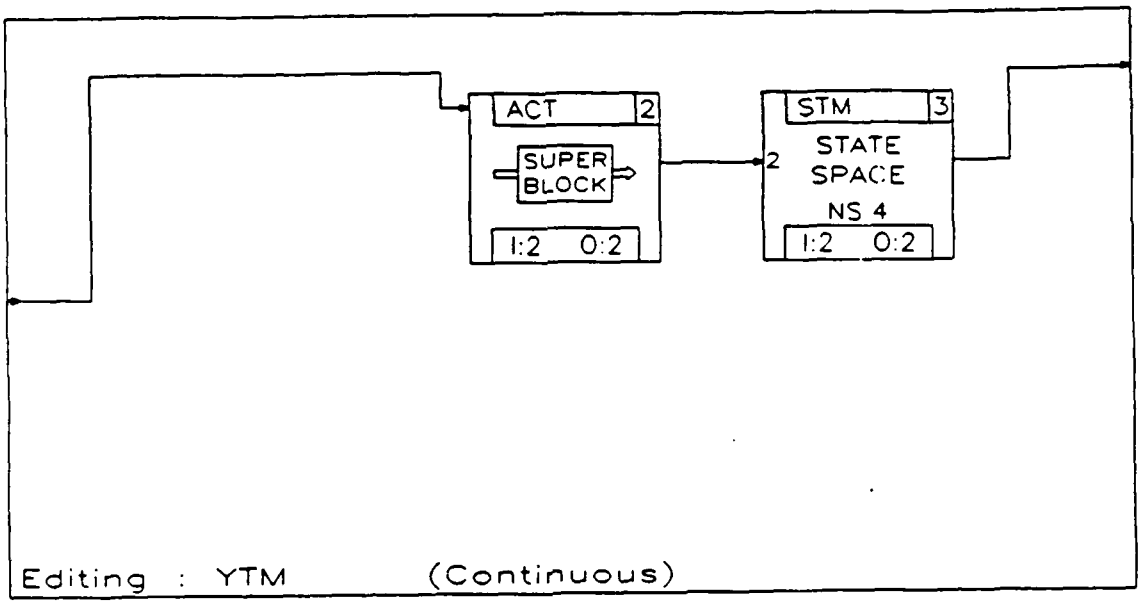



Figure A.8. The YTM Super-Block

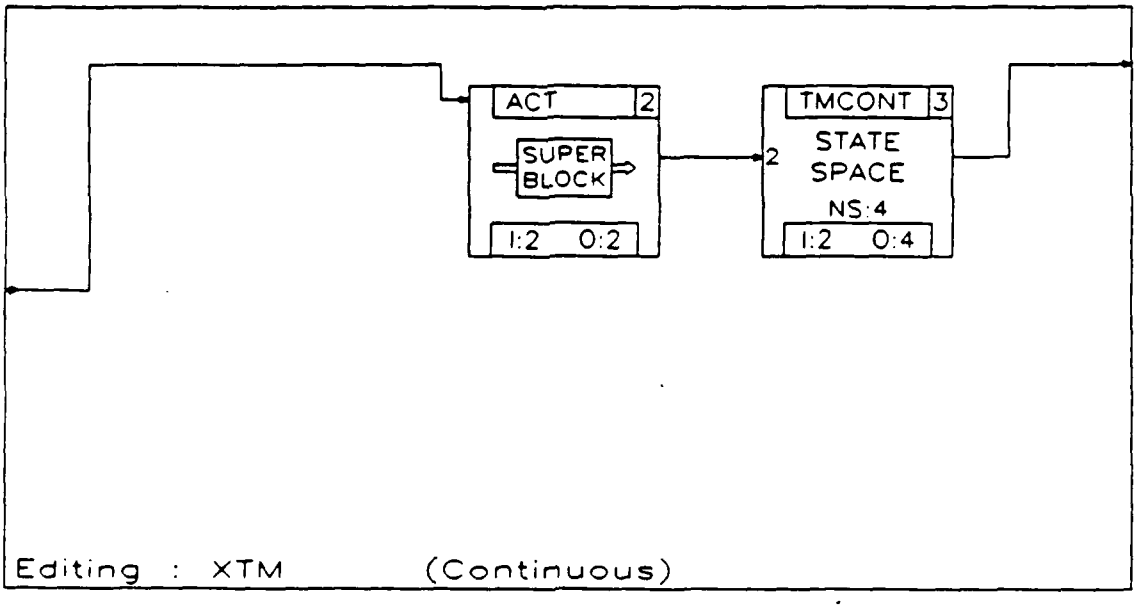



Figure A.9. The XTM Super-Block

Figure A.10 shows the super-block ACT. The input to the block is again UCM, and this is fed into super-block LAW, which generates the closed-loop CGT/PI control law. The output of LAW is the optimal control $\mathbf{u}(t_i)$, as in Equation (2.61). Block 2 represents actuator dynamics above first order. If these dynamics do not exist, this block becomes the identity matrix. RLIMIT in the Block 3 position imposes actuator rate limits, if they exist. ACT1 represents first order actuator dynamics, while PLIMIT imposes actuator position limits. For blocks 2 through 5, the identity matrix will be substituted if the respective dynamics or limits do not exist. The outputs of the ACT super-block are the actuator responses.

The LAW block is the heart of the deterministic simulation, and is shown in Figure A.11. As previously stated, the input is the command model input, UCM, and the output is the optimal control $\mathbf{u}$, which was given in Equation (2.61). Super-block BLOCK1 (Figure A.12) is actually two super-blocks, those being KZYCM (Figure A.13) and GAIN3 (Figure A.14). KZYCM generates the term $\mathbf{K_z}[[\mathbf{C_m x_m}(t_{i-1})]+[\mathbf{D_m u_m}(t_i)]]$, and GAIN3 generates the term $[\mathbf{K_x A_{11}} + \mathbf{A_{21}}][\mathbf{x_m}(t_i) - \mathbf{x_m}(t_{i-1})]$.

Returning to the LAW block, the optimal control law is the output of Block 3, the summing junction. This is then fed through Block 6, the antiwindup compensation, if that option has been selected; otherwise Block 6 is the identity matrix. The output of Block 6 is fed through a time delay to generate $\mathbf{u}(t_{i-1})$, and it also is fed to Blocks 4 and 5. At this point it is useful to note that, for this simulation, the plant is simulated inside the LAW block, thereby eliminating the need of any complicated feedback paths from super-block YTM to represent full-state feedback. KZYTM (Figure A.15) takes the optimal control law as its input, feeds this through the truth model representation (the STMD matrix makes up Block 1), time delays it, and multiplies the result by KZ, to get the term $\mathbf{K_z}[[\mathbf{C x}(t_{i-1})] + [\mathbf{D u}(t_i)]]$.

The GAIN1 super-block (Figure A.16) generates the term $\mathbf{K_x}[\mathbf{x}(t_i) - \mathbf{x}(t_{i-1})]$.

Figure A.10 shows the super-block ACT. The input to the block is again UCM, and this is fed into super-block LAW, which generates the closed-loop CGT/PI control law. The output of LAW is the optimal control **u**. Block 2 represents actuator dynamics above first order. If these dynamics do not exist, this block becomes the identity matrix. RLIMIT in the Block 3 position imposes actuator limits, if they exist. ACT1 represents first order actuator dynamics, while PLIMIT imposes actuator position limits. For blocks 2 through 5, the identity matrix will be substituted if the respective dynamics or limits do not exist. The outputs of the ACT super-block are the actuator responses. The LAW block is the heart of the deterministic simulation, and is shown in Figure A.11. As previously stated, the input is the command model input, UCM, and the output is the optimal control **u**, which was given in Equation (2.61). Super-block BLOCK1 (Figure A.12) is actually two super-blocks, those being KZYCM (Figure A.13) and GAIN3 (Figure A.14). KZYCM generates the term $\mathbf{K_z}[[\mathbf{C_m x_m}(t_{i-1})] + [\mathbf{D_m u_m}(t_i)]]$ GAIN3 generates the term $[\mathbf{K_x A_{11}} + \mathbf{A_{21}}][\mathbf{x_m}(t_i) - \mathbf{x_m}(t_{i-1})]$.

Returning to the LAW block, the optimal control law is the output of Block 3, the summing junction. This is then fed through Block 6, the antiwindup compensation, if that option has been selected; otherwise Block 6 is the identity matrix. The output of Block 6 is fed through a time delay to generate $\mathbf{u}(t_{i-1})$, and it also is fed to Blocks 4 and 5. At this point it is useful to note that for this simulation, the plant is simulated inside the LAW block, thereby eliminating the need of any complicated feedback paths from super-block YTM to represent full-state feedback. KZYTM (Figure A.15) takes the optimal control law as its input, feeds this through the truth model representation (the STMD matrix makes up Block 1), time delays it, and multiplies the result by KZ, to get the term $\mathbf{K_z}[[\mathbf{C x}(t_{i-1})] + [\mathbf{D u}(t_i)]]$.

The GAIN1 super-block (Figure A.16) generates the term $\mathbf{K_x}[\mathbf{x}(t_i) - \mathbf{x}(t_{i-1})]$. The TTM matrix (Block 4) is used to transform the vector coming out of the summing junction, which has NSTM states, into a vector with NSDM states, to be

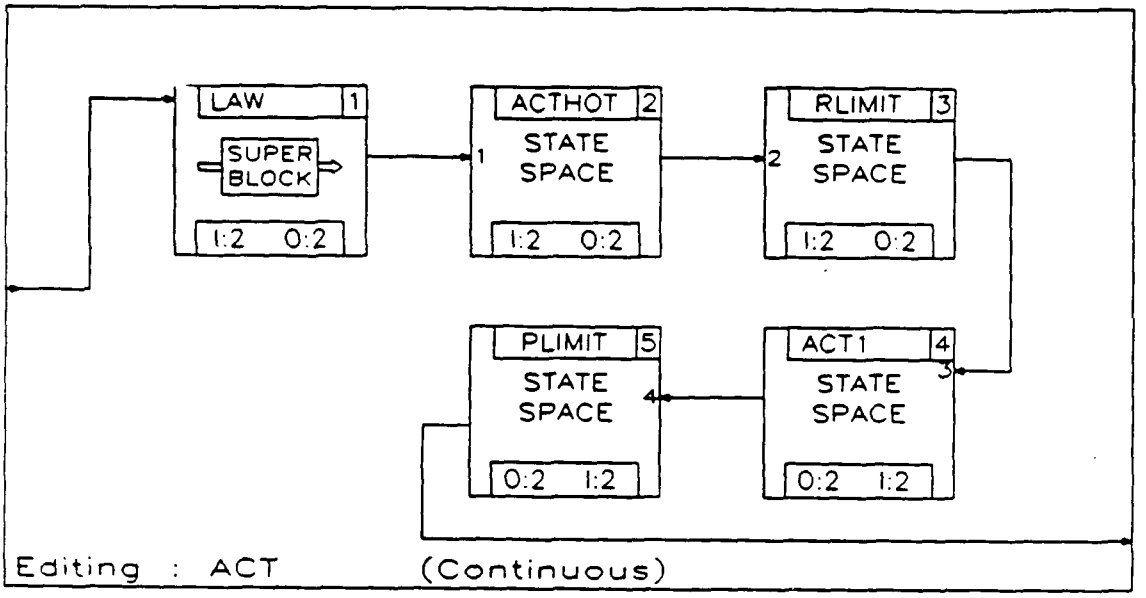



Figure A.10. The ACT Super-Block

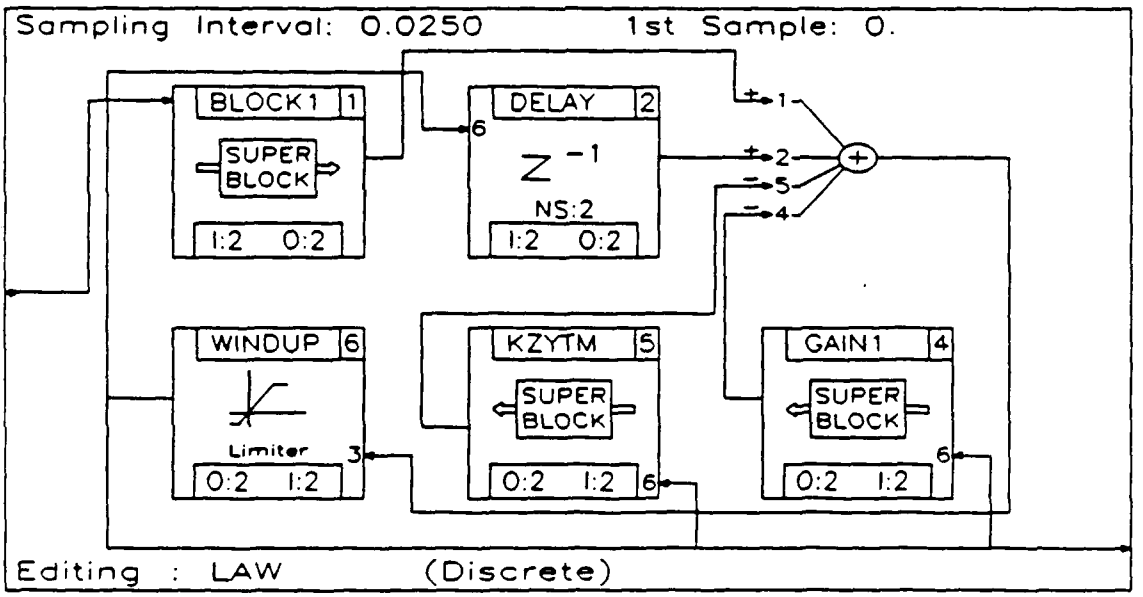



Figure A.11. The LAW Super-Block

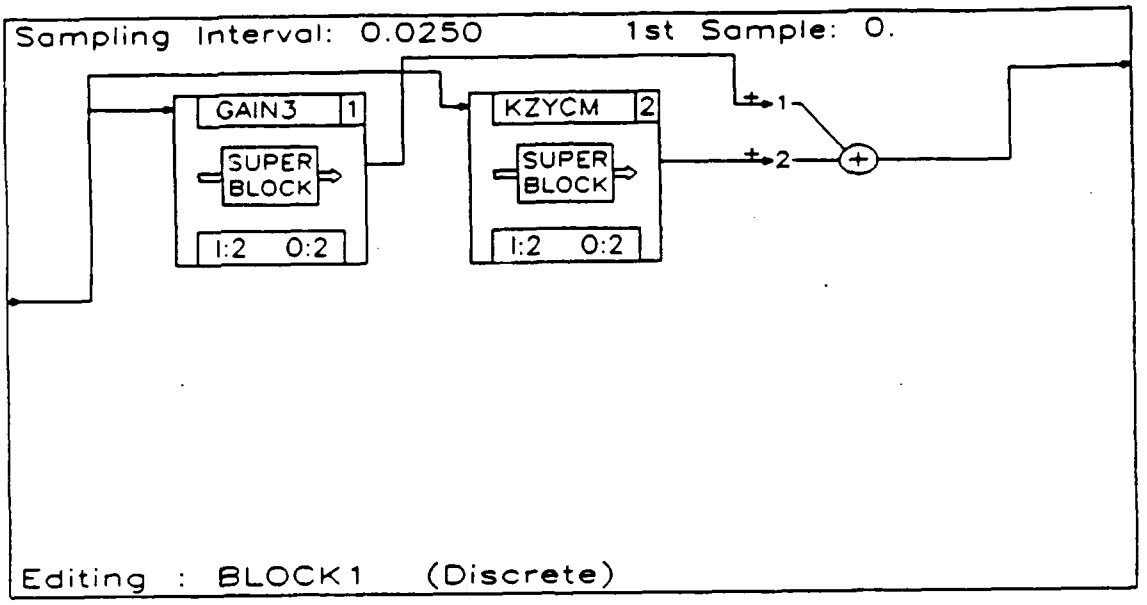



Figure A.12. The BLOCK1 Super-Block

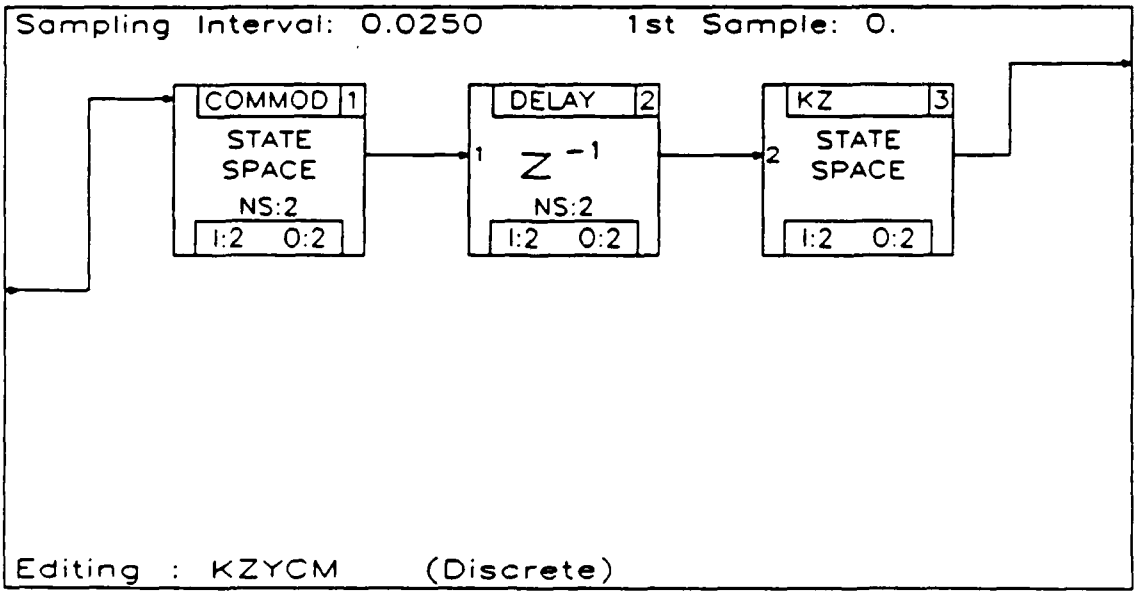



Figure A.13. The KZYCM Super-Block

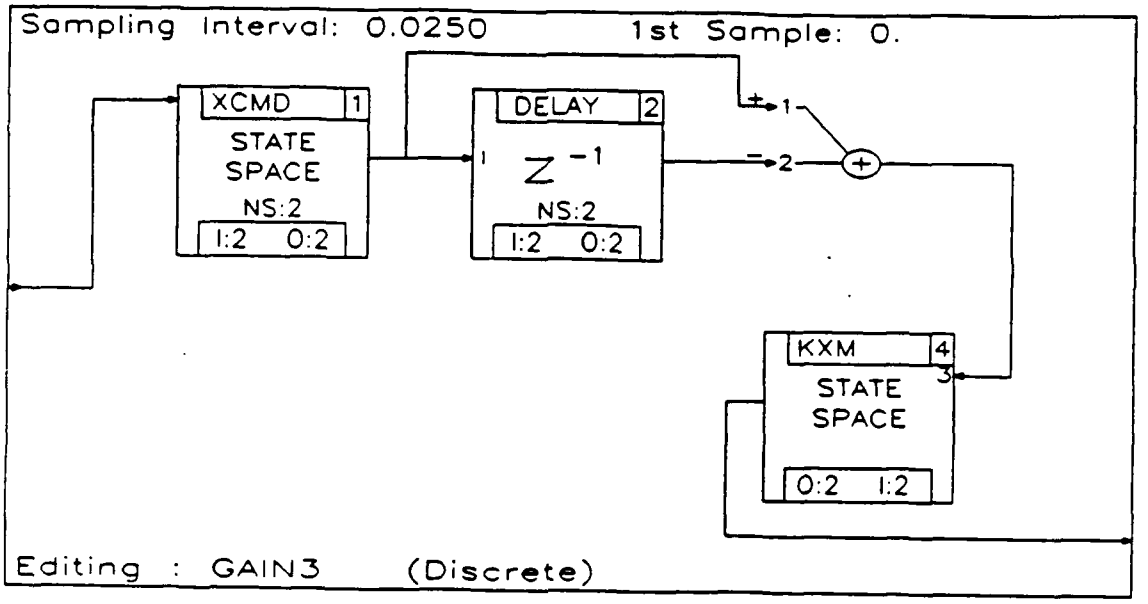



Figure A.14. The GAIN3 Super-Block

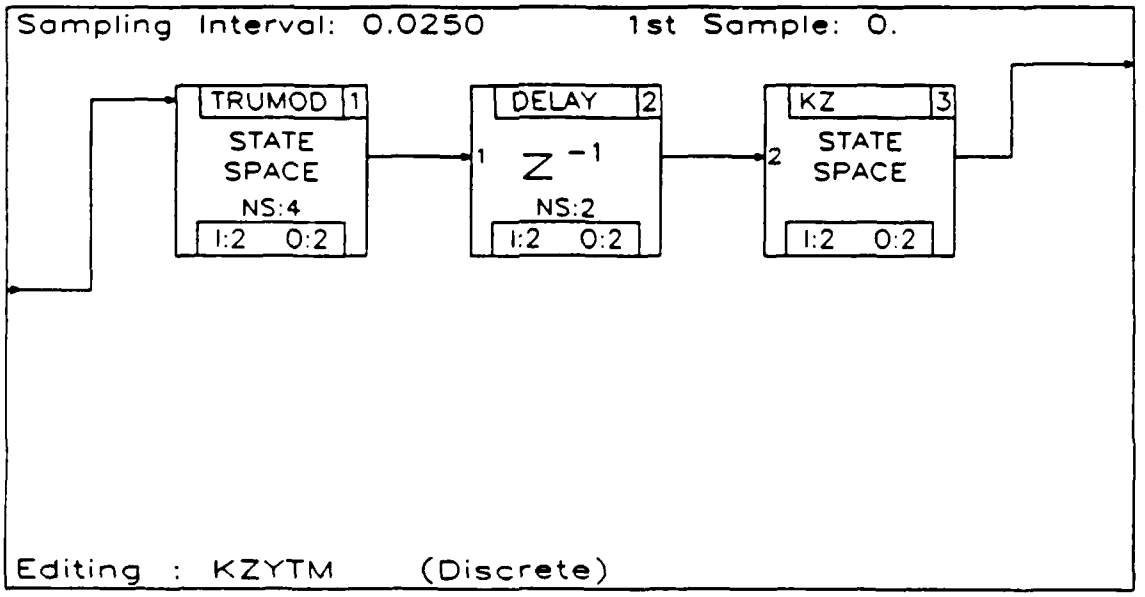



Figure A.15. The KZYTM Super-Block

of the correct dimensions for multiplication by KX. This concludes the discussion of the CGT/PI simulation.

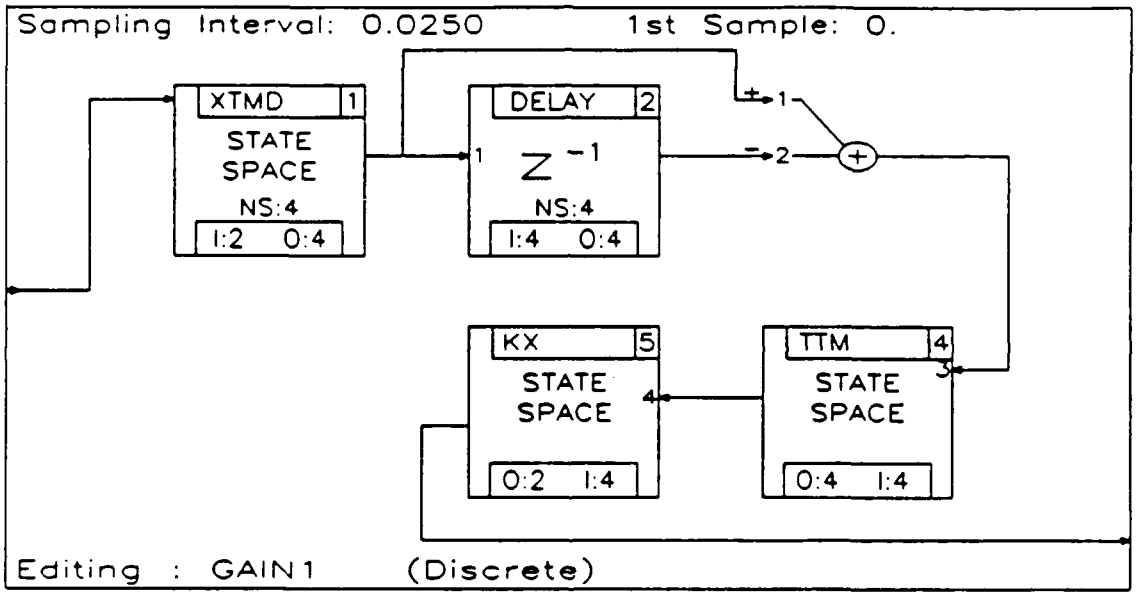



Figure A.16. The GAIN1 Super-Block

*A.6.2 Filter-in-the-Loop Simulation* Turning our attention to super-block YTMF (Figure A.17), we see that the structure is identical to that of YTM. This is also true of XTMF and ACTF (Figures A.18 and A.19). The 'F' is added to the designation to identify these blocks as being associated with the Kalman filter.

Figure A.20 shows the super-block LAWF. This super-block creates the CGT/PI/KF control law. Its structure has been changed from that of the LAW super-block to allow for the incorporation of the Kalman filter. Super-block BLOCK1 is identical to that used by LAW. The antiwindup compensation is in the Block 4 position, but functionally it has not changed. The FILTER super-block (Figure A.21) generates the system state estimates. The XHAT- ("XHAT minus") super-block generates the state estimate prior to measurement update. The optimal control comes into Block 1, which generates the measurement vector $\mathbf{z}(t_i)$. Block 2 generates $\mathbf{H}\hat{\mathbf{x}}(t_i^-)$, and the difference of these two vectors is multiplied by the Kalman filter gain matrix. When this product is added to the state estimate (prior to measurement update), the resulting vector is the state estimate $\hat{\mathbf{x}}(t_i^+)$, as in Equation (2.75).

The $\hat{\mathbf{x}}(t_i^-)$ estimate of Equation (2.72) is generated in super-block XHAT-. The input to the block consists of both the optimal control, which is fed into Block 1, and the $\hat{\mathbf{x}}(t_i^+)$ estimate, which is fed through the state transition matrix of Block 2. Note that Blocks 1 and 2 do not have states, but are rather algebraic loops. Hence the requirement of adding the time delay.

Super-block BLOCK5 of the LAWF super-block is shown in Figure A.23. The inputs to this super-block are the optimal control $\mathbf{u}$, which is fed into super-block KZUDM, and the state estimates, which are input to super-block GAIN1F, and to Block4. GAIN1F (Figure A.24) performs the same function as GAIN1 did. KZUDM generates the term $\mathbf{K_z}[\mathbf{D_m u_m}(t_i)]]$, as is shown in Figure A.25.

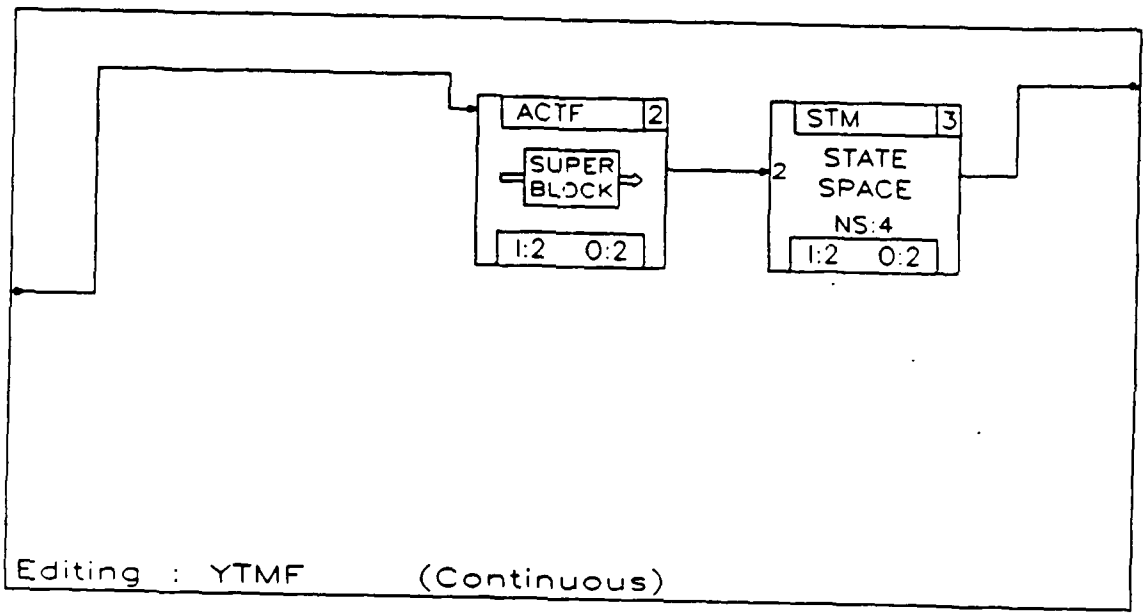



Figure A.17. The YTMF Super-Block

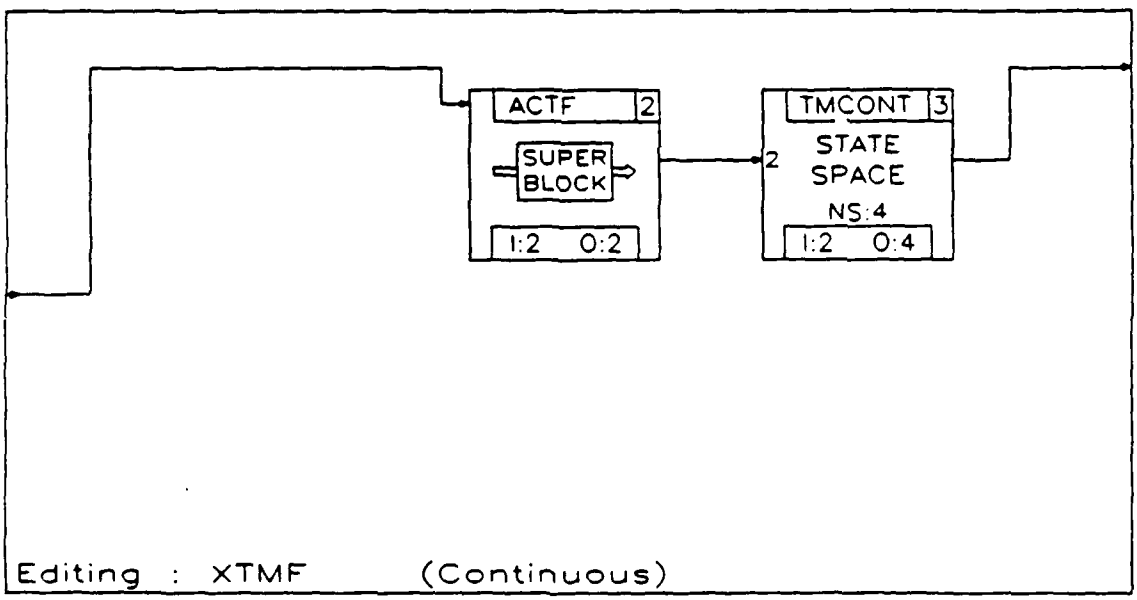



Figure A.18. The XTMF Super-Block

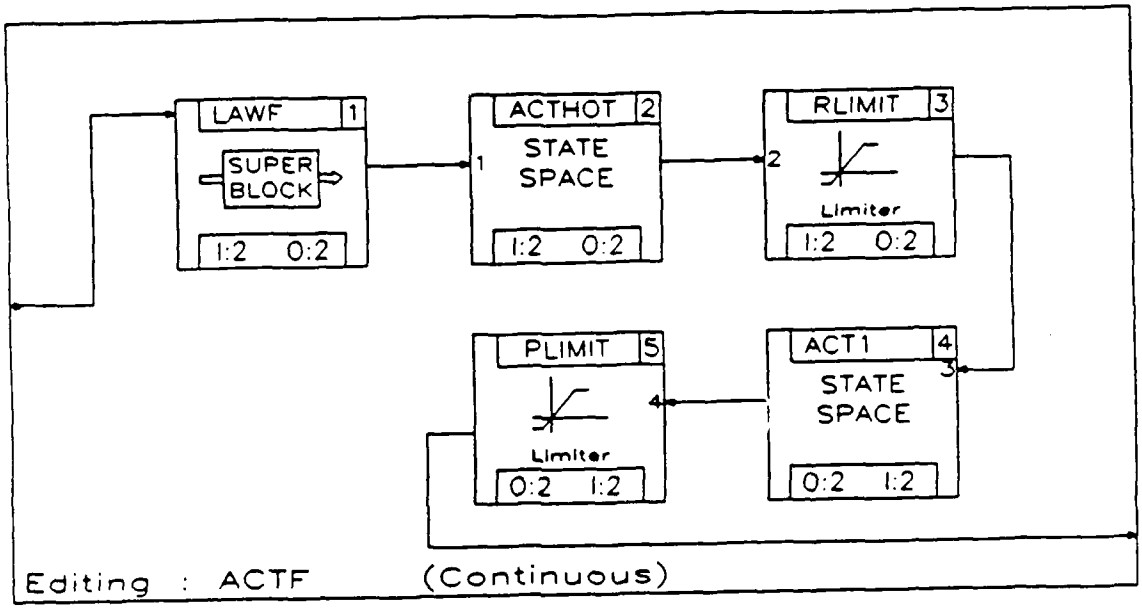



Figure A.19. The ACTF Super-Block

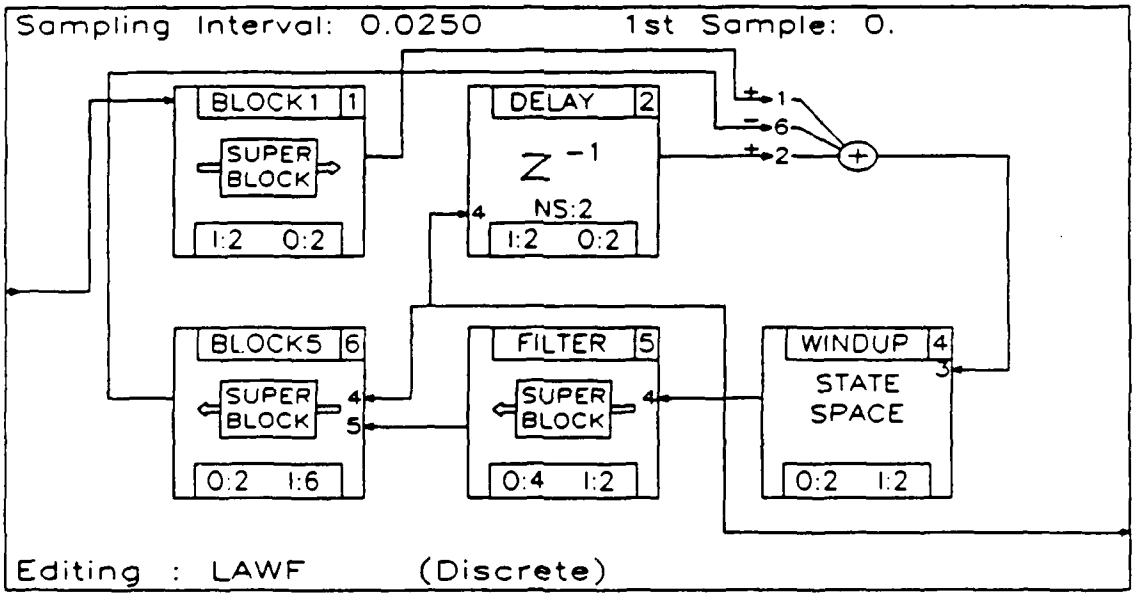



Figure A.20. The LAWF Super-Block

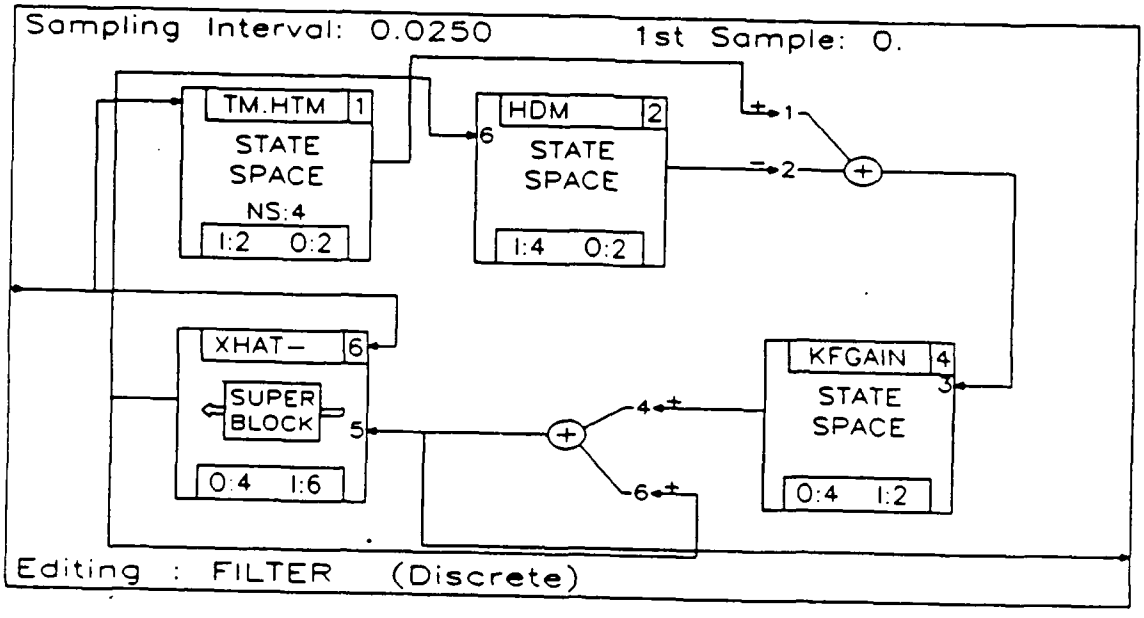



Figure A.21. The FILTER Super-Block

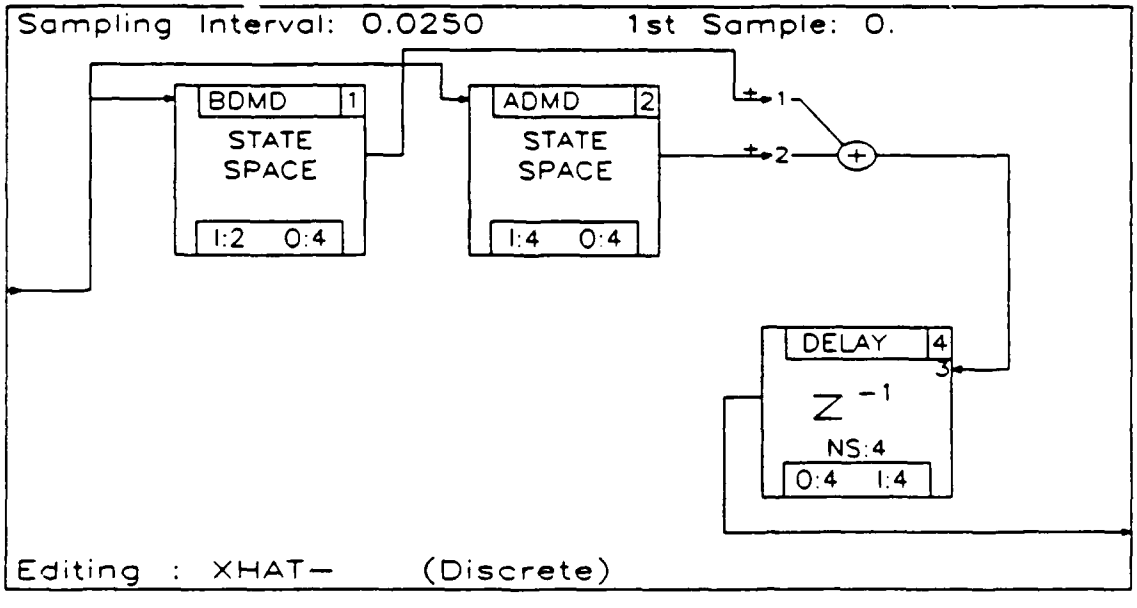



Figure A.22. The XHAT- Super-Block

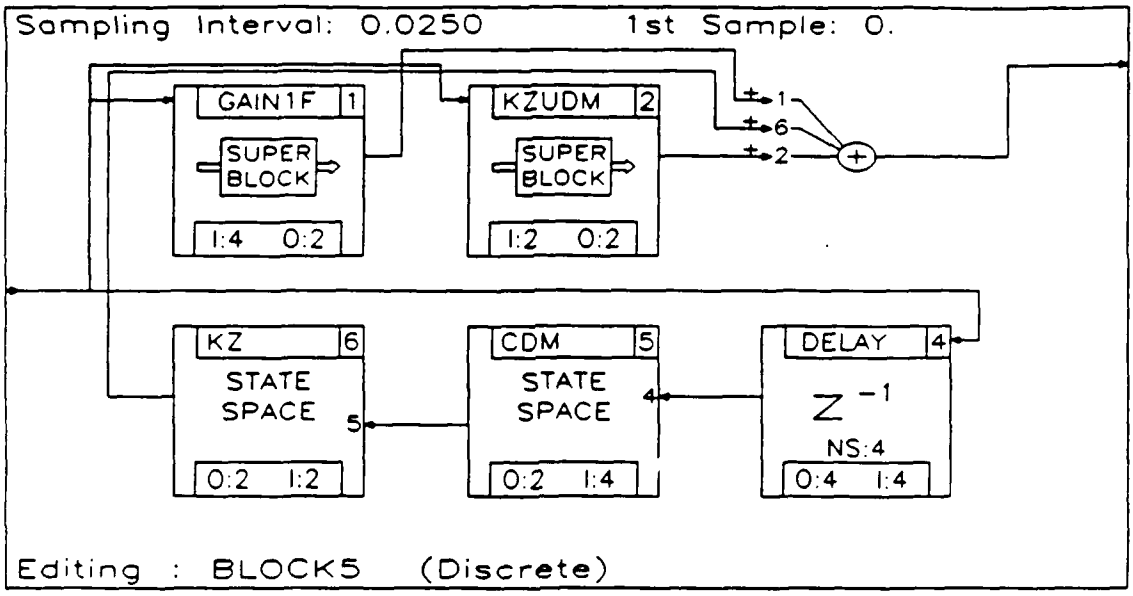



Figure A.23. The BLOCK5 Super-Block

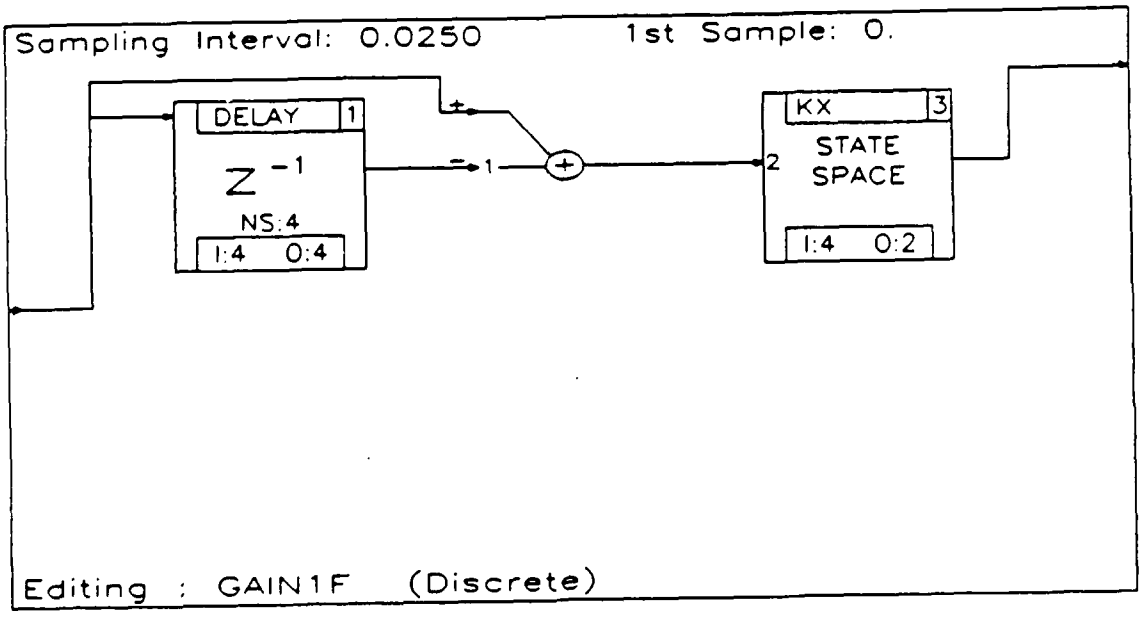



Figure A.24. The GAIN1F Super-Block

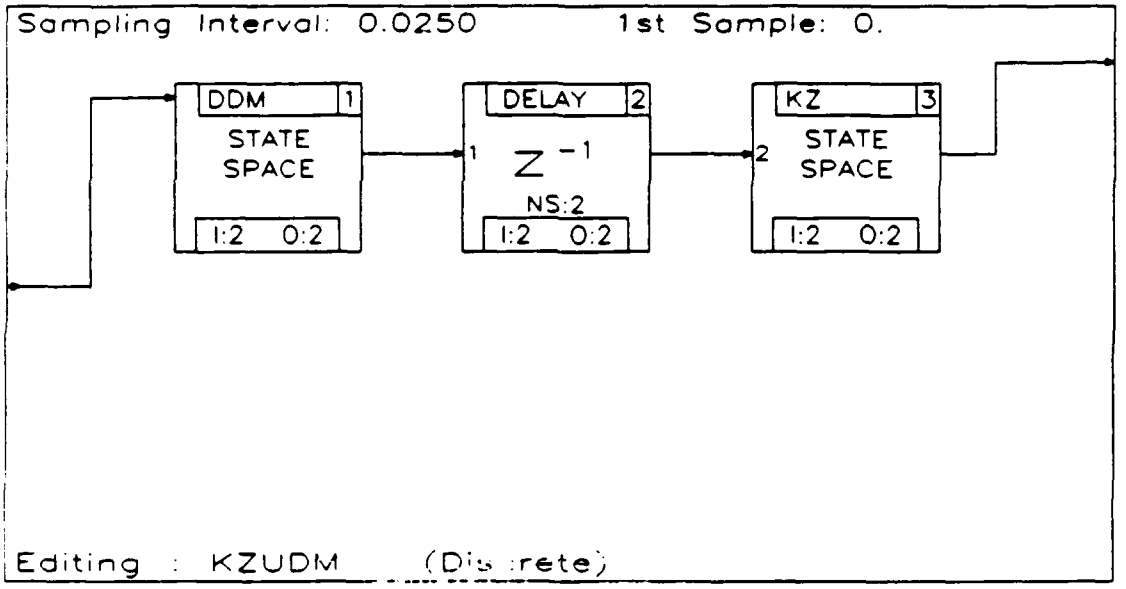



Figure A.25. The KZUDM Super-Block

## *Bibliography*


1. ASD/ENESS. Military Specification - Flying Qualities of Piloted Aircraft. November 1980. Mil-F-8785C, Wright Patterson AFB, OH.
2. A. Y. Barraud. A Numerical Algorithm to Solve $A^T X A - X = Q$. *IEEE Trans. Automat. Control AC-22 (5)*, 883–884, Oct 1977.
3. J. Blakelock. *Automatic Control of Aircraft and Missiles.* John Wiley and Sons, Inc., New York, 1965.
4. J. J. D'Azzo and C. H. Houpis. *Linear Control Systems Analysis and Design.* McGraw-Hill Book Company, New York, 1981.
5. R. M. Floyd. *Design of Advanced Digital Flight Control Systems Via Command Generator Tracker (CGT) Synthesis Methods.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1981.
6. E. G. Gilbert. Conditions for Minimizing the Norm Sensitivity of Characteristic roots. *Conference on Information Sciences and Systems, Baltimore, Maryland.*, March 1983.
7. G. L. Gross. *LQG/LTR Design of a Robust Flight Controller for the Stol F-15.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1985.
8. D. Hammond. *Multivariable Control Law Design For The Control Reconfigurable Combat Aircraft (CRCA).* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.
9. R.A. Houston. *An LQG Up-And-Away Flight Control Design For The STOL F-15 Aircraft.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1985.
10. Integrated Systems, Inc. *MATRIXx Reference Manual.* February 1988.
11. P. S. Maybeck. *Stochastic Models, Estimation and Control.* Volume 3, Academic Press, New York, 1982.
12. P. S. Maybeck. *Stochastic Models, Estimation and Control.* Volume 1, Academic Press, New York, 1979.
13. P. S. Maybeck, W. G. Miller, and J. M. Howey. Robustness Enhancement for LQG Digital Flight Controller design. *IEEE 1984 National Aerospace and Electronics Conference (NAECON 1984), Dayton, Ohio*, 518–525, May 1984.
14. W. G. Miller. *Robust Multivariable Controller Design Via Implicit Model-Following Methods.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1983.

15. K.N. Neumann. *A Digital Rate Controller For The Control Reconfigurable Combat Aircraft Designed Using Quantitative Feedback Theory.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.

16. D. L. Pogoda. *Multiple Model Adaptive Controller For The STOL F-15 With Sensor/Actuator Failures.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.

17. M.S. Sobota. *LQG/LTR Digital Control Law Design of a Robust Lateral Directional CGT/PI/KF Flight Controller for a STOL F-15 in a Landing Configuration.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1986.

18. Warren Weinstein, et.al. Control Reconfigurable Combat Aircraft Development Phase 1 - R and D Design Evaluation. May 1987. AFWAL-TR-87-3011, Wright Patterson AFB, OH.

# *Vita*

Captain Steven S. Payson attended the United States Air Force Academy from June 1980 to May 1984. He received a Bachelor of Science in Engineering. Captain Payson was then assigned to the Ballistic Missile Office at Norton AFB, where he was a Guidance and Control Project Officer working on the Small ICBM program. His present assignment is at the Air Force Institute of Technology in a master's program. Following graduation, Captain Payson will be reassigned to the Foreign Technology Division.

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>AFIT/GE/ENG/89M-6 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>School of Engineering<br>6b. OFFICE SYMBOL (If applicable)<br>AFIT/ENG | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS (City, State, and ZIP Code)<br>Air Force Institue of Technology<br>Wright-Patterson AFB Ohio 45433 | 7b. ADDRESS (City, State, and ZIP Code) |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>AFWAL Flight Dynamics Lab<br>8b. OFFICE SYMBOL (If applicable)<br>AFWAL/FDCLA | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (City, State, and ZIP Code)<br>Wright Patterson AFB Ohio 45433 | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. |

11. TITLE (Include Security Classification)
Flight Control System for the CRCA Using A Command Generator Tracker and Kalman Filter Volume I

12. PERSONAL AUTHOR(S)
Steven S. Payson, Captain, USAF

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Thesis | FROM _____ TO _____ | | |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | Command Generator Tracker, Kalman Filter, Computer Aided Design, Linear Quadratic Gaussian, Flight Control Systems |
| 01 | 04 | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
Thesis Advisor: Dr Peter S. Maybeck

| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | UNCLASSIFIED |

| 22a. NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE (Include Area Code) | 22c. OFFICE SYMBOL |
|---|---|---|
| Peter S. Maybeck, Professor | (513) 255-2057 | AFIT/EN |

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

This research develops an integrated software design package useful in the synthesis of CGT/PI/KF control systems, and uses this software package to design and evaluate a longitudinal flight control system for the Control Reconfigurable Combat Aircraft (CRCA). The software package, called CGTPIKF and built with MATRIXX commands, allows for the synthesis and evaluation of a Command Generator Tracker (CGT) which provides inputs to the system and acts as a precompensator, and a regulator with proportional plus integral (PI) feedback which forces the system outputs to mimic the model output. The software also allows the incorporation of a Kalman filter for estimation of the system states. Certainty equivalence can be invoked by adopting the LQG assumptions, thereby allowing the Kalman filter to be designed independently of the CGT/PI controller. The total CGT/PI/KF controller can then be evaluated and the design refined. CGTPIKF is an interactive, menu driven CAD package which can be used in the development of any CGT/PI/KF control system, regardless of application.

A flight control system was designed for the CRCA air combat mode (ACM) entry using CGTPIKF. This control system was designed to force the aircraft to emulate a first order response in pitch rate. The command model of the command generator tracker represented a first order pitch rate response with a rise time of .6 sec. Various weighting matrices were evaluated and refined in the development of the PI controller; the different controller designs were tested against the simulation containing various modelling errors, particularly failure conditions. The Kalman filter was later added, and the controller was again tested against the failure conditions. Loop Transmission Recovery (LTR) was successfully implemented to enhance robustness. The results confirm that a robust control system can be designed using the software package developed in this research.