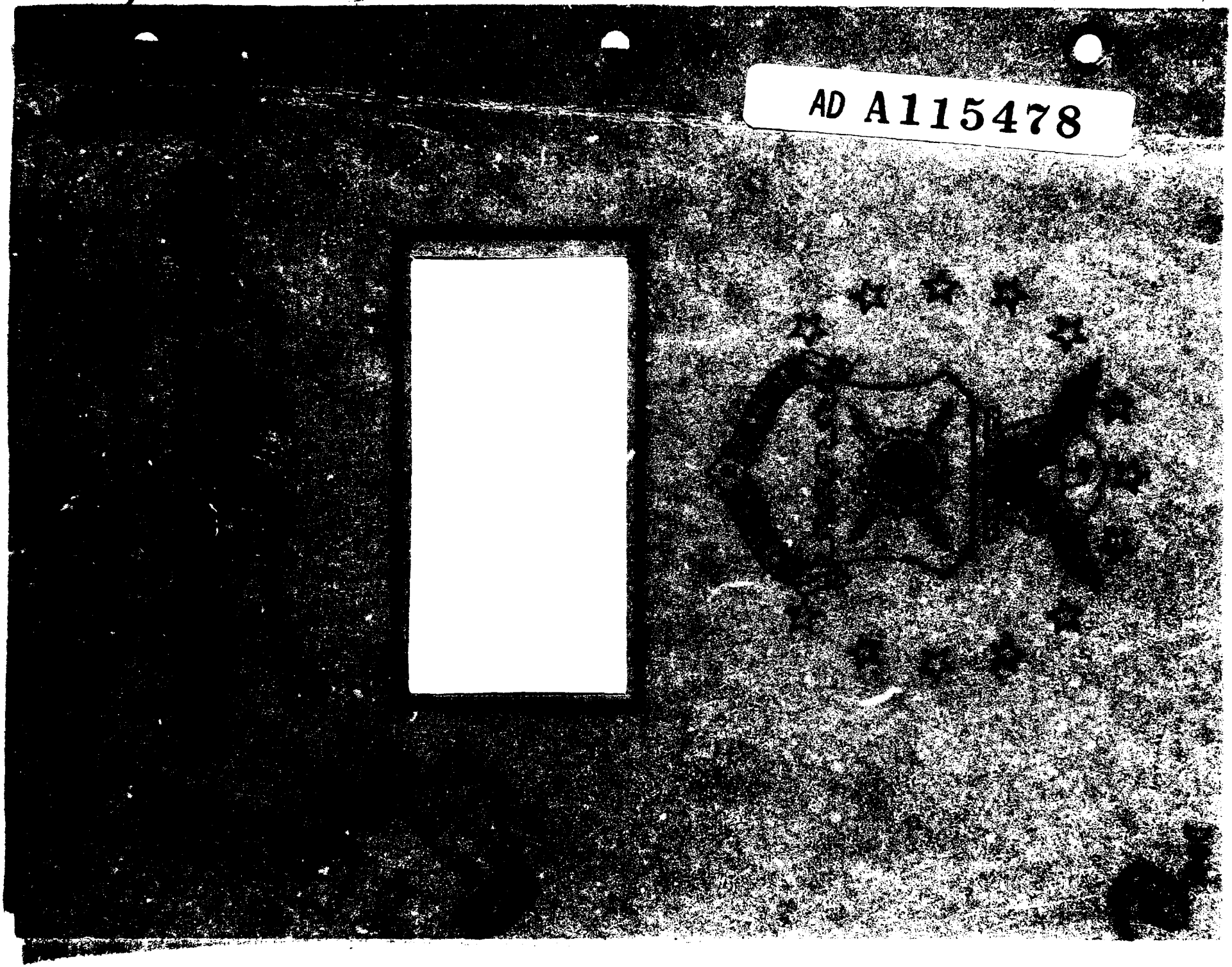
AD-A115 478 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 1/3
ROBUST CONTROL SYSTEMS.(U)
DEC 81 E D LLOYD
UNCLASSIFIED AFIT/GE/EE/81D-36 NL



AFIT/GE/EE/81D-36

ROBUST CONTROL SYSTEMS

THESIS

AFIT/GE/EE/81D-36 Eric D. Lloyd
Capt USAF

AFIT/GE/EE/81D-36

ROBUST CONTROL SYSTEMS

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

by

Eric D. Lloyd

Capt USAF

Graduate Electrical Engineering

December 1981

Preface

The Air Force (and the Department of Defense in general) is particularly interested (as evidenced by the fact that many of the references used in preparing this thesis were sponsored by Department of Defense agencies) in research in robust control systems design since the results are directly applicable to many of its sophisticated weapon systems. Several of the laboratories in the Air Force Systems Commands' Aeronautical Systems Division are helping to sponsor this Air Force Institute of Technology (AFIT) Masters thesis project. The primary motivation for this project is that many current control systems are and most future control systems will be, implemented in digital computers and, therefore, will be discrete-time controllers (Ref 7). Furthermore, if robust controllers can be used, there exists the possibility of reduced computational and hardware expense.

Thanks are due Professor Peter S. Maybeck for his invaluable guidance concerning the basic nature of this robust control system study as well as the final format of this thesis. I would also like to thank the other thesis committee members, Lt. Col. Carpinella and Capt. Silverthorn, for comments and guidance during the final preparation of this thesis. Special thanks are due to Sandra A., Todd Q., Weston S., and Jodi S. Lloyd, my family, for putting up with me during the sometimes frustrating but rewarding task of completing this thesis.

Contents

Contents

List of Figures

List of Figures

List of Figures

List of Tables

Abstract

The Doyle and Stein robustness enhancement technique for continuous-time LQG stochastic controllers was investigated in application to simple examples and a realistic Apollo Command Service Module/Lunar Module Thrust Vector Control System that exhibited severe robustness problems in its initial design. This technique was then extended to discrete-time systems in two ways. First, the continuous-time controller to which the Doyle and Stein technique had been applied was discretized using first order approximations. Second, an approximation to their continuous-time technique was developed for sampled-data control systems. In addition, an attempt was made to enhance the robustness of sampled-data systems by directly picking the gain of the Kalman filter within the controller structure based on an approach similar to that of Doyle and Stein.

Sampled-data controllers were designed using each of these approaches. The resulting performance analysis for each closed-loop system was based on the time histories of the mean and covariance of the "truth model" states and controls as well as on the eigenvalues of the closed-loop system. In both the discretized continuous-time and sampled-data cases, significant steady-state robustness enhancement was observed. Results for picking the Kalman filter gain directly were inconclusive. General purpose interactive software for developing robustified LQG controllers was also produced and documented.

I Introduction

The purpose of this thesis is to demonstrate a systematic procedure to design computationally efficient, discrete-time control system algorithms that will perform adequately (i.e., at least maintain closed-loop system stability) when uncertain parameters in the system design models vary significantly. Such a control algorithm is said to have stability robustness or more simply is said to be "robust". This introduction provides a background for this study, a summary of recent efforts in the design of robust control systems, and a discussion of the approach taken in this thesis. Following this, there is a brief discussion of the notation used in the remainder of this thesis.

Background

Stability robustness is a concern in control systems since it determines if control systems will operate in a stable fashion even though certain design parameters may change from the nominal values used in the design of the control system. One reason parameter changes may occur is that a systems' physical operating characteristics may change with environmental conditions. For example, aircraft control systems are designed to operate at or near certain flight conditions in the flight envelope and must be adjusted when the operating point changes. Parameter changes may also occur

because they are not known exactly at the time of the controller design and/or because during system operation physical components may fail or may degrade with age or environmental conditions (Refs 6 and 10).. For instance, in designing controllers for wing flutter suppression in aircraft and thrust vector control of missles and spacecraft, the bending mode description of these flexible vehicles can not be specified exactly. Thus, when a controller is designed based on the nominal description of these modes, the actual closed-loop system may perform inadequately or become unstable if the true values are different from the nominal ones. In addition, characterization of the bending modes may change as a result of changing loads such as when the fuel supply decreases. Two additional areas of concern that potentially affect the stability robustness of control systems are sensor failures and computer wordlength. Systems can be designed so that a certain number of sensor failures can be tolerated without causing unstable control system operation. Another equally important consideration, computer wordlength, affects robustness in at least two ways. First, if a control system is implemented using a computer program, finite computer wordlength affects the accuracy of any calculations and, subsequently, the stability. Second, even if the program results in a stable closed-loop system on one computer, there are no guarantees that the program will result in a stable closed-loop system if a different computer with a different

wordlength is used (Ref 1). Ackerman (Ref 1) and Maybeck (Ref 10) discuss still more areas that may affect robustness, but it is more important at this point to discuss robustness itself and to consider why robustness is an important issue.

Robustness. An automatic control system that exhibits the property of stability robustness is one in which the closed-loop system will remain stable should certain system design parameters change from the design values. More precisely, robustness specifies the finite regions of the design model around a nominal model in which stable control system operation is preserved. Although some papers (Refs 6 and 13) deal with robustness only in regard to parameter variations within the basic controlled system, robustness actually encompasses all possible variations in design models that can affect control system stability (Ref 10). For a detailed rigorous discussion of robustness, see Maybeck (Ref 10).

Importance of Robustness. There are several reasons why robustness is an important control system property. One reason is that the models used in the control system design are just that, models! Subsequently, no matter how much effort is put into defining the system model there will always be variations between the model and the physical system it represents (Ref 10). In addition to not having perfect models, the physical components of a system tend to degrade with age or environmental conditions (Ref 6). For either of these two

reasons, a control system must have robustness if it ever is to attain stable operation.

By defining robustness properties with respect to various areas of concern, systems or portions of systems that require additional or different stabilizing efforts can be pinpointed. For example, certain portions of a control system might be implemented using adaptive control techniques when large uncertainties in design parameters exist. Actually, adaptive control techniques could possibly handle most systems with uncertain parameters. But, since adaptive control is comparatively expensive, a system's robustness can be used to indicate when the additional expense is warranted. It should be pointed out that robust designs generally have some performance degradation when compared to adaptive designs (Ref 11). Furthermore, robustness studies can be used to determine how much of critical control system components (i.e., actuators, sensors) such as those onboard aircraft or spacecraft, should be implemented in quadruplex redundancy to guarantee reliability and stability. The need for expensive quadruplex redundancy may in some cases be reduced by using robust control system designs. For example, robust automatic flight control systems that result in a stable closed-loop system even though some actuators and/or sensors fail are much less expensive than control systems that require quadruplex redundancy (Ref 1).

Recent Efforts in Robust Control System Design

Robustness is the subject of several recent articles in control systems literature. Safonov (Ref 15), for instance, in a paper presented at the 1979 IEEE Conference on Decision and Control, proves a theorem based on L_2 conic-sector techniques, that leads to a precise quantitative characterization of feedback sensitivity to large-but-bounded frequency-dependent plant variations. He points out that an interesting implication of the theorem is that there exists a fundamental limit on the amount by which output feedback can reduce a given plant's sensitivity to frequency-dependent plant variations. In an earlier work, Safonov and Athans (Ref 14) discuss the robustness properties of a restricted class of controllers with respect to large plant parameter variations. Specifically, they suggest that linear-quadratic-Gaussian, controllers have the desirable robustness properties of full state feedback controllers (i.e., guaranteed classical gain margins of -6dB to +00dB and phase margins of +60° on all channels even when implemented using a Kalman filter for a plant state estimator.

Doyle (Ref 3), however, shows by a simple counter-example that the results claimed by Safonov and Athans do not hold in general for the LQ controller-Kalman filter combination. Since then, Doyle and Stein (Ref 2) developed a technique that recovers the desirable robustness properties of a full state feedback controller that uses a standard LQG controller in which the Kalman filter gains are adjusted in a particular

fashion (to be discussed later). In addition to demonstrating their technique, they also show that other frequently mentioned techniques to recover robustness do "not work in general" unless the techniques drive *some* observer poles toward stable plant zeros and the others toward infinity as their technique does.

Approach

This study will be concerned with extending a particular technique for designing robust continuous-time controllers to discrete-time controllers, since the current trends indicate most future control systems will be implemented in digital computers. The technique that will be the basis of this study is proposed by J.C. Doyle and G. Stein (Ref 2). Their technique is directly applicable to the design of the robust continuous-time Linear Quadratic Gaussian (LQG) controllers with uncertain parameters embedded in the system model. The basic idea of their technique is to add pseudonoise at the control points of entry (See the Enhancing Robustness in Continuous-time Systems section of Chapter II for a discussion of how this is accomplished). Note that their technique is restricted to linear plants that are both observable and controllable, have the same number of inputs as outputs, and have no transmission zeros in the right half of the s-plane.

In this thesis a relatively simple known system model with a single uncertain parameter is used as the basis for design of robust controllers. For this model several different controllers are developed. First a continuous-time LQG controller is developed. Next, several different approaches are taken to adapt Doyle and Stein's procedure to discrete-time LQG controllers. After this, a procedure described by Maybeck (Ref 10) for designing robust sampled-data controllers is used. In all cases above, the performance is analyzed using a covariance analysis. The development of all the controllers and the performance analysis algorithms is discussed in detail in Chapter II. The results and conclusions are discussed in Chapter III.

One of the principle by-products of this thesis is the general-purpose user-oriented interactive computer program that has been developed. The program mechanizes the formation of and the performance analysis of robust LQG controllers. Appendices A and B describe the program, Appendix C discusses some of the considerations that were involved in the programming. Appendix D contains the software verification description and Appendix E is a users manual for the program.

Notation

Before leaving this introduction it is necessary to introduce some of the notation used in the following sections of this thesis. Random variables are indicated by an under

tilde, i.e., $\underset{\sim}{x}$ is the notation for a random variable x. If x in this case is also a vector, it will also be underlined, i.e., $\underline{x}$. All matrices are capitalized to distinguish them from vectors and underlined unless they represent a one-dimensional square matrix. All other notational devices will be introduced as they are needed. Additionally, the symbol "$\underline{\underline{\Delta}}$" is read as "defined as".

II Robust LQG Controllers

Introduction.

The purpose of this section is to discuss the approach taken in this thesis toward designing robust Linear Quadratic Gaussian (LQG) controllers. In particular, robustness with respect to uncertain parameters embedded in the system model is the primary concern of this study. Starting with a relatively simple known system model with a single uncertain parameter, Doyle and Stein's (Ref 2) technique for designing robust continuous-time LQG controllers is applied and the performance evaluated. Next, several different approaches are taken to try to adapt Doyle and Stein's procedure to discrete-time LQG controllers. In addition to discussing the different controllers developed in this section, the software used to implement the design and performance analysis is also discussed.

There are seven major subsections in this chapter. First, the continuous-time LQG controller and performance analysis is introduced. Next, the Doyle and Stein technique for enhancing robustness in continuous-time controllers is discussed. Then the model to be used in this study is introduced. Following this, the sampled-data LQG controller and performance analysis are introduced. Next, the three different approaches to extending Doyle and Stein's technique to discrete-time LQG controllers are presented. The first involves simply discretizing the continuous-time controller

after the Doyle and Stein technique is applied. The second is a sampled-data controller for the given model in which $\underline{Q}_d = \underline{Q}_{cont}\,\Delta t$, where $\underline{Q}_d$ is the strength of the assumed discrete-time dynamics noise input and $\underline{Q}_{cont}$ is the strength of the assumed continuous-time dynamics noise input from the Doyle and Stein. The third approach involves directly picking the Kalman filter gain $\underline{K}$ to achieve robustness without solving a Riccati equation so as to attain the desired $\underline{K}$.

Continuous-Time LQG Controller

The following development of the continuous-time LQG controller is based on Maybeck (Ref 10).

The LQG controller shown in Fig 2.1 is an optimal controller in the sense that it minimizes the cost function J_c:

$$J_c = E\left\{ \tfrac{1}{2}\,\underset{\sim}{x}^T(t_f)\,\underline{X}_f\,\underset{\sim}{x}(t_f) + \int_{t_o}^{t_f} \tfrac{1}{2}\left(\begin{bmatrix} \underset{\sim}{x}(t) \\ \underset{\sim}{u}(t) \end{bmatrix}^T \begin{bmatrix} \underline{W}_{xx}(t) & \underline{W}_{xu}(t) \\ \underline{W}_{ux}(t) & \underline{W}_{uu}(t) \end{bmatrix} \begin{bmatrix} \underset{\sim}{x}(t) \\ \underset{\sim}{u}(t) \end{bmatrix} \right) dt \right\} \quad (2.1)$$

where $\underset{\sim}{x}(t)$ represents a system state at time t, $\underset{\sim}{u}(t)$ represents a set of controls applied at time t, $\underline{X}_f$ is the cost-weighting matrix for the final state, $\underline{W}_{xx}(t)$ is the cost-weighting matrix associated with all the states at time t, $\underline{W}_{uu}(t)$ is the cost-weighting matrix associated with applying control inputs at time t, and $\underline{W}_{xu}$ and $\underline{W}_{ux}$ are cross terms relating cost for specific states and controls combinations. Note that $\underline{W}_{xu} = \underline{W}_{ux}^T$. Note also that E is the expected value

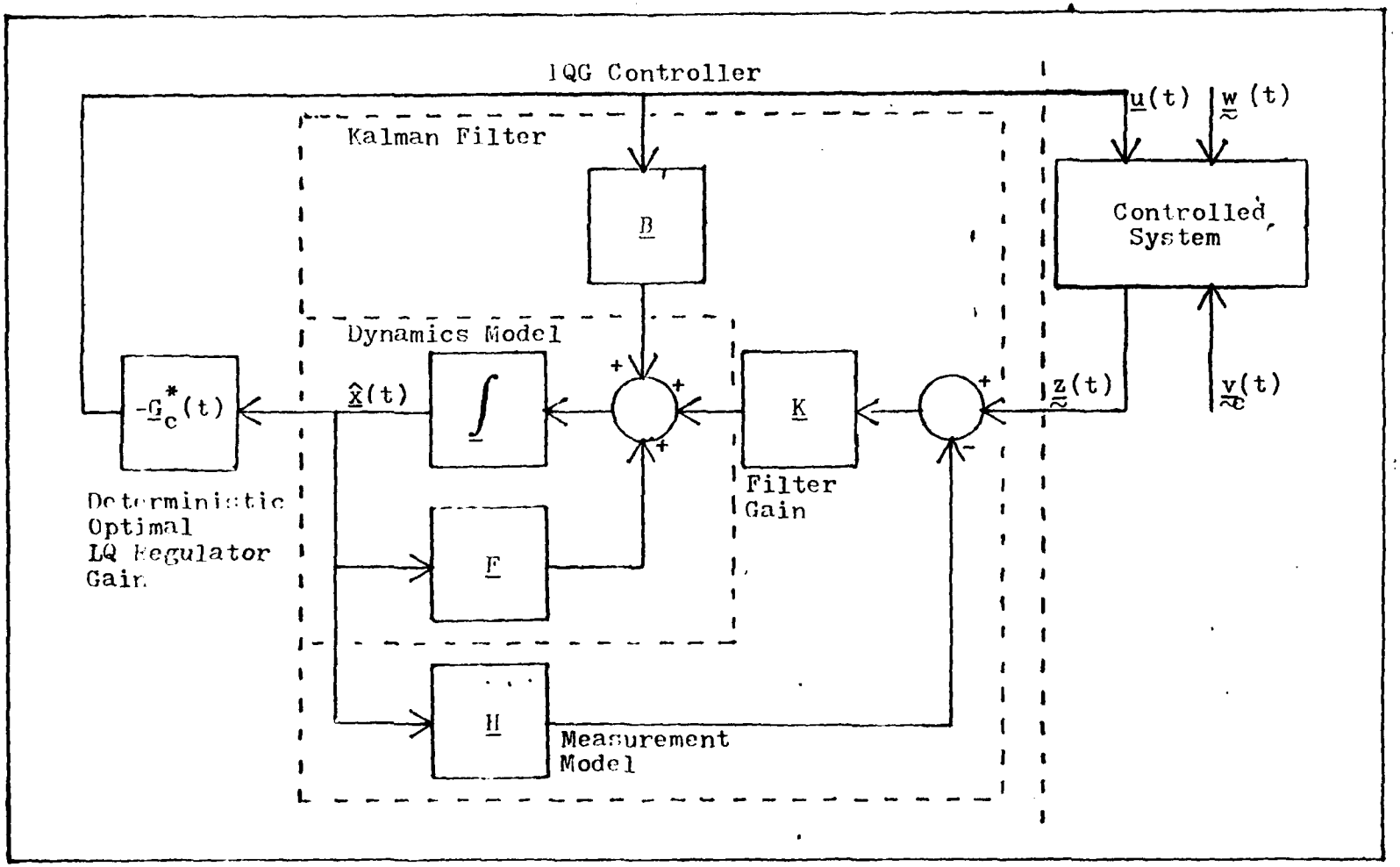


Fig 2.1 Continuous-time LQG Controller (Ref 10)

operator.

For a physical system as in Fig 2.1, the state of the system at time t is described by

$$\dot{\underset{\sim}{x}}(t) = \underline{F}(t)\ \underset{\sim}{x}(t) + \underline{B}(t)\ \underset{\sim}{u}(t) + \underline{G}(t)\ \underset{\sim}{w}(t) \tag{2.2}$$

where $\underset{\sim}{w}(t)$ is a zero mean white Gaussian noise output of strength $\underline{Q}(t)$. That is

$$E\left[\underset{\sim}{w}(t)\ \underset{\sim}{w}^T(t+\tau)\right] = \underline{Q}(t)\ \delta(\tau) \tag{2.3}$$

A Kalman filter is used to estimate the mean of $\underset{\sim}{x}(t)$, conditioned on measurements of the form

$$\underset{\sim}{z}(t) = \underline{H}(t)\ \underset{\sim}{x}(t) + \underset{\sim}{v}_c(t) \tag{2.4}$$

$\underline{R}_c(t)$ is the strength of the zero-mean white Gaussian measurement noise $\underset{\sim}{v}_c(t)$ and is

$$E\left[\underset{\sim}{v}(t)\ \underset{\sim}{v}^T(t+\tau)\right] = \underline{R}_c(t)\ \delta(\tau) \tag{2.5}$$

The estimate is denoted by $\hat{\underline{x}}(t)$ and is described by the following relationships:

$$\dot{\hat{\underline{x}}}(t) = \underline{F}(t)\ \hat{\underline{x}}(t) + \underline{B}(t)\ \underline{u}(t) + \underline{K}(t)\left[\underline{z}(t) - \underline{H}(t)\ \hat{\underline{x}}(t)\right] \tag{2.6}$$

$$\underline{K}(t) = \underline{P}(t)\ \underline{H}^T(t)\ \underline{R}_c^{-1}(t) \tag{2.7}$$

$\underline{P}$ in Eq (2.7) is the associated error covariance and is the solution to the foward Riccati equation

$$\dot{\underline{P}}(t) = \underline{F}(t)\ \underline{P}(t) + \underline{P}(t)\ \underline{F}^T(t) + \underline{G}(t)\ \underline{Q}(t)\ \underline{G}^T(t) - \underline{P}(t)\ \underline{H}^T(t)\ \underline{R}_c^{-1}(t)\ \underline{H}(t)\ \underline{P}(t) \tag{2.8}$$

$\hat{\underline{x}}_o$ and $\underline{P}_o$ are the initial conditions of differential equations given in Eqs (2.6) and (2.8) respectively, where these are the defining parameters of an a priori Gaussian density function for $\underline{x}(t_o)$.

The deterministic controller to be cascaded with the Kalman filter to form the LQG controller is described by the following equations:

$$\underline{u}^*(t) = -\underline{G}_c^*(t)\, \underline{x}(t) \tag{2.9}$$

$$\underline{G}_c^*(t) = \underline{W}_{uu}^{-1}(t)\, \underline{B}^T(t)\, \underline{K}_c(t) \tag{2.10}$$

where $\underline{u}^*(t)$ is the optimal control to be applied, $\underline{G}_c^*(t)$ is the optimal controller gain matrix and $\underline{K}_c(t)$ is the solution to the backward Riccati equation with $\underline{W}_{xu} = \underline{O}$.

$$-\dot{\underline{K}}_c(t) = \underline{F}^T(t)\, \underline{K}_c(t) + \underline{K}_c(t)\, \underline{F}(t) + \underline{W}_{xx}(t) - \underline{K}_c(t)\, \underline{B}(t)\, \underline{W}_{uu}^{-1}(t)\, \underline{B}^T(t)\, \underline{K}_c(t) \tag{2.11}$$

$$\underline{K}_c(t_f) = \underline{X}_f$$

(For the case when $\underline{W}_{xu} \neq \underline{O}$, see the discussion in Appendix C.) Note that the certainty equivalence principle applies to Eq (2.9) so that $\underline{x}(t)$ can be replaced by $\hat{\underline{x}}(t)$ in that equation when measurements given by (2.4) replace perfect knowledge of $\underline{x}(t)$ (Ref 10).

Since there are numerical complexities in handling the time varying LQG controller and since these can ofter be neglected in actual implementation, a constant-gain time

invariant solution with stationary noise inputs will be used. That is, $\underline{F}$, $\underline{B}$, $\underline{G}$, $\underline{H}$, $\underline{Q}$ and $\underline{R}_c$ will be constant and the initial filter transients and final deterministic controller transients will be ignored during the design of the controller. Therefore, in this case, the steady state error covariance $\bar{\underline{P}}$ will be used in place of $\underline{P}(t)$ and steady state $\bar{\underline{K}}_c$ will be used instead of $\underline{K}_c(t)$. $\bar{\underline{P}}$ and $\bar{\underline{K}}_c$ are given by (Ref 10)

$$\dot{\bar{\underline{P}}} = \underline{0} = \underline{F}\,\bar{\underline{P}} + \bar{\underline{P}}\,\underline{F}^T + \underline{G}\,\underline{Q}\,\underline{G}^T - \bar{\underline{P}}\,\underline{H}^T\,\underline{R}_c^{-1}\,\underline{H}\,\bar{\underline{P}} \qquad (2.13)$$

$$\dot{\bar{\underline{K}}}_c = \underline{0} = \underline{F}^T\,\bar{\underline{K}}_c + \bar{\underline{K}}_c\,\underline{F} + \underline{W}_{xx} - \bar{\underline{K}}_c\,\underline{B}\,\underline{W}_{uu}^{-1}\,\underline{B}^T\,\bar{\underline{K}}_c \qquad (2.14)$$

Two software routines were written specifically to handle the Kalman filter and the deterministic optimal controller. The flowcharts and source code listings are in Appendices A and B respectively. Note that many subroutines called in the software package come from a set of routines generated by Kleinman (Ref 5) and modified by Floyd (Ref 4).

Continuous-Time Performance Analysis

Since the control systems designed in this study are stochastic regulators, the time histories of the mean and covariance of the truth model states $\underline{x}_t(t)$ and the generated controls $\underline{u}(t)$ are used as the basis of performance analysis. In the test setup of Fig 2.2, the robustness of each controller design to plant parameter variations is evaluated by comparing the mean and covariance time histories when plant parameter values in the truth model are varied from those in the controller design model. The truth model in Fig 2.2 represents the most

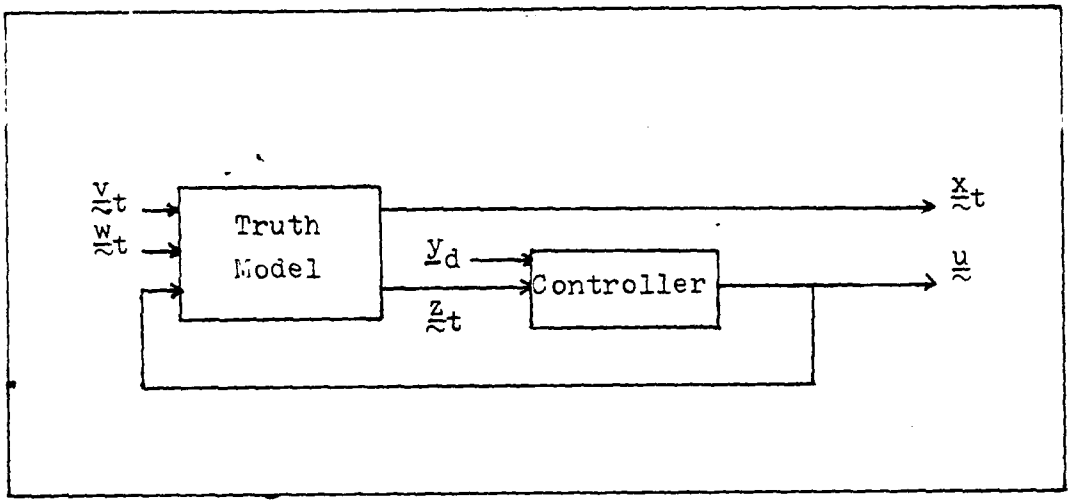


Fig 2.2 Performance Evaluation

complete and accurate mathematical model available to describe the physical system to be controlled. This is in contrast to the model upon which the controller is based, which is usually a mathematical model of much lower dimension so that it can be readily implemented in an online controller. Note that if the system models and/or controllers are nonlinear, a Monte Carlo simulation analysis would be required instead. Note also that a time history of the quadratic cost function J_c of Eq (2.1) is of little use since it gives no information as to individual channel costs (Ref 10).

This test setup is described in detail for the discrete-time case in Maybeck (Ref 10). The following continuous-time performance analysis closely follows that discrete-time development where possible. The following subscripts will be used throughout this development:

c= controller model
t= truth model
a= augmented model

cx= state controller gain
cy= input controller gain
cz= measurement controller gain

The truth model dynamics are given by

$$\dot{\underset{\sim}{x}}_t = \underline{F}_t(t)\ \underset{\sim}{x}_t(t) + \underline{B}_t(t)\ \underset{\sim}{u}(t) + \underline{G}_t(t)\ \underset{\sim}{w}_t(t) \qquad (2.15)$$

The measurements available to the controller are

$$\underset{\sim}{z}_t(t) = \underline{H}_t(t)\ \underset{\sim}{x}_t(t) + \underset{\sim}{v}_t(t) \qquad (2.16)$$

The initial conditions and strengths of the noises in these two equations are:

$$E\left|\underset{\sim}{w}_t(t)\right| = \underline{0} \qquad (2.17)$$

$$E\left|\underset{\sim}{w}_t(t)\ \underset{\sim}{w}_t^T(t+\tau)\right| = \underline{Q}_t\ \delta(\tau) \qquad (2.18)$$

$$E\left|\underset{\sim}{v}_t(0)\right| = \underline{0} \qquad (2.19)$$

$$E\left|\underset{\sim}{v}_t(t)\ \underset{\sim}{v}_t^T(t+\tau)\right| = \underline{R}_t\ \delta(\tau) \qquad (2.20)$$

$$E\left|\underset{\sim}{x}_t(0)\right| = \bar{\underset{\sim}{x}}_{t_o} \qquad (2.21)$$

$$E\left|\left[\underset{\sim}{x}_t(0) - \bar{\underset{\sim}{x}}_{t_o}\right]\left[\underset{\sim}{x}_t(0) - \bar{\underset{\sim}{x}}_{t_o}\right]^T\right| = \underline{P}_{t_o} \qquad (2.22)$$

In general, the control input $\underline{u}(t)$ and the controller states will be a function of the measurements, $\underline{z}_t$, the controller states, $\underline{x}_c$, and the desired inputs, $\underline{y}_d$. It can thus be written as (Ref 10)

$$\underline{u}(t) = \underline{G}_{cx}(t)\ \underline{x}_c(t) + \underline{G}_{cz}(t)\ \underline{z}_t(t) + \underline{G}_{cy}(t)\ \underline{y}_d(t) \qquad (2.23A)$$

$$\dot{\underset{\sim}{x}}_c(t) = \underset{\sim}{F}_c(t)\ \underset{\sim}{x}_c(t) + \underset{\sim}{B}_{cy}(t)\ \underset{\sim}{y}_d(t) + \underset{\sim}{B}_{cz}(t)\ \underset{\sim}{z}_t(t) \qquad (2.23B)$$

Note that in general, $\underset{\sim}{y}_d$ is not zero but that in the case of the LQG regulators used in this thesis, $\underset{\sim}{y}_d$ is zero.

As stated earlier, the performance analysis provides time histories of the mean and covariance of

$$\underset{\sim}{y}_a(t) = \begin{bmatrix} \underset{\sim}{x}_t(t) \\ \underset{\sim}{u}(t) \end{bmatrix} \qquad (2\text{-}24)$$

For $\underset{\sim}{y}_a(t)$, the mean is $\underset{\sim}{m}_{ya}(t)$, the covariance is $\underset{\sim}{P}_{ya}(t)$ and the autocorrelation is $\underset{\sim}{\Psi}_{ya}(t)$ (which is simply $\underset{\sim}{P}_{ya}(t) + \underset{\sim}{m}_{ya}(t)\ \underset{\sim}{m}^T_{ya}(t)$).

As in Maybeck (Ref 10), let the cost be described as

$$\frac{dJ_c}{dt} = E\left\{ \tfrac{1}{2} \sum_{k=1}^{m} w_k\ q_k^2 \right\} \qquad (2.25)$$

where $q_1 \ldots\ldots q_m$ are the scalar quantities of interest and are linear combinations of $\underset{\sim}{y}_a$ given by $q_k = \underset{\sim}{q}_k^T\ \underset{\sim}{y}_a$. For $\underset{\sim}{W}_{ya} = \sum_{k=1}^{m} w_k\ \underset{\sim}{q}_k\ \underset{\sim}{q}_k^T$ then

$$\frac{dJ_c}{dt} = E\left| \tfrac{1}{2}\ \underset{\sim}{y}_a^T\ \underset{\sim}{W}_{ya}\ \underset{\sim}{y}_a \right| = \tfrac{1}{2}\ tr\ \left| \underset{\sim}{W}_{ya}\ \underset{\sim}{\Psi}_{ya} \right| \qquad (2.26)$$

From these relationships, it can be seen that $\underset{\sim}{\Psi}_{ya}$, and thus $\underset{\sim}{m}_{ya}$ and $\underset{\sim}{P}_{ya}$ as generated in the performance analysis, will be of importance in producing J_c if desired (Ref 9).

Now, to characterize the statistics of $\underset{\sim}{y}_a$, the statistics of the inernal process $\underset{\sim}{x}_a$ must be characterized where

$$\underset{\sim}{x}_a = \begin{bmatrix} \underset{\sim}{x}_t \\ \underset{\sim}{x}_c \end{bmatrix} \tag{2.27}$$

The first step is to eliminate $\underset{\sim}{u}$ and $\underset{\sim}{z}_t$ from the equations for $\underset{\sim}{x}_t$ and $\underset{\sim}{x}_c$. Note, time arguments will be removed for compactness wherever it creates no ambiguities. Equation (2.15) becomes

$$\begin{aligned} \dot{\underset{\sim}{x}}_t &= \underline{F}_t \underset{\sim}{x}_t + \underline{B}_t \left(\underline{G}_{cx} \underset{\sim}{x}_c + \underline{G}_{cy} \underline{y}_d + \underline{G}_{cz} \left(\underline{H}_t \underset{\sim}{x}_t + \underset{\sim}{v}_t \right) \right) \\ &\quad + \underline{G}_t \underset{\sim}{w}_t \\ &= \left(\underline{F}_t + \underline{B}_t \underline{G}_{cz} \underline{H}_t \right) \underset{\sim}{x}_t + \underline{B}_t \underline{G}_{cx} \underset{\sim}{x}_c + \underline{B}_t \underline{G}_{cy} \underline{y}_d \\ &\quad + \underline{B}_t \underline{G}_{cz} \underset{\sim}{v}_t + \underline{G}_t \underset{\sim}{w}_t \end{aligned} \tag{2.28}$$

Eq (2.23B) becomes

$$\dot{\underset{\sim}{x}}_c = \underline{F}_c \underset{\sim}{x}_c + \underline{B}_{cy} \underline{y}_d + \underline{B}_{cz} \left(\underline{H}_t \underset{\sim}{x}_t + \underset{\sim}{v}_t \right) \tag{2.29}$$

Letting

$$\underset{\sim}{w}_a = \begin{bmatrix} \underset{\sim}{w}_t \\ \underset{\sim}{v}_t \end{bmatrix} \tag{2.30}$$

then

$$\underline{Q}_a = \begin{bmatrix} \underline{Q}_t & \underline{0} \\ \underline{0} & \underline{R}_t \end{bmatrix} \tag{2.31}$$

and an augmented system can be formed such that

$$\dot{\underset{\sim}{x}}_a = \underline{F}_a \underset{\sim}{x}_a + \underline{B}_a \underline{y}_d + \underline{G}_a \underset{\sim}{w}_a \tag{2.32}$$

where

$$\underline{F}_a = \begin{bmatrix} \underline{F}_t + \underline{B}_t \underline{G}_{cz} \underline{H}_t & \underline{B}_t \underline{G}_{cx} \\ \underline{B}_{cz} \underline{H}_t & \underline{F}_c \end{bmatrix} \tag{2.33}$$

$$\underline{B}_a = \begin{bmatrix} \underline{B}_t \underline{G}_{cy} \\ \underline{B}_{cy} \end{bmatrix} \tag{2.34}$$

$$\underline{G}_a = \begin{bmatrix} \underline{G}_t & \underline{B}_t \underline{G}_{cz} \\ \underline{0} & \underline{B}_{cz} \end{bmatrix} \tag{2.35}$$

The initial conditions for $\underset{\sim}{x}_a$ are

$$E\left\{ \underset{\sim}{x}_a(0) \right\} = E \begin{Bmatrix} \underset{\sim}{x}_t(0) \\ \underset{\sim}{x}_c(0) \end{Bmatrix} = \bar{\underline{x}}_{a_o} \tag{2.36}$$

$$E\left\{ \left[\underset{\sim}{x}_a(t_o) - \bar{\underline{x}}_{a_o} \right] \left[\underset{\sim}{x}_a(t_o) - \bar{\underline{x}}_{a_o} \right]^T \right\} = \begin{bmatrix} \underline{P}_{t_o} & \underline{0} \\ \underline{0} & \underline{0} \end{bmatrix} \tag{2.37}$$

The mean covariance and autocorrelation of $\underset{\sim}{x}_a$ are

$$\underline{m}_{x_a} = E\left\{ \underset{\sim}{x}_a \right\} \tag{2.38}$$

$$\underline{\Psi}_{x_a x_a} = E\left\{ \underset{\sim}{x}_a \underset{\sim}{x}_a^T \right\} = \underline{m}_{x_a} \underline{m}_{x_a}^T + \underline{P}_{x_a x_a} \tag{2.39}$$

$$\underline{P}_{x_a x_a} = E\left\{ (\underset{\sim}{x}_a - \underline{m}_{x_a}) (\underset{\sim}{x}_a - \underline{m}_{x_a})^T \right\} \tag{2.40}$$

The time propagation equations of the mean and covariance are

$$\dot{\underline{m}}_{x_a} = \underline{F}_a \underline{m}_{x_a} + \underline{B}_a \underline{V}_d \tag{2.41A}$$

$$\dot{\underline{P}}_{x_a x_a} = \underline{F}_a \underline{P}_{x_a x_a} + \underline{P}_{x_a x_a} \underline{F}_a^T + \underline{G}_a \underline{Q}_a \underline{G}_a^T \tag{2.42A}$$

or alternately

$$\underline{m}_{x_a}(t) = \underline{\Phi}_a(t, t_o)\, \underline{m}_{x_a}(t_o) + \int_{t_o}^{t} \underline{\Phi}_a(t, \tau)\, \underline{B}_a(\tau)\, \underline{y}_d(\tau)\, d\tau \qquad (2.41B)$$

$$\underline{P}_{x_a x_a}(t) = \underline{\Phi}_a(t, t_o)\, \underline{P}_{x_a x_a}(t_o)\, \underline{\Phi}_a^T(t, t_o) + \int_{t_o}^{t} \underline{\Phi}_a(t, \tau)\, \underline{G}_a\, \underline{Q}_a\, \underline{G}_a^T\, \underline{\Phi}_a^T(t, \tau)\, d\tau \qquad (2.42B)$$

where $\underline{\Phi}_a(t, t_o)$ is the state transition matrix associated with $\underline{F}_a$ as given in Eq (2.33). This form is more straightforward for computer implementation when time-invariant systems and controllers are used, since the integration involved in computing $\underline{\Phi}_a(t, t_o)$ need only be accomplished once. At this point in Maybeck's (Ref10) discrete-time performance analysis, he presents the means of expressing the cost equation in terms of the augmented vectors. A similar derivation is not done here since, as was mentioned earlier, computing the value of the cost function J_c is rarely of practical interest.

Recalling that the statistics of $\underset{\sim}{y}_a$ is of particular interest, $\underset{\sim}{y}_a$ can now be related to $\underset{\sim}{x}_a$ via

$$\underset{\sim}{y}_a = \begin{bmatrix} \underset{\sim}{x}t \\ \underset{\sim}{u} \end{bmatrix} = \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{G}_{cz}\ \underline{H}_t & \underline{G}_{cx} \end{bmatrix} \begin{bmatrix} \underset{\sim}{x}t \\ \underset{\sim}{x}c \end{bmatrix} + \begin{bmatrix} \underline{0} \\ \underline{G}_{cy} \end{bmatrix} \underline{y}_d + \begin{bmatrix} \underline{0} \\ \underline{G}_{cz} \end{bmatrix} \underset{\sim}{v}t \qquad (2.43)$$

where

$$\underset{\sim}{x}_a = \begin{bmatrix} \underset{\sim}{x}_t \\ \underset{\sim}{x}_c \end{bmatrix} \tag{2.44}$$

and where $\underline{z}$ has been eliminated from $\underline{u}$.

Since $\underset{\sim}{y}_a$ is a linear combination of variables with known statistics, that is $\underset{\sim}{x}_a$, $\underset{\sim}{v}_t$ and $\underline{y}_d$, its mean and covariance can be written as (Ref 9)

$$\underline{m}_{y_a} = \begin{bmatrix} \underline{m}_{x_t} \\ \underline{m}_u \end{bmatrix} = \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{G}_{cz}\ \underline{H}_t & \underline{G}_{cy} \end{bmatrix} \underline{m}_{x_a} + \begin{bmatrix} \underline{0} \\ \underline{G}_{cy} \end{bmatrix} \underline{y}_d \tag{2.45}$$

$$\underline{P}_{y_a} = \begin{bmatrix} \underline{P}_{x_t x_t} & \underline{P}_{x_t u} \\ \underline{P}^T_{x_t u} & \underline{P}_{uu} \end{bmatrix}$$

$$= \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{G}_{cz}\ \underline{H}_t & \underline{G}_{cx} \end{bmatrix} \underline{P}_{x_a x_a} \begin{bmatrix} \underline{I} & \underline{H}^T_t\ \underline{G}^T_{cz} \\ \underline{0} & \underline{G}^T_{cx} \end{bmatrix}$$

$$+ \begin{bmatrix} \underline{0} \\ \underline{G}_{cz} \end{bmatrix} \underline{R}_t \begin{bmatrix} \underline{0} & \underline{G}^T_{cz} \end{bmatrix} + \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{G}_{cz}\ \underline{H}_t & \underline{G}_{cx} \end{bmatrix} \underline{P}_{x_a v_t}$$

$$\begin{bmatrix} \underline{0} & \underline{G}^T_{cz} \end{bmatrix} + \begin{bmatrix} \underline{0} \\ \underline{G}_{cz} \end{bmatrix} \underline{P}^T_{x_a v_t} \begin{bmatrix} \underline{I} & \underline{H}^T_t\ \underline{G}^T_{cz} \\ \underline{0} & \underline{G}^T_{cx} \end{bmatrix} \tag{2.46}$$

It is obviously necessary to calculate a value for $\underline{P}_{x_a v_t}$ in order to use Eq (2.46). By definition, $\underline{P}_{x_a v_t}$ is

$$\underline{P}_{x_a v_t} = E\left\{ \left[\underset{\sim}{x}_a(t) - \underline{m}_{x_a}(0) \right] \left[\underset{\sim}{v}_t(t) - \underline{m}_{v_t}(0) \right]^T \right\} \tag{2.47}$$

Recalling that $\underline{m}_{v_t} = \underline{0}$, Eq (2.47) can be rewritten as

$$\underline{P}_{x_a v_t} = E\left| \underset{\sim}{x}_a(t)\, \underset{\sim}{v}_t^T(t) \right| - \underline{m}_{x_a}(t)\, \underset{\sim}{v}_t^T(t)\Big|$$

$$= E\left| \underset{\sim}{x}_a(t)\, \underset{\sim}{v}_t^T(t)\right| - E\left| \underline{m}_{x_a}(0)\, \underset{\sim}{v}_t^T(t)\right|$$

$$= E\left| \underset{\sim}{x}_a(t)\, \underset{\sim}{v}_t^T(t)\right| - \underline{m}_{x_a}(0)\, E\left| \underset{\sim}{v}_t^T(t)\right| \qquad (2.48)$$

Note that the expected value in the second term of Eq (2.48) is simply the mean of $\underset{\sim}{v}_t^T$, which is zero so that

$$\underline{P}_{x_a v_t} = E\left| \underset{\sim}{x}_a(t)\, \underset{\sim}{v}_t^T(t)\right| \qquad (2.49)$$

Now, using the solution form of Eq (2.32), $\underline{P}_{x_a v_t}$ becomes

$$\underline{P}_{x_a v_t} = E\Big| \underline{\Phi}_a(t, t_o)\, \underset{\sim}{x}_a(t_o)\, \underset{\sim}{v}_t^T(t)$$

$$+ \int_{t_o}^{t} \underline{\Phi}_a(t,\tau) \left[\underline{B}_a(\tau)\, \underline{y}_d(\tau)\right.$$

$$\left. + \underline{G}_a(\tau)\, \underset{\sim}{w}_a(\tau)\right] \underset{\sim}{v}_t^T(t)\, d\tau \Big| \qquad (2.50)$$

The first term is zero since $\underset{\sim}{x}_a(t_o)$ and $\underset{\sim}{v}_t(t)$ are assumed independent and the mean of $\underset{\sim}{v}_t(t)$ is zero. Now after explicitly writing out the augmented matrices, $\underline{P}_{x_a v_t}$ is

$$\underline{P}_{x_a v_t} = E\left\{ \int_{t_o}^{t} \underline{\Phi}_a(t, \tau) \left(\begin{bmatrix} \underline{B}_t(\tau)\, \underline{G}_{cy}(\tau) \\ \underline{B}_{cy}(\tau) \end{bmatrix} \underline{y}_d \right.\right.$$

$$\left.\left. + \begin{bmatrix} \underline{G}_t(\tau)\, \underset{\sim}{w}_t(\tau) + \underline{B}_t(\tau)\, \underline{G}_{cz}(\tau)\, \underset{\sim}{v}_t(\tau) \\ \underline{B}_{cz}(\tau)\, \underset{\sim}{v}_t(\tau) \end{bmatrix} \right) \underset{\sim}{v}_t^T(t)\, d\tau \right\} \qquad (2.51)$$

$$\underline{P}_{x_a v_t} = \int_{t_o}^{t} \underline{\Phi}_a(t, \tau)\, E\left\{ \underline{B}_a(\tau)\, \underline{y}_d(\tau)\, \underline{v}^T(t) + \left[\begin{matrix} \underline{G}_t(\tau)\, \underline{w}_t(\tau)\, \underline{v}_t^T(t) + \underline{B}_t(\tau)\, \underline{G}_{cz}(\tau)\, \underline{v}_t(\tau)\, \underline{v}_t^T(t) \\ \underline{B}_{cz}(\tau)\, \underline{v}_t(\tau)\, \underline{v}_t^T(\tau) \end{matrix} \right] \right\} d\tau \tag{2.52}$$

The first term in Eq (2.52) is zero since $\underline{v}_t(\tau)$ is zero-mean and the only random variable factor in the expression. Also, the term with $\underline{G}_t(\tau)\, \underline{w}_t(\tau)\, \underline{v}_t^T(t)$ is zero since $\underline{v}_t(t)$ and $\underline{w}_t(t)$ are assumed to be independent and zero mean. This leaves a constant matrix multiplying $\underline{v}_t(\tau)\, \underline{v}_t^T(t)$ as in Eq (2.53)

$$\underline{P}_{x_a v_t} = \int_{t_o}^{t} \underline{\Phi}_a(t, \tau) \begin{bmatrix} \underline{B}_t(\tau)\, \underline{G}_{cz}(\tau) \\ \underline{B}_{cz}(\tau) \end{bmatrix} E\left\{ \underline{v}_t(\tau)\, \underline{v}_t^T(t) \right\} d\tau \tag{2.53}$$

The factor $E\ \underline{v}_t(\tau)\, \underline{v}_t^T(t)$ is defined to be $\underline{R}_t(t)\ \ \delta(t - \tau)$ in Eq (2.5). Now applying the dirac delta sifting property where t is the upper limit of the integration (Ref 8), Eq (2.53) becomes

$$\underline{P}_{x_a v_t} = \underline{\Phi}_a(t, t) \begin{bmatrix} \underline{B}_t(t)\, \underline{G}_{cz}(t) \\ \underline{B}_{cz}(t) \end{bmatrix} \tfrac{1}{2}\, \underline{R}_t(t) \tag{2.54}$$

The state transition matrix evaluated from time t to time t is the identity matrix, $\underline{I}$. The factor of ½ in Eq (2.54) is a result of integrating the dirac delta function over the range t_o and t instead of integrating τ out past time t. The final result is

$$\underline{P}_{x_a v_t} \approx \tfrac{1}{2} \begin{bmatrix} \underline{B}_t \underline{G}_{cz} \\ \underline{B}_{cz} \end{bmatrix} \underline{R}_t \qquad (2.55)$$

At this point all necessary computational forms have been derived for a performance analysis of a linear continuous-time, time-varying system and controller. The performance analysis software implements a time invariant version of the general form given above. Accordingly, it requires $\underline{G}_{cx}$, $\underline{G}_{cy}$, $\underline{G}_{cz}$, $\underline{B}_{cy}$, $\underline{B}_{cz}$ and $\underline{F}_c$ be specified for Eq (2.23) by the user in addition to the truth model dynamics equation and measurement equation matrices. The flowcharts and Fortran source code for this software are in Appendices A and B respectively.

As noted above, this is a general performance analysis routine and can analyze the performance of any continuous-time controller. It will be used in this study only to characterize the performance of several different LQG regulating controllers. To put the LQG regulator into the proper format for this routine, let $\underline{x}_c$ in Eqs (2.23A) and (2.23B) be the state estimate $\hat{\underline{x}}$ from the Kalman filter such that Eq (2.6) becomes

$$\dot{\underline{x}}_c = \underline{F}_f \underline{x}_c + \underline{B}_f \underline{u} + \underline{K}_f (\underline{z}_t - \underline{H}_f \underline{x}_c) \qquad (2.56)$$

The subscript "f" indicates a quantity associated with the Kalman filter. The optimal control law for an LQG regulator is $\underline{u}^* = -\underline{G}_c^* \underline{x}_c$, implying from Eq (2.23A) that $\underline{G}_{cx} = -\underline{G}_c^*$, $\underline{G}_{cy} = \underline{0}$, $\underline{v}_d = \underline{0}$, and $\underline{G}_{cz} = \underline{0}$. Now substituting this into Eq (2.56), it becomes

$$\dot{\underline{x}}_c = \underline{F}_f \underline{x}_c + \underline{B}_f (-\underline{G}_c^* \underline{x}_c) + \underline{K}_f (\underline{z}_t - \underline{H}_f \underline{x}_c) \qquad (2.57)$$

Matching like quantities from Eqs (2.56) and (2.23B) implies that for the LQG regulator

$$\underline{F}_c = \underline{F}_f - \underline{B}_f \underline{G}_c^* - \underline{K}_f \underline{H}_f \tag{2.58}$$

$$\underline{B}_{cz} = \underline{K}_f \tag{2.59}$$

$$\underline{B}_{cy} = \underline{0} \tag{2.60}$$

The flowcharts and FORTRAN source code for the software routine to put the LQG regulator into this format are in Appendices A and B respectively.

Enhancing Robustness in Continuous-Time LQG Controllers

An automatic control system exhibits stability robustness when the closed-loop system remains stable even though certain system design parameters change from their design values or when other unmodeled variations occur. More precisely, robustness specifies the finite regions in parameter space of the design model around a nominal model in which stable closed-loop system operation is preserved. Some recent papers (Refs 5 and 9) deal with robustness only in regard to parameter variations within the controlled system plant matrix, robustness actually encompasses all possible variations in design models that affect closed-loop system stability (Ref 10).

There are many guarantees of robustness for control systems designed using full-state feedback (Ref 10). In many cases, however, full-state feedback is not available or is impractical. In these cases an observer or state estimator is often used to supply estimates of all the states. While

there are claims about robustness of systems using observers in the literature (Refs 12 and 13), J.C. Doyle (Ref 3) proved in 1978 that there are no robustness guarantees in general. Since then Doyle and Stein (Ref 2) have developed a technique, applicable to Linear Quadratic Gaussian continuous-time controllers, that recovers some of the robustness properties of a full-state feedback system. Their simple technique, which assumes the n-state plant is controllable, observable, and has no transmission zeros in the right half plane, requires choosing the gain for the Kalman filter in the controller in a particular way.

Doyle and Stein's technique is based on making the return difference mappings for full-state feedback controllers and observer based controllers equal. (See Fig 2.3). When these mappings, or loop transfer functions, are asymptotically equal for the control loops broken at the input to the physical system (point x in Fig 2.3) then the robustness properties of the full-state feedback controller can be asymptotically recovered by the observer based controller (Ref).

The return difference mappings of Fig (2.3a) and (2.3b) are identical if the observer dynamics satisfy

$$\underline{K}_f \left[\underline{I} + \underline{H} (s \underline{I} - \underline{F})^{-1} \underline{K}_f\right] = \underline{B} \left[\underline{H} (s \underline{I} - \underline{F})^{-1} \underline{B}\right]^{-1} \quad (2.61)$$

If $\underline{K}_f$ is parameterized as a function of a scalar q , as $\underline{K}_f(q)$, then Eq (2.61) will be satisfied asymptotically as $q \to \infty$ if

$$\frac{\underline{K}_f(q)}{q} \to \underline{B}\,\underline{W} \quad (2.62)$$

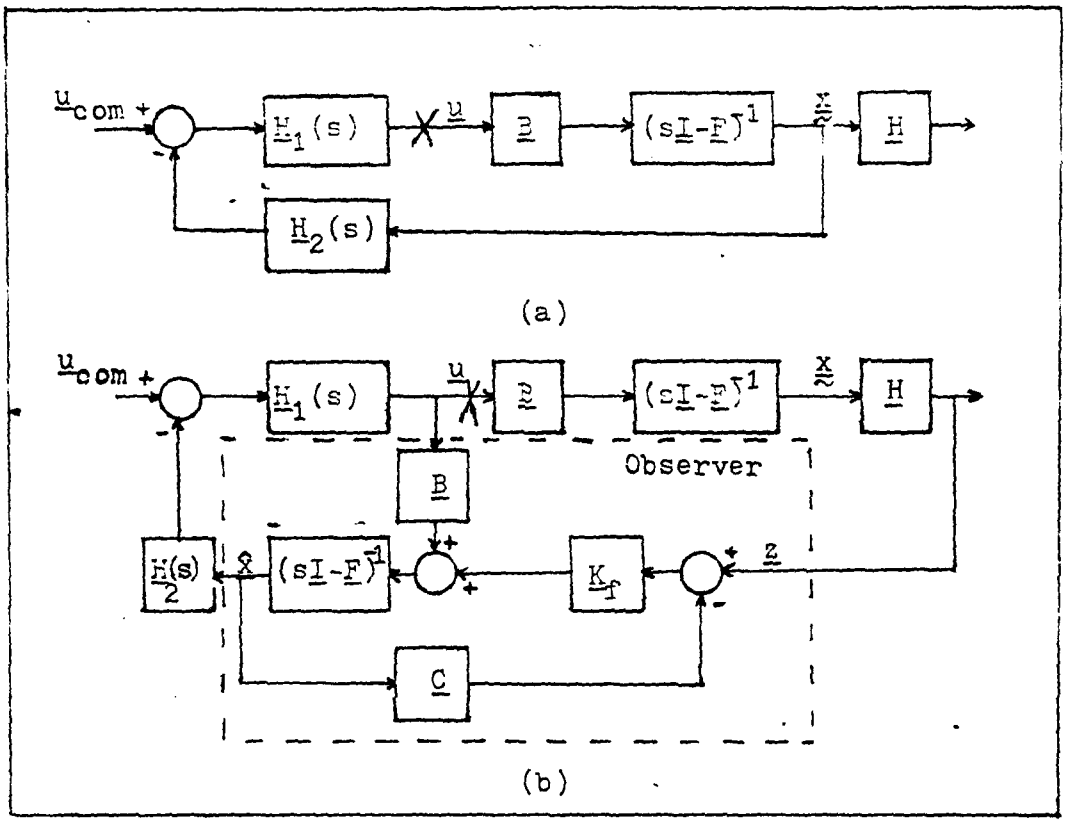


Fig 2.3 a) Full-state feedback, b) Observer based implementation (Ref 2)

where $\underline{W}$ is any nonsingular matrix. When this requirement is implemented using a Kalman filter to insure stable error dynamics, $\underline{K}_f(q)$ becomes

$$\underline{K}_f(q) = \underline{\bar{P}}(q)\ \underline{H}^T\ \underline{R}_c^{-1} \tag{2.63}$$

where $\underline{\bar{P}}(q)$ is used to replace $\underline{P}$ in the matrix Riccati Eq (2.13).

Using their technique involves changing the value of $\underline{G}\ \underline{Q}\ \underline{G}^T$ used in Eq (2.13). They define $\underline{Q}_o$ to be the original $\underline{G}\ \underline{Q}\ \underline{G}^T$ of the system and $\underline{Q}(a)$ to be their modified $\underline{Q}$ to be used in place of $\underline{G}\ \underline{Q}\ \underline{G}^T$ in Eq (2.13. They define

$$\underline{Q}(q) \triangleq \underline{Q}_o + q^2 \underline{B}\,\underline{V}\,\underline{B}^T \tag{2.64}$$

where q is a design parameter and is set as desired. Note that q= 0 gives $\underline{Q}(q) = \underline{Q}_o$. As q approaches ∞, the robustness properties of full-state feedback controllers are recovered. Doyle and Stein state however, that some robustness may be recovered even for small values of q, i.e., for q= 1, 10, 100. In Eq (2.64) the $\underline{V}$ matrix is also a design parameter with the stipulation that it must be positive, definite and symmetric (Ref 1). Note that Eq (2.63) physically corresponds to pseudo-noise being added at the points of entry of $\underline{u}$ rather than the entry points of the original dynamics noise $\underset{\sim}{w}(t)$.

When Eq (2.64) is the basis of calculating the Kalman filter gain $\underline{K}_f$, the steady state covariance equation Eq (2.13) divided by q^2 is

$$\frac{\dot{\underline{P}}}{q^2} = \underline{F}\left(\frac{\underline{P}}{q^2}\right) + \left(\frac{\underline{P}}{q^2}\right)\underline{F}^T + \frac{\underline{Q}_o}{q^2} + \underline{B}\,\underline{V}\,\underline{B}^T$$

$$- q^2 \left(\frac{\underline{P}}{q^2}\right) \underline{H}^T \underline{R}_c^{-1} \underline{H} \left(\frac{\underline{P}}{q^2}\right) = 0 \tag{2.65}$$

then as $q \to \infty$

$$\left(\frac{\underline{P}}{q^2}\right) \to 0 \tag{2.66}$$

and

$$q^2 \left(\frac{\underline{F}}{q^2}\right) \underline{H}^T \underline{R}_c^{-1} \underline{H} \left(\frac{\underline{F}}{q^2}\right) \rightarrow \underline{B}\,\underline{V}\,\underline{B}^T \tag{2.67}$$

and also upon making appropriate substitutions,

$$\frac{\underline{K}_f \underline{R}_c \underline{K}_f^T}{q^2} \rightarrow \underline{B}\,\underline{V}\,\underline{B}^T \tag{2.68}$$

Solutions to Eq (2.68) are of the form

$$\frac{1}{q}\underline{K}_f \rightarrow \underline{B}\,\underline{V}^{\frac{1}{2}}\,(\underline{R}_c^{\frac{1}{2}})^{-1} \tag{2.69}$$

where $\underline{V}^{\frac{1}{2}}$ is some square root of $\underline{V}$ and $\underline{R}_c^{\frac{1}{2}}$ is some square root of $\underline{R}_c$. Eq (2.69) is a special case of Eq (2.62) so it follows that the given $\underline{Q}_o$ adjustment procedure in Eq (2.64) will achieve the desired robustness improvement objective (Ref 2).

In the evaluation of this technique, several different values of q are used, with $\underline{V} = \underline{I}$. Choosing $\underline{V} \neq \underline{I}$ allows selective weighting of the pseudonoise added to each state. A Fortran software routine was written to provide for the adjustments indicated by Eq (2.64). The flowcharts and FORTRAN source code are in Appendices A and B respectively.

The Model

The model chosen for the basis of this study is the thrust vector control system for the docked configuration of the Apollo Command and Service Module (CSM) and the Lunar Module (LM). Maybeck (Ref 8) is the source for this model description and contains a more detailed description. There is only one uncertain parameter in the system description used

and that is the natural bending frequency of the docked combination.

The Apollo CSM/LM vehicle is initially aligned using small attitude control jets. The main engine is then ignited and the proper attitude is maintained by the thrust vector control system (TVCS). In addition to this function, the TVCS also attempts to counteract any rigid body rotations or bending motions. This is necessary to minimize the stress on the docking tunnel between the CSM/LM (Ref 8).

Only the model for the pitch plane with the most significant bending mode is used. The rigid body motion for this system is described by (Ref 8):

$$\dot{\underset{\sim}{\omega}}(t) = \frac{-T(L\sigma_e + d_e)}{I}\ \underset{\sim}{q}(t) + \frac{T}{I}\left[\underset{\sim}{\delta}(t) + \underset{\sim}{w}(t)\right] \qquad (2.70)$$

$$\dot{\underset{\sim}{\theta}}(t) = \underset{\sim}{\omega}(t) \qquad (2.71)$$

where

$\underset{\sim}{\omega}(t)$= rigid body angular velocity with respect to the inertial reference frame

$\underset{\sim}{\theta}(t)$= angular attitude relative to inertial space

T= thrust of engine; 22,000 lbs

I= pitch moment of inertial of the rigid vehicle; 370,000 slug-ft^2

L= distance between center of mass and engine; 19 ft

σ_e= slope of bending mode at the engine station; -.13 radian/ft

d_e= displacement of bending mode at engine station; 1.1 ft/ft

$\underset{\sim}{q}(t)$= generalized bending coordinate

$\underset{\sim}{\delta}(t)$= main engine nozzle angle relative to the CSM

$\underset{\sim}{w}(t)$= a white noise superimposed on $\delta(t)$

The bending mode dynamics are described by the state variables $v_b(t)$ and $q(t)$; the velocity and position of the generalized bending coordinate. They are related to the other system variables by

$$\dot{\underset{\sim}{v}}_b(t) = -\underline{\omega}_b^2 \, \underset{\sim}{q}(t) - a \, d_e \left[\underset{\sim}{\delta}(t) + \underset{\sim}{w}(t)\right] \tag{2.72}$$

$$\dot{\underset{\sim}{q}}(t) = \underset{\sim}{v}_b(t) \tag{2.73}$$

where

a= vehicle acceleration due to main engine thrust; 10 ft/sec^2

ω_b= the natural frequency of the bending mode; value is uncertain

In addition, the main engine servo-mechanism can be modeled as

$$\dot{\delta}(t) = -1/\tau \, \delta(t) + 1/\tau \, \delta_{com}(t) \tag{2.74}$$

where

δ_{com}= the commanded value of engine gimbal angle; output of controller

τ = lag time constant with which the engine follows the command; .1 sec

If $\delta_{com}(t)$ is known, for instance as a computed input (see Deterministic State Augmentation) then $\delta(t)$ is known deterministically; otherwise, if $\delta_{com}(t)$ is random, then $\delta(t)$ will also be random. Note that the peak rate limit of .1 radian/sec will be accounted for in the cost function of the

optimal controller (Ref 7).

The white noise disturbance $\underset{\sim}{w}(t)$ occurs as vibration at the bottom end of the CSM as a result of engine firing. It is assumed to enter the equations as a random thrust vector angle. Therefore, the true nozzle angle is composed of a deterministic portion $\delta(t)$ and a statistically random portion $\underset{\sim}{w}(t)$. The mean of $\underset{\sim}{w}(t)$ is zero and it has a low frequency spectral density of 0.0004 radian^2 per cycle per second. This disturbance could cause a lateral velocity of about 2 ft/sec during a 100 second engine firing (Ref 7).

Combining the above information into a five dimensional state vector equation, the vehicle dynamics are governed by:

$$\frac{d}{dt}\begin{bmatrix} \underset{\sim}{\omega}(t) \\ \underset{\sim}{\theta}(t) \\ \underset{\sim}{v}_b(t) \\ \underset{\sim}{q}(t) \\ \delta(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0.0815 & 1.13 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\omega_b^2 & -11 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -10 \end{bmatrix} \begin{bmatrix} \underset{\sim}{\omega}(t) \\ \underset{\sim}{\theta}(t) \\ \underset{\sim}{v}_b(t) \\ \underset{\sim}{q}(t) \\ \delta(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 10 \end{bmatrix} \delta_{com}(t) + \begin{bmatrix} 1.13 \\ 0 \\ -11 \\ 0 \\ 0 \end{bmatrix} \underset{\sim}{w}(t) \qquad (2.75)$$

By processing Inertial Measurement Unit (IMU) data with a suitable algorithm, a continuous-time measurement $z(t)$ or a sampled data measurement $z(t_i)$ can be obtained:

$$\underset{\sim}{z}(t) = \underset{\sim}{\theta}(t) + \sigma_a \underset{\sim}{q}(t) + \underset{\sim}{v}_c(t) \qquad (2.76A)$$

$$\underset{\sim}{z}(t_i) = \underset{\sim}{\theta}(t_i) + \sigma_a \underset{\sim}{q}(t_i) + \underset{\sim}{v}(t_i) \qquad (2.76B)$$

where $\underset{\sim}{\theta}(t)$ and $\underset{\sim}{q}(t)$ are as before

$\underset{\sim}{\theta}(t_i)$= $\underset{\sim}{\theta}(t)$ at time t= t_i

$\underset{\sim}{q}(t_i)$= $\underset{\sim}{q}(t)$ at time t= t_i

$\underset{\sim}{v}(t_i)$= discrete-time white Gaussian measurement noise with mean zero and variance $1/12(0.0002)^2$ radian^2

$\underset{\sim}{v}_c(t)$= continuous-time zero mean white Gaussian measurement noise of strength approximated by the strength of $\underset{\sim}{v}(t_i)$ times Δt, the sample period over which the measurements were actually made: $R_c(t_i) = R(t_i)\,\Delta t$

σ_a= slope of the bending mode at the IMU station; -0.13 radian/ft

Note the approximation to $R_c(t_i)$ is motivated by a derivation of the continuous-time IQG controller which starts with a sampled-data controller and then allowing the sample time to approach zero (Ref 8).

In the real system the measurement is made available once every 0.1 sec. $\underset{\sim}{z}(t)$ can be written in a more compact form, $\underline{H}\,\underset{\sim}{x}(t) + \underset{\sim}{v}_c(t)$, as

$$\underset{\sim}{z}(t) = [0 \quad 1 \quad 0 \quad -0.13 \quad 0] \begin{bmatrix} \underset{\sim}{\omega}(t) \\ \underset{\sim}{\theta}(t) \\ \underset{\sim}{v}_i(t) \\ \underset{\sim}{q}(t) \\ \delta(t) \end{bmatrix} + \underset{\sim}{v}_c(t) \qquad (2.77)$$

The following conditions and a priori knowledge are assumed and are consistent with those used by Maybeck (Ref 3) in an adaptive controller for this model. Eqs (2.78A) through (2.78D) apply to the truth model, and Eqs. (2.79A) and (2.79B) apply to the controller model.

$$\omega_o = 0.08 \text{ degree/sec} \tag{2.78A}$$

$$\theta_o = 0.8 \text{ degree} \tag{2.78B}$$

$$v_{b_o} = 0.7 \text{ ft/sec} \tag{2.78C}$$

$$q_o = 0.07 \text{ ft} \tag{2.78D}$$

$$\omega_b^2 = (10 \text{ rad/sec})^2 \tag{2.79A}$$

$$\underline{\hat{x}}_o = \underline{0} \tag{2.79B}$$

The value of ω_b^2 in the truth model is set at various values - 90, 100, 110, 200, 300....., rad^2/sec^2. Note the ω_b^2 in Eq (2.79A) is the value upon which the controller model is based and is different from that in the true model. The effect on controller performance of several different values for ω_b^2 in the truth model is evaluated (Ref 8). The cost-weighting matrices, as specified in Maybeck (Ref 11) are

$$\underline{W}_{xx} = \begin{bmatrix} 4.4(10)^7 & 0 & 0 & 0 & 0 \\ 0 & 185000 & 0 & 0 & 0 \\ 0 & 0 & 185000 & 0 & 0 \\ 0 & 0 & 0 & 1100 & 0 \\ 0 & 0 & 0 & 0 & 165000 \end{bmatrix} \tag{2.80}$$

$$\underset{\sim}{W}_{uu} = \begin{bmatrix} 4.4 & (10)^7 \end{bmatrix} \tag{2.81}$$

$$\underset{\sim}{W}_{xu} = \begin{bmatrix} -4.4(10)^7 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.82}$$

The entire system is depicted in Fig 2.4. In Fig 2.4 it is evident that the driving noise $\underset{\sim}{w}(t)$ does not enter the state $\delta(t)$. Accordingly, then, only four states can and need be estimated by a Kalman filter in the LQG controller. The four state models on which the Kalman filter is based is

$$\frac{d}{dt}\begin{bmatrix} \underset{\sim}{\omega}(t) \\ \underset{\sim}{\theta}(t) \\ \underset{\sim}{v}_b(t) \\ \underset{\sim}{q}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0.0815 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\omega_b^2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \underset{\sim}{\omega}(t) \\ \underset{\sim}{\theta}(t) \\ \underset{\sim}{v}_b(t) \\ \underset{\sim}{q}(t) \end{bmatrix} + \begin{bmatrix} 1.13 \\ 0 \\ -11 \\ 0 \end{bmatrix} \delta(t) + \begin{bmatrix} 1.13 \\ 0 \\ -11 \\ 0 \end{bmatrix} \underset{\sim}{w}(t) \tag{2.83}$$

The deterministic state is handled in the manner described in the following section, Deterministic State Augmentation section.

Deterministic State Augmentation

In some cases, as in the model of the preceding section, certain states of a controller will be known deterministically as a function of the computed control value. A priori, they are random, but they are functions of the computed u, which is not random once computed. If these states are introduced

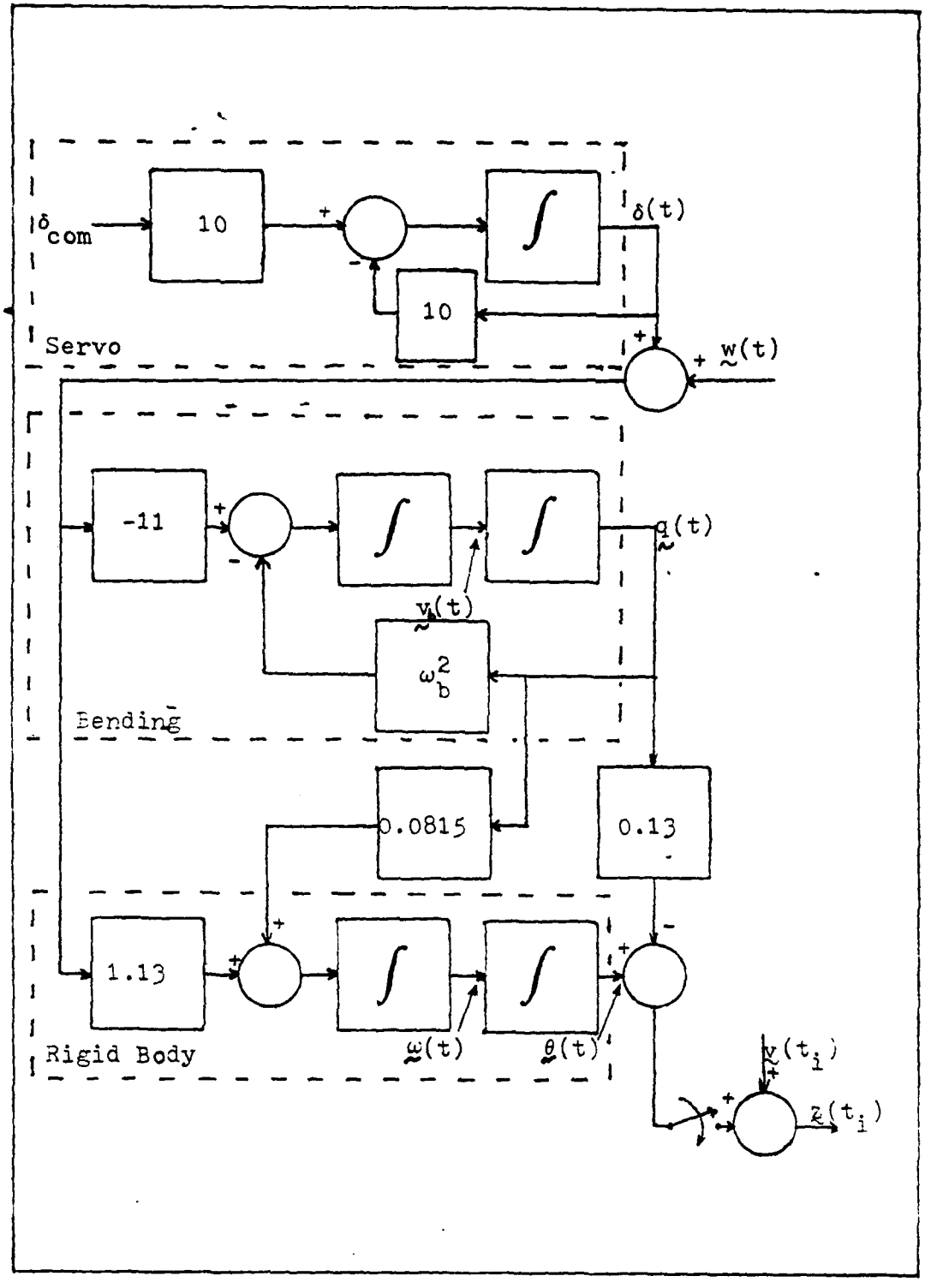


Fig 2.4 Thrust Vector Control Dynamics (Ref 8)

into the Kalman filter equations, the associated Kalman filter gain's calculations may become intractable. In particular, these inputs are not controllable from the entry point of $\underset{\sim}{w}(t)$. It is therefore necessary to remove these states from the controller design model while generating the Kalman filter. Following that, they must be augmented again into the controller model (Ref 11).

Let a controller model and the measurements upon which it is based be described by

$$\dot{\underset{\sim}{x}} = \underline{F}\,\underset{\sim}{x} + \underline{B}\,\underset{\sim}{u} + \underline{G}\,\underset{\sim}{w} \tag{2.84}$$

$$\underset{\sim}{z} = \underline{H}\,\underset{\sim}{x} + \underset{\sim}{v} \tag{2.85}$$

If there are deterministic states, then the system and measurement equations may be put in forms,

$$\begin{bmatrix} \dot{\underset{\sim}{x}}_1 \\ \dot{\underset{\sim}{x}}_2 \end{bmatrix} = \begin{bmatrix} \underline{F}_{11} & \underline{0} \\ \underline{F}_{21} & \underline{F}_{22} \end{bmatrix} \begin{bmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{bmatrix} + \begin{bmatrix} \underline{B}_1 \\ \underline{B}_2 \end{bmatrix} \underset{\sim}{u} + \begin{bmatrix} \underline{0} \\ \underline{G}_2 \end{bmatrix} \underset{\sim}{w} \tag{2.86}$$

$$\underset{\sim}{z} = \begin{bmatrix} \underline{H}_1 & \underline{H}_2 \end{bmatrix} \begin{bmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{bmatrix} + \underset{\sim}{v}_c \tag{2.87}$$

by means of appropriate ordering of state variables. The vector $\underset{\sim}{x}_1$ of dimension p contains all deterministic states and the vector $\underset{\sim}{x}_2$ of dimension m contains the stochastic states. The zero matrices in the partitioned $\underline{F}$ and $\underline{G}$ matrices indicate that there is no direct noise inputs into states $\underset{\sim}{x}_1$ and that $\underset{\sim}{x}_2$ is not directly coupled into $\underset{\sim}{x}_1$. Note that $\underline{F}_{21} \neq \underline{0}$ allows the stochastic states to be functions of the deterministic states.

Now to produce an estimate of $\underset{\sim}{x}_2$, $\hat{\underset{\sim}{x}}_2$, a Kalman filter, for a system partitioned as in Eq(2.86), can be determined using Eqs (2.2) through (2.13) so that

$$\dot{\hat{\underset{\sim}{x}}}_2 = \underline{F}_{22}\,\hat{\underset{\sim}{x}}_2 + \underline{F}_{21}\,\underset{\sim}{x}_1 + \underline{B}_2\,\underline{u} + \underline{K}_f\left(\underline{z} - \begin{bmatrix}\underline{H}_1 & \underline{H}_2\end{bmatrix}\begin{bmatrix}\underset{\sim}{x}_1 \\ \hat{\underset{\sim}{x}}_2\end{bmatrix}\right) \tag{2.88}$$

Note, the Kalman equations require only the m by m $\underline{F}_{22}$ matrix and the m by r $\underline{G}_2$ matrix (where r is the dimension of $\underset{\sim}{w}$) from Eq (2.86) in order to compute the Kalman filter gains.

Once the Kalman filter gains $\underline{K}_f$ are determined, it is necessary to reform the complete controller model as in Eq (2.89)

$$\dot{\underset{\sim}{x}}_c = \begin{bmatrix}\dot{\underset{\sim}{x}}_1 \\ \dot{\hat{\underset{\sim}{x}}}_2\end{bmatrix} = \begin{bmatrix}\underline{F}_{11} & \underline{0} \\ \underline{F}_{21} & \underline{F}_{22}\end{bmatrix}\underset{\sim}{x}_c + \begin{bmatrix}\underline{B}_1 \\ \underline{B}_2\end{bmatrix}\underline{u} + \begin{bmatrix}\underline{0} \\ \underline{K}_f\end{bmatrix}(\underline{z} - \underline{H}\,\underset{\sim}{x}_c) \tag{2.89}$$

Now the controller will provide values for all controller states: known values for the deterministic states and estimates of the stochastic states.

Sampled-Data LQG Controller

The following discussion of the sampled-data LQG controller is based on the presentation by Maybeck (Ref10). It assumes that the underlying physical system to be controlled can be represented by

$$\underset{\sim}{x}(t_{i+1}) = \underset{\sim}{\Phi}(t_{i+1}, t_i)\ \underset{\sim}{x}(t_i) + \underset{\sim}{B}_d(t_i)\ \underset{\sim}{u}(t_i) + \underset{\sim}{G}_d(t_i)\ \underset{\sim}{w}_d(t_i) \qquad (2.90)$$

where $\underset{\sim}{\Phi}(t_{i+1}, t_i)$ is the state transition matrix and $\underset{\sim}{B}_d(t_i)$, $\underset{\sim}{G}_d(t_i)$ and $\underset{\sim}{w}_d(t_i)$ are the discrete-time counterparts of $\underset{\sim}{B}(t)$, $\underset{\sim}{G}(t)$ and $\underset{\sim}{w}(t)$ described previously for continuous-time systems. Note, if the underlying physical system is a continuous-time system as in Fig (2.5), then Eq (2.90) represents the equivalent discrete-time representation of that system as opposed to an approximate discrete-time representation (Ref 10).

A sampled-data controller for the system of Fig (2.5) is an optimal controller in the sense that it minimizes J in Eq (2.91)

$$J = E\left\{ \tfrac{1}{2}\, \underset{\sim}{x}^T(t_{N+1})\ \underset{\sim}{X}_f\ \underset{\sim}{x}(t_{N+1}) + \sum_{i=0}^{N} \tfrac{1}{2}\left[\underset{\sim}{x}^T(t_i)\ \underset{\sim}{X}(t_i)\ \underset{\sim}{x}(t_i) + \underset{\sim}{u}^T(t_i)\ \underset{\sim}{U}(t_i)\ \underset{\sim}{u}(t_i) + 2\ \underset{\sim}{x}^T(t_i)\ \underset{\sim}{S}(t_i)\ \underset{\sim}{u}(t_i)\right]\right\} + \sum_{i=0}^{N} J_r(t_i) \qquad (2.91)$$

In Eq (2.91) $\underset{\sim}{X}_f$ is the cost-weighting matrix for the final state which occurs at the final time t_{N+1}, $\underset{\sim}{X}(t_i)$ is the cost-weighting matrix for the states at time t_i, $\underset{\sim}{U}(t_i)$ is the cost-weighting matrix for applying controls at time t_i, $\underset{\sim}{S}(t_i)$ is the cost-weighting matrix at time t_i relating certain control values to certain states, and $J_r(t_i)$ is a residual cost. Note that the applied control is held constant throughout each interval between sample times and that no control is applied at the final time, t_{N+1}.

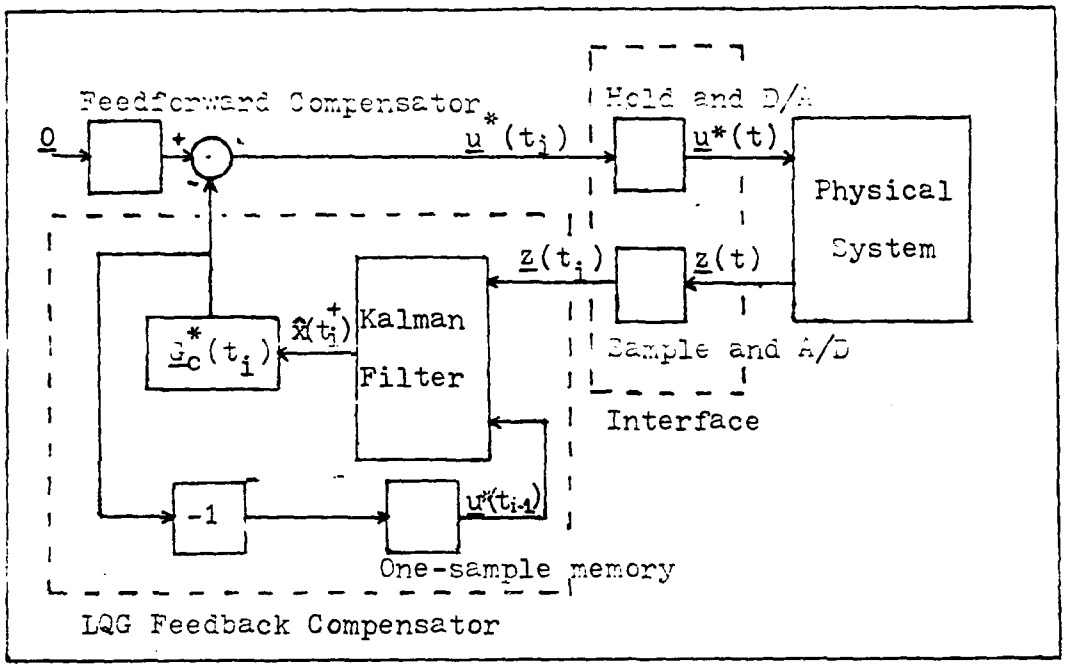


Fig 2.5 Sampled-Data LQG Controller (Ref 10)

When Eq (2.90) is an equivalent discrete-time representation of a system, a complete characterization of the states and cost can be generated by simultaneously integrating the differential equations, Eq (2.92) through (2.98) forward from time t to t_i. (Note $\underline{G}_d = \underline{I}$ for such a representation).

$$\dot{\bar{\underline{\Phi}}}(t, t_i) = \underline{F}(t)\, \bar{\underline{\Phi}}(t, t_i) \tag{2.92}$$

$$\dot{\bar{\underline{B}}}(t, t_i) = \underline{F}(t)\, \bar{\underline{B}}(t, t_i) + \underline{B}(t) \tag{2.93}$$

$$\dot{\bar{\underline{Q}}}(t, t_i) = \underline{F}(t)\, \bar{\underline{Q}}(t, t_i) + \bar{\underline{Q}}(t, t_i)\, \underline{F}^T(t) + \underline{G}(t)\, \underline{Q}(t)\, \underline{G}(t) \tag{2.94}$$

$$\dot{\bar{\underline{X}}}(t) = \underline{\Phi}^T(t, t_i)\, \underline{W}_{xx}(t)\, \underline{\Phi}(t, t_i) \tag{2.95}$$

$$\dot{\underline{U}}(t, t_i) = \underline{\bar{B}}^T(t, t_i)\, \underline{W}_{xx}(t)\, \underline{\bar{B}}(t, t_i) + \underline{W}_{uu}(t) + \underline{\bar{B}}^T(t, t_i)\, \underline{W}_{xu}(t) + \underline{W}_{xu}^T(t)\, \underline{\bar{B}}(t, t_i) \quad (2.96)$$

$$\dot{\underline{S}}(t, t_i) = \underline{\Phi}^T(t, t_i)\, \underline{W}_{xx}(t)\, \underline{\bar{B}}(t, t_i) + \underline{\Phi}^T(t, t_i)\, \underline{W}_{xu}(t) \quad (2.97)$$

$$\dot{\bar{J}}_r(t, t_i) = \mathrm{tr}\left[\underline{W}_{xx}(t)\, \underline{\bar{Q}}(t, t_i)\right] \quad (2.98)$$

Initial conditions for all integrations are $\underline{0}$, except for $\underline{\Phi}(t_i, t_i) = \underline{I}$. At the completion of the integration to t_{i+1}, the desired results are $\underline{\Phi}(t_{i+1}, t_i)$, $\underline{B}_d(t_i) = \bar{\underline{B}}(t_{i+1}, t_i)$, $\underline{X}(t_i) = \bar{\underline{X}}(t_{i+1}, t_i)$, $\underline{U}(t_i) = \bar{\underline{U}}(t_{i+1}, t_i)$, $\underline{S}(t_i) = \bar{\underline{S}}(t_{i+1}, t_i)$ and $\underline{J}_r(t_i) = \bar{\underline{J}}_r(t_{i+1}, t_i)$. The integration must be performed for every sample period, except in the case of a time invariant system model with constant cost-weighting matrices and stationary noise inputs with fixed sampling period, where the integrations need only be performed once. In this later case, the $\underline{\Phi}$, $\underline{B}_d$ and $\underline{G}_d$ matrices in Eq (2.90) and the $\underline{X}$, $\underline{U}$ and $\underline{S}$ matrices in Eq (2.91) are constant matrices (Ref 10).

For a system described by Eq (2.90), the LQG regulator consists of an optimal deterministic state feedback controller cascaded with a sampled-data Kalman filter as in Fig (2.5). The Kalman filter provides an estimate of the states. This conditional mean estimate $\hat{\underline{x}}(t_i^+)$ is described by Eqs (2.99A) through (2.99E).

$$\hat{\underline{x}}(t_i^-) = \underline{\Phi}(t_i, t_{i-1})\, \hat{\underline{x}}(t_{i-1}^+) + \underline{B}_d(t_{i-1})\, \underline{u}(t_{i-1}) \quad (2.99A)$$

$$\underline{P}(t_i) = \underline{\Phi}(t_i^-, t_{i-1})\ \underline{P}(t_{i-1}^+)\ \underline{\Phi}^T(t_i, t_{i-1}) + \underline{G}_d(t_{i-1})\ \underline{Q}_d(t_{i-1})\ \underline{G}_d(t_{i-1}) \tag{2.99B}$$

$$\underline{K}(t_i) = \underline{P}(t_i^-)\ \underline{H}^T(t_i) \left[\underline{H}(t_i)\ \underline{P}(t_i^-)\ \underline{H}^T(t_i) + R(t_i)\right]^{-1} \tag{2.99C}$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i) \left[\underline{z}_i - \underline{H}(t_i)\ \underline{x}(t_i^-)\right] \tag{2.99D}$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-) - K(t_i)\ \underline{H}(t_i)\ \underline{P}(t_i^-) \tag{2.99E}$$

The initial conditions necessary for beginning the recursions indicated by Eqs (2.99A) through (2.99E) are the a priori knowledge of $\hat{\underline{x}}_o$ and $\underline{P}_o$, that is,

$$\hat{\underline{x}}(t_o) = \hat{\underline{x}}_o \tag{2.100A}$$

$$\underline{P}(t_o) = \underline{P}_o \tag{2.100B}$$

$\underline{Q}_d(t_i)$ in Eq (2.99B) represents the covariance of the assumed zero mean input noise $\underline{w}_d(t_i)$, that is,

$$E\left[\underline{w}_d(t_i)\ \underline{w}_d^T(t_j)\right] = \begin{cases} \underline{Q}_d(t_i) & t_i = t_j \\ \underline{0} & t_j \neq t_j \end{cases} \tag{2.101}$$

$\underline{z}_i$ in Eq (2.99D) is the measurement available at time t_i and is of the form

$$\underline{z}(t_i) = \underline{H}(t_i)\ \underline{x}(t_i) + \underline{v}(t_i) \tag{2.102}$$

where $\underline{v}(t_i)$ is an assumed zero-mean measurement noise of covariance $\underline{R}(t_i)$, that is,

$$E\left[\underset{\sim}{v}(t_i)\ \underset{\sim}{v}^T(t_j)\right] = \begin{cases} \underline{R}(t_i) & t_i = t_j \\ \underline{0} & t_i \neq t_j \end{cases} \qquad (2.103)$$

Note that $\underset{\sim}{w}_d(t_i)$ and $\underset{\sim}{v}(t_i)$ are also assumed to be independent of each other. The description of the various matrices and vectors in Eqs (2.99) through (2.103) parallels the continuous-time case with the exception that there are now two values for $\hat{\underline{x}}$ and $\underline{P}$. The value at t_i^- is the value before the measurement at t_i, $\underline{z}(t_i)$, is incorporated. The value at t_i^+ incorporates the new information made available by the measurement at time t_i (Ref 10).

The optimal deterministic controller to be cascaded with the Kalman filter in Fig (2.5) is described by

$$\underline{u}^*(t_i) = -\underline{G}_c^*(t_i)\ \underline{x}(t_i) \qquad (2.104)$$

Eq (2.104) assumes perfect knowledge of $\underline{x}(t_i)$ at the sample time. Since the certainty equivalence principle applies, $\underline{x}(t_i)$ can be replaced by $\hat{\underline{x}}(t_i^+)$ when knowledge of $\underline{x}(t_i)$ comes from incomplete noise corrupted measurements. $\underline{G}_c^*(t_i)$ is, from deterministic LQG controller theory (Ref10),

$$\underline{G}_c^*(t_i) = \left[\underline{U}(t_i) + \underline{B}_d^T(t_i)\ \underline{K}_c(t_{i+1})\ \underline{B}_d(t_i)\right]^{-1} \cdot \left[\underline{B}_d^T(t_i)\ \underline{K}_c(t_{i+1})\ \underline{\Phi}(t_{i+1}, t_i) + \underline{S}^T(t_i)\right] \qquad (2.105)$$

where $\underline{K}_c(t_i)$ satisfies the backward Riccati recursion

$$\underline{K}_c(t_i) = \underline{X}(t_i) + \underline{\Phi}^T(t_{i+1}, t_i)\, \underline{K}_c(t_{i+1})\, \underline{\Phi}(t_{i+1}, t_i) - \left[\underline{B}_d^T(t_i)\, \underline{K}_c(t_{i+1})\, \underline{\Phi}(t_{i+1}, t_i) + \underline{S}^T(t_i) \right]^T \underline{G}_c^*(t_i) \quad (2.106)$$

$$\underline{K}_c(t_{N+1}) = \underline{X}_f \quad (2.107)$$

Flowcharts and FORTRAN source code required to implement the sampled-data LQG controller appear in Appendices A and B respectively.

Sampled-Data Performance Analysis

This performance analysis is based on Fig (2.2), with the only difference being that a sampled-data controller is used versus a continuous-time measurement controller. This performance analysis scheme is from Maybeck (Ref 10).

In this scheme the truth model is represented by

$$\dot{\underset{\sim}{x}}_t(t) = \underline{F}_t(t)\, \underset{\sim}{x}_t(t) + \underline{B}_t(t)\, \underset{\sim}{u}(t) + \underline{G}_t\, \underset{\sim}{w}_t(t) \quad (2.108)$$

and the measurements available to the controller are sampled-data measurements of the form

$$\underset{\sim}{z}_t(t_i) = \underline{H}_t(t_i)\, \underset{\sim}{x}_t(t_i) + \underset{\sim}{v}_t(t_i) \quad (2.109)$$

The discrete-time controller, which is similar in form to Eq (2.23A) and (2.23B), is

$$\underline{u}(t_i) = \underline{G}_{cx}(t_i)\, \underline{x}_c(t_i) + \underline{G}_{cz}(t_i)\, \underline{z}_{ti} + \underline{G}_{cy}(t_i)\, \underline{y}_d(t_i) \quad (2.110A)$$

$$\underline{x}_c(t_{i+1}) = \underline{\Phi}_c(t_{i+1}, t_i)\, \underline{x}_c(t_i) + \underline{B}_{cz}(t_i)\, \underline{z}_{ti} + \underline{B}_{cy}(t_i)\, \underline{y}_d(t_i) \qquad (2.110B)$$

The primary differences between Eqs (2.23) and (2.110) are that the differential equation is replaced by a difference equation and that the counterpart to $\underline{z}_t(t)$ is written $\underline{z}_{ti}$ as opposed to $\underline{z}_t(t_i)$ (Ref 10).

Using a controller as described in Eq (2.110), and an equivalent discrete-time model for the truth model, the sampled-data performance analysis is very similar to the continuous-time performance analysis where $\underset{\sim}{x}_a(t)$ and $\underline{y}_a(t)$ become $\underset{\sim}{x}_a(t_i)$ and $\underline{y}_a(t_i)$. That is, if in Eqs (2.27) through (2.35) $\dot{\underline{x}}_c(t)$, $\underline{x}_c(t)$, $\dot{\underset{\sim}{x}}_t(t)$, $\underset{\sim}{x}_t(t)$, $\underline{F}_c(t)$, $\underline{F}_t(t)$, are replaced by $\underline{x}_c(t_{i+1})$, $\underline{x}_c(t_i)$, $\underset{\sim}{x}_t(t_{i+1})$, $\underset{\sim}{x}_t(t_i)$, $\underline{\Phi}_c(t_{i+1}, t_i)$, $\underline{\Phi}_t(t_{i+1}, t_i)$, respectively. Then $\underline{F}_a(t)$ becomes $\underline{\Phi}_a(t_{i+1}, t_i)$, $\underline{B}_a(t)$ becomes $\underline{B}_{d_a}(t_i)$ and $\underline{G}_a(t)$ becomes $\underline{G}_{d_a}(t_i)$ where the upper left partition in $\underline{G}_a$ is the identity matrix I, since this is an equivalent discrete-time representation. If the underlying truth model is a discrete-time system, an appropriate $\underline{G}_d$ would replace the $\underline{I}$.

The mean $\underline{m}_{x_a}(t_i)$ and the covariance $\underline{P}_{x_a}(t_i)$ of the internal process $\underline{x}_a$ are propagated by

$$\underline{m}_{x_a}(t_{i+1}) = \underline{\Phi}_a(t_{i+1}, t_i)\, \underline{m}_{x_a}(t_i) + \underline{B}_{d_a}(t_i)\, \underline{y}_d(t_i) \qquad (2.111A)$$

$$\underline{P}_{x_a}(t_{i+1}) = \underline{\Phi}(t_{i+1}, t_i)\, \underline{P}_{x_a}(t_i)\, \underline{\Phi}^T(t_{i+1}, t_i) + \underline{G}_{d_a}(t_i)\, \underline{Q}_{d_a}(t_i)\, \underline{G}_{d_a}^T(t_i) \qquad (2.111B)$$

The mean and covariance of the augmented vector of desired quantities, $\underset{\sim}{y}_a$, are given as a linear combination of the statistics of $\underset{\sim}{x}_a$ consistent with the definition of $\underset{\sim}{y}_a$ (Ref10):

$$\underline{m}_{y_a}(t_i) = \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{G}_{cz}(t_i)\ \underline{H}_t(t_i) & \underline{G}_{cx}(t_i) \end{bmatrix} \underline{m}_{x_a}(t_i) + \begin{bmatrix} \underline{0} \\ \underline{G}_{cy}(t_i) \end{bmatrix} \underline{y}_d(t_i) \qquad (2.112A)$$

$$\underline{P}_{y_a}(t_i) = \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{G}_{cz}(t_i)\ \underline{H}_t(t_i) & \underline{G}_{cx}(t_i) \end{bmatrix} \underline{P}_{x_a}(t_i) \begin{bmatrix} \underline{I} & \underline{H}_t^T(t_i)\ \underline{G}_{cz}^T(t_i) \\ \underline{0} & \underline{G}_{cx}^T(t_i) \end{bmatrix} + \begin{bmatrix} \underline{0} \\ \underline{G}_{cz}(t_i) \end{bmatrix} \underline{R}(t_i) \begin{bmatrix} \underline{0} & \underline{G}_{cz}^T(t_i) \end{bmatrix} \qquad (2.112B)$$

Eqs (2.111) and (2.112) will give an accurate description of the desired statistics at the sampled times t_i. However, it is often desired to know the statistics at particular moments between sample times. The differential equation for $\underset{\sim}{y}_a(t)$ is

$$\dot{\underset{\sim}{y}}_a(t) = \begin{bmatrix} \dot{\underset{\sim}{x}}_t \\ \dot{\underset{\sim}{u}}_t \end{bmatrix} = \begin{bmatrix} \underline{F}_t(t) & \underline{B}_t(t) \\ \underline{0} & \underline{0} \end{bmatrix} \begin{bmatrix} \underset{\sim}{x}_t(t) \\ \underset{\sim}{u}(t) \end{bmatrix} + \begin{bmatrix} \underline{G}_t(t) \\ \underline{0} \end{bmatrix} \underset{\sim}{w}_t(t) \qquad (2.113)$$

Based on Eq (2.113), the mean $\underline{m}_{y_a}(t)$ and covariance $\underline{P}_{y_a}(t)$ are propagated between sample times by

$$\dot{\underline{m}}_{y_a}(t) = \begin{bmatrix} \underline{F}_t(t) & \underline{B}_t(t) \\ \underline{0} & \underline{0} \end{bmatrix} \underline{m}_{y_a}(t) \qquad (2.114A)$$

$$\dot{\underline{P}}_{y_a}(t) = \begin{bmatrix} \underline{F}_t(t) & \underline{B}_t(t) \\ \underline{0} & \underline{0} \end{bmatrix} \underline{P}_{y_a}(t) + \underline{P}_{y_a} \begin{bmatrix} \underline{F}_t^T(t) & \underline{0} \\ \underline{B}_t^T(t) & \underline{0} \end{bmatrix}$$

$$+ \begin{bmatrix} \underline{G}_t(t) \underline{Q}_t(t) \underline{G}_t^T(t) & \underline{0} \\ \underline{0} & \underline{0} \end{bmatrix} \qquad (2.114B)$$

The initial conditions for the sample period beginning at time t_i, come from Eq (2.112). Note that no $\underline{P}_{x_a v_t}(t_i)$ terms appear in Eq (2.112B) as they do in the continuous-time counterpart, Eq (2.46). This is true when $\underset{\sim}{w}_d(t_i)$ and $\underset{\sim}{x}_a(t_o)$ are assumed independent of $\underset{\sim}{v}_t(t_i)$ (Ref 9).

The optimal LQG regulator must be put into the form of Eq (2.110B) to be evaluated. The optimal LQG regulator has the control law

$$\underline{u}^*(t_i) = -\underline{G}_c^*(t_i)\, \hat{\underline{x}}(t_i^+) \qquad (2.115)$$

With $\underline{y}_d = 0$ the necessary associations are, from Maybeck (Ref 10),

$$\underline{G}_{cx}(t_i) = -\underline{G}_c^*(t_i) \left[\underline{I} - \underline{K}(t_i)\, \underline{H}(t_i)\right] \qquad (2.116)$$

$$\underline{G}_{cz}(t_i) = -\underline{G}_c^*(t_i)\, \underline{K}(t_i) \qquad (2.117)$$

$$\underline{\Phi}_c(t_{i+1}, t_i) = \left[\underline{\Phi}(t_{i+1}, t_i) - \underline{B}_d(t_i)\, \underline{G}_c^*(t_i)\right] \left[\underline{I} - \underline{K}(t_i)\, \underline{H}(t_i)\right] \tag{2.118}$$

$$\underline{B}_{cz}(t_i) = \left[\underline{\Phi}(t_{i+1}, t_i) - \underline{B}_d(t_i)\, \underline{G}_c^*(t_i)\right] \cdot \underline{K}(t_i) \tag{2.119}$$

$$\underline{B}_{cy} = \underline{0} \tag{2.120}$$

$$\underline{G}_{cy} = \underline{0} \tag{2.121}$$

where the $\underline{\Phi}$, $\underline{B}_d$, $\underline{G}_c^*$ and the filter gain $\underline{K}$ are those associated with the controller design model.

Since there are minor differences between this performance analysis and the continuous-time performance analysis, only one software routine was written to accomplish both of these performance analyses. External flags set by the calling routine indicate to the performance analysis routine whether it is to perform a continuous-time or discrete-time performance analysis. Flowcharts and FORTRAN source code for the performance analysis appear in Appendices A and B respectively.

Doyle and Stein Technique in Discrete-Time Systems - 1

This section describes the first approach taken in this thesis to try to extend Doyle and Stein's (Ref 2) technique for enhancing robustness of continuous-time LQG controllers to sampled-data LQG controllers.

In this approach, the continuous-time LQG developed

using Doyle and Stein's technique is merely discretized.

This discretized controller must be put into the performance analysis format which requires values for $\underline{G}_{cx}(t_i)$, $\underline{G}_{cz}(t_i)$ and $\underline{G}_{cy}(t_i)$ in Eq (2.110A) and $\underline{\Phi}_c(t_{i+1}, t_i)$, $\underline{B}_{cz}(t_i)$ and $\underline{B}_{cy}(t_i)$ in Eq (2.110B). Since $\underline{G}_c^*(t)$ is constant, $\underline{G}_{cx}(t_i)$ $= -\underline{G}_c^*$ where the control law has become $\underline{u}(t_i) = -\underline{G}_c^* \ \underline{x}_c(t_i)$. $\underline{G}_{cz}$ is zero and since $\underline{y}_d = \underline{0}$, so is $\underline{G}_{cy}$. Then for the controller of Eq (2.23B)

$$\dot{\underline{x}}_c(t) = \underline{F}_c(t)\ \underline{x}_c(t) + \underline{B}_{cz}(t)\ \underline{z}_t(t) + \underline{B}_{cy}(t)\ \underline{y}_d(t) \quad (2.23B)$$

$\underline{x}_c(t_{i+1})$ in Eq (2.110B) becomes

$$\underline{x}_c(t_{i+1}) = \left[\underline{I} + \left(\underline{F}_c(t_i)\ \Delta t\right)\right]\underline{x}_c(t_i) + \left[\underline{B}_{cz}(t_i) \cdot \Delta t\right]\underline{z}_{ti} + \left[\underline{B}_{cy}(t_i)\ \Delta t\right]\underline{y}_d(t_i) \quad (2.122)$$

where Δt is the sample time and $\underline{I} + \left(\underline{F}_c(t_i)\ \Delta t\right)$, $\underline{B}_{cz}(t_i)\Delta t$ and $\underline{B}_{cy}(t_i)\Delta t$ are first order discrete-time approximations of $\underline{\Phi}_c(t_{i+1}, t_i)$, $\underline{B}_{cy}(t_i)$ and $\underline{B}_{cz}(t_i)$ required in Eq (2.110B) (Ref 9). Also note that a discrete-time approximation of $\underline{R}_c(t)$ is required. From Maybeck (Ref 9) an appropriate approximation, $\underline{R}(t_i)$, is

$$\underline{R}(t_i) = \underline{R}_c(t_i)\ /\Delta t \quad (2.123)$$

Recall that $\underline{F}_c$, $\underline{B}_{cy}$ and $\underline{B}_{cz}$ are defined in Eqs (2.58) through (2.60) and are described in terms of the continuous-time controller model matrices, the Kalman filter gain (which is calculated using Doyle and Stein's technique) and the deterministic controller gain.

The Doyle and Stein Technique in Discrete-Time Systems - 2

This section describes the second approach taken in this thesis to extend Doyle and Stein's (Ref 2) technique for enhancing robustness of continuous-time LQG controllers to sampled-data LQG controllers.

In this approach, a sampled-data LQG controller is used. In order to apply the Doyle and Stein technique, $\underline{G}_d \underline{Q}_d \underline{G}_d^T$ in Eq (2.99B) is replaced by $\underline{Q}_d'$. In this controller, $\underline{Q}_d'$ the assumed discrete dynamics input noise strength, is related to Q(q) of Eq (2.64) and is

$$\underline{Q}_d' = \underline{G}_d \underline{Q}_d \underline{G}_d^T + q^2 \underline{B}_c \underline{V} \underline{B}_c^T \Delta t \qquad (2.124)$$

Eq (2.124) is a format similar to using $\underline{G}\,\underline{Q}\,\underline{G}^T$ of the continuous-time system multiplied by Δt as a first order approximation to $\underline{Q}_d$ (Ref 9). Δt is the sample period of the sampled-data controller. Note that the subscript c in Eq (2.124) is to indicate that the $\underline{B}$ matrix is the continuous-time model $\underline{B}$ matrix.

Flowcharts and FORTRAN source code for the software necessary to implement this approach appear in Appendices A and B respectively.

Enhancing Robustness of Discrete-Time Systems by Directly Choosing K

The third approach to enhancing robustness of sampled-data controllers involves directly picking the Kalman filter gain $\underline{K}$. This approach is related to that of Doyle and Stein

for continuous-time systems in that a similar strategy of making the return difference mappings for a full-state feedback system and an observer-based system asymptotically equal is used. Fig (2.6a) shows the full-state feedback system while Fig (2.6b) shows the observer-based suboptimal control law configuration where $\underline{u}(t_i) = -\bar{\underline{G}}_c^* \hat{\underline{x}}(t_i^-)$ instead of the optimal control law where $\underline{u}^*(t_i) = -\underline{G}_c^* \hat{\underline{x}}(t_i^+)$. Note that the labeling in Fig (2.6) indicates that this analysis is done in the z-domain versus the s-domain for continuous-time systems (Ref 10).

The return difference of the full-state feedback and the observer-based design need to be equal in order for the observer-based design to recover the robustness properties of the full-state feedback controller. That is, $\left[\underline{\Phi}\ \bar{\underline{K}}\right]$ is to be found such that

$$\underline{\Phi}\ \bar{\underline{K}}\ \left[\underline{I} + \underline{H}\ (z\ \underline{I} - \underline{\Phi})^{-1}\ \underline{\Phi}\ \bar{\underline{K}}\right]^{-1} = \underline{B}_d \left[\underline{H}\ (z\ \underline{I} - \underline{\Phi})^{-1}\ \underline{B}_d\right]^{-1} \qquad (2.125)$$

Note that $\bar{\underline{K}}$ is the steady-state Kalman filter gain and $\underline{\Phi}$ is the state transition matrix. If $\underline{K}(q)$, parameterized as a function the scalar q as in the continuous-time case, is selected such that

$$\lim_{q \to \infty} \frac{\underline{\Phi}\ \bar{\underline{K}}}{q} = \underline{B}_d\ \underline{W} \qquad (2.126)$$

for any nonsingular m x m $\underline{W}$ then Eq (2.126) is satisfied asymptotically. $\bar{\underline{K}}$ is thus chosen as

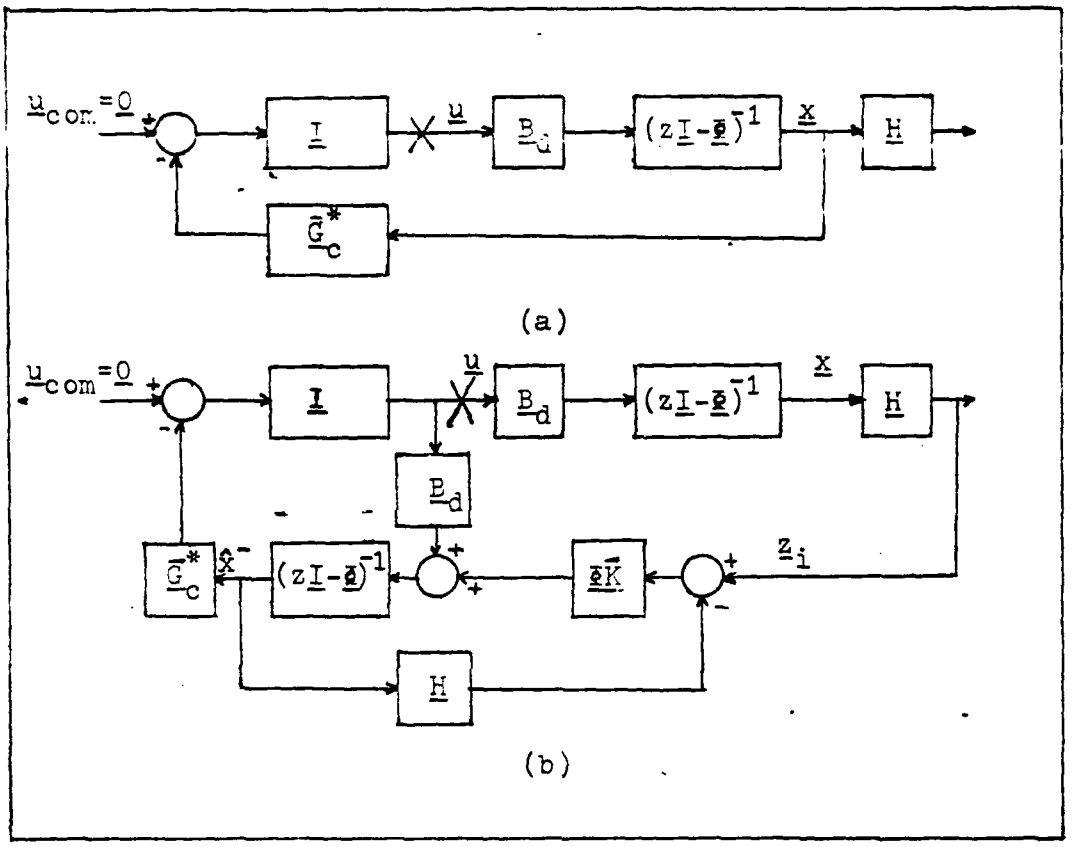


Fig 2.6 (a) Full-state feedback (b) Suboptimal control law $\underline{u}(t_i)=-\bar{\underline{G}}_c^* \hat{\underline{x}}(t_i^-)$ (Ref 10)

$$\bar{\underline{K}}= q\, \underline{\Phi}^{-1}\, \underline{B}_d\, \underline{W} \qquad (2.127)$$

Now, q and $\underline{W}$ can be varied as in the continuous-time case to achieve the desired degree of robustness. Note, in the discrete-time case, $\bar{\underline{K}}$ is chosen directly, as opposed to choosing a $\underline{Z}(s)$ in the continuous-time case (Eq (2.64)) and then calculating a $\bar{\underline{K}}$ based upon the solution to the matrix Riccati Eq (2.13) (Ref 10).

Maybeck (Ref 10) indicates there are many possible choices of $\underline{W}$ in Eq (2.127) above but that one particular

choice is motivated by considering the dual state equations. This choice, Eq (2.128),

$$\underline{W} = (\underline{H}\ \underline{\Phi}^{-1}\ \underline{B}_d)^{-1} \tag{2.128}$$

assigns m eigenvalues of the closed loop system to the origin and the remaining (n - m) eigenvalues to the invariant zeros of the system for a system of n states and m inputs.

In order to use the performance analysis algorithm, the suboptimal control law must be put into the proper format. The proper format, from Maybeck (Ref 10), is

$$\underline{G}_{cx}(t_i) = -\underline{G}_c^*(t_i) \tag{2.129A}$$

$$\underline{\Phi}_c(t_{i+1}, t_i) = \underline{\Phi}(t_{i+1}, t_i) \left[\underline{I} - \underline{K}(t_i)\ \underline{H}(t_i)\right] - \underline{B}_d(t_i)\ \underline{G}_c^*(t_i) \tag{2.129B}$$

$$\underline{B}_{cz}(t_i) = \underline{\Phi}(t_{i+1}, t_i)\ \underline{K}(t_i) \tag{2.129C}$$

$$\underline{G}_{cz}(t_i) = \underline{B}_{cy}(t_i) = \underline{G}_{cy}(t_i) = \underline{0} \tag{2.129D}$$

where $\underline{\Phi}(t_{i+1}, t_i)$, $\underline{B}_d(t_i)$, $\underline{G}_c^*(t_i)$ and the Kalman filter gain $\underline{K}(t_i)$ are those associated with the controller model.

Flowcharts and FORTRAN source code for the software necessary to accomplish this choice of $\underline{W}$ and to vary the parameter q appear in Appendices A and B respectively.

III Results and Conclusions

Introduction

This chapter discusses the results and conclusions of this study, based on data generated by the interactive computer program written to support this study (see Appendices A through E for program description). There are several items about the following discussion that need to be addressed at this point. First, the following discussion of the various controllers and performance enhancement techniques includes data obtained for the software verification models (Test Cases 1, 2 and 3) that are introduced in Appendix D. Second, the Apollo model state vector was rearranged to meet the software requirements for handling the deterministic state (see Deterministic State Augmentation Section of Chapter II). In this rearrangement the original states 1, 2, 3, 4, and 5 become states 2, 3, 4, 5, and 1 respectively, that is

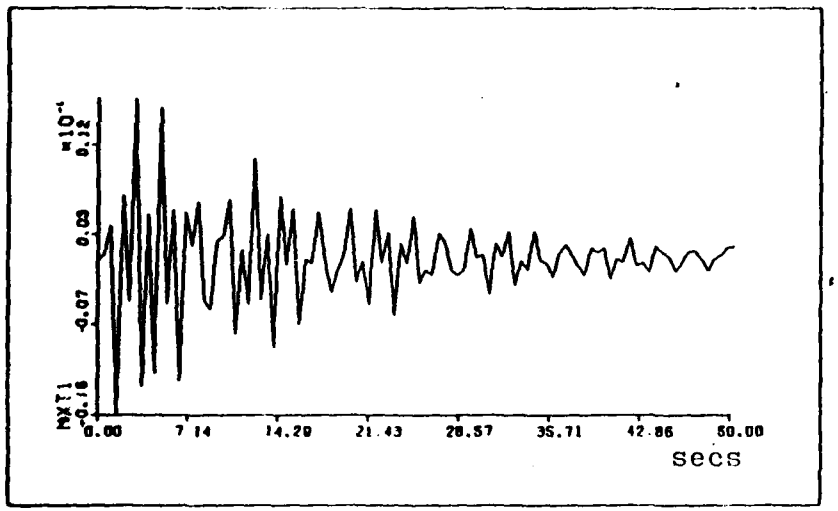
$$\underset{\sim}{x}^T = [\underset{\sim}{\delta}\ \underset{\sim}{\omega}\ \underset{\sim}{\theta}\ \underset{\sim}{v}_b\ \underset{\sim}{q}]^T \qquad (3.1)$$

Third, when discussing the results obtained for the Apollo model only states 1 and 4 ($\underset{\sim}{\delta}$ and $\underset{\sim}{v}_b$) will be used to demonstrate performance since the behavior of states 2 and 3 is similar to that of state 1 and state 5's behavior closely resembles that of state 4, which is the state most direclty affected by changes in $\underset{\sim}{u}_b$. Fourth, even though the time histories of the mean and covariance of the states and controls form the primary basis on which to judge closed-loop system performance,

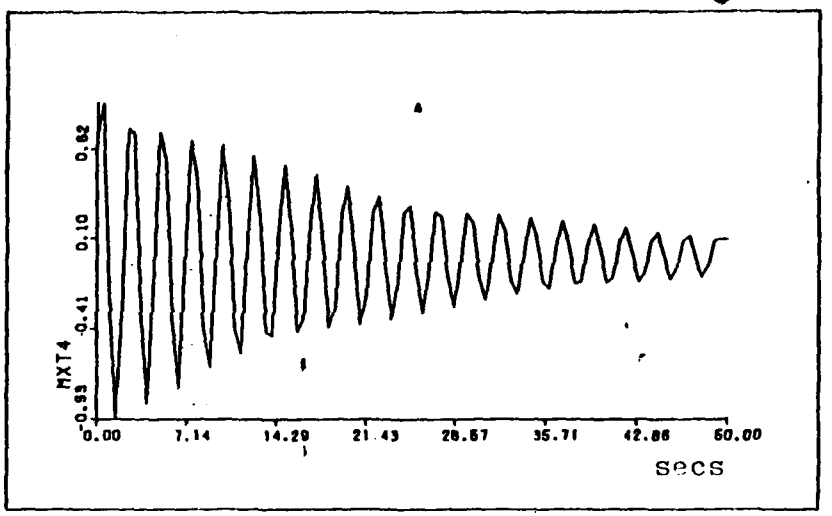
the eigenvalues of each closed-loop system are computed and examined to determine stability (prior to running the mean and covariance analysis). Last, the matrix design parameters $\underline{V}$ and $\underline{W}$ in the two performance enhancement techniques are always 1-by-1 matrices for the applications considered. They are thus always set to 1 since any desired change can be accomplished by adjusting the appropriate scalar design parameters. The order of the discussion is continuous-time controllers first, discretized continuous-time controllers second, sampled-data controllers third and finally remarks about some general trends applicable to all three-controller types.

Continuous-Time Controllers

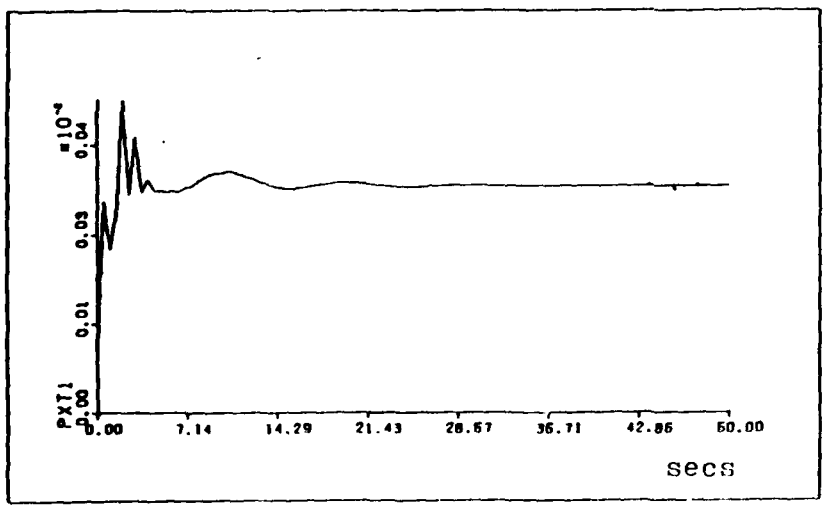
The steady state performance of the continuous-time controllers without first applying the Doyle and Stein technique for the three software verification test cases is presented in Table D.2, Appendix D. The steady state performance of the controller for the Apollo model is displayed in Figs 3.1 and 3.2 for ω_b^2 of the truth model set at 100 and 400, respectively (ω_b^2 in the controller design model is set at 100 for all cases). Figs F.1 through F.3 of Appendix F show Apollo model performance for several other values of ω_b^2. Note from these Figs that the closed-loop Apollo system is unstable for $\omega_b^2 \leq 50$ and $\omega_b^2 \geq 400$ (evident in state 4 only for this last case).



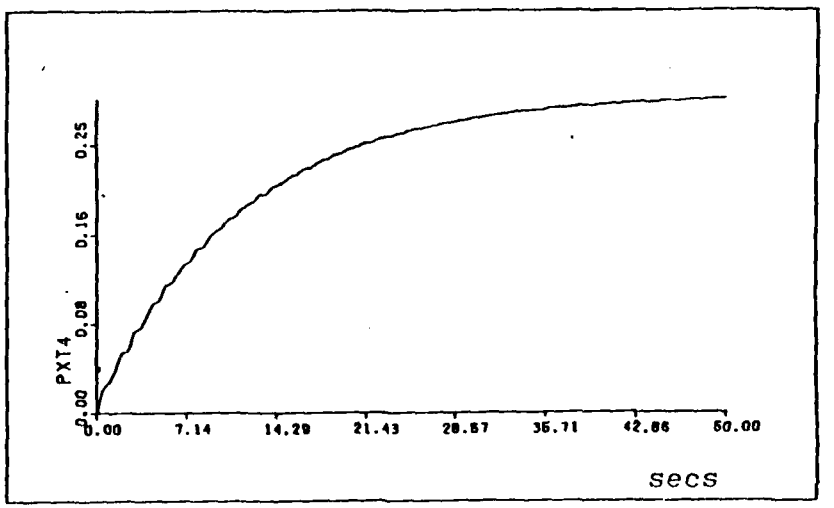
a) mean of state 1



c) mean of state 4

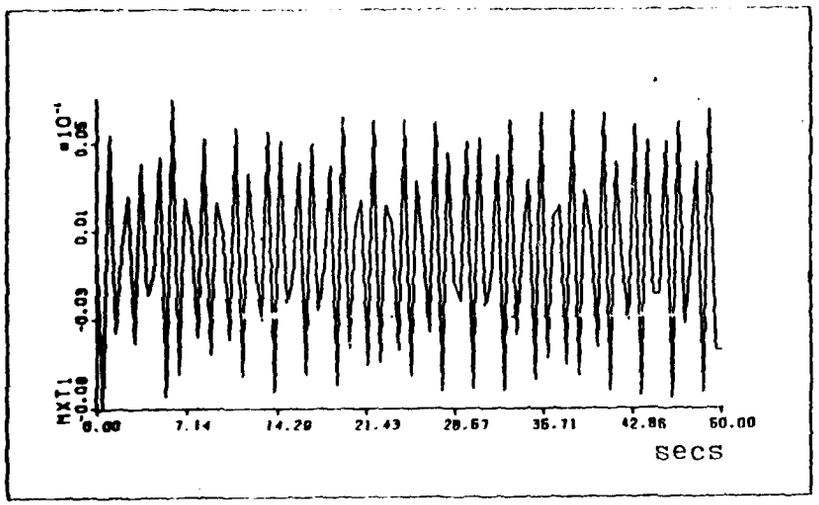


b) variance of state 1

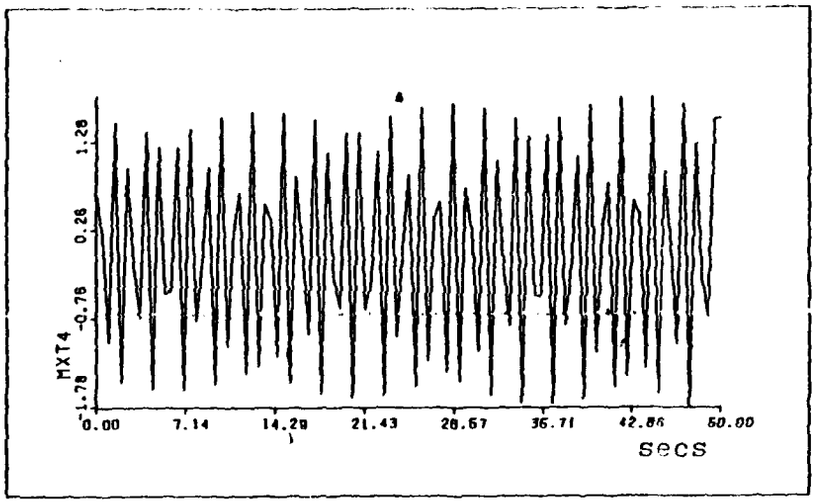


d) variance of state 4

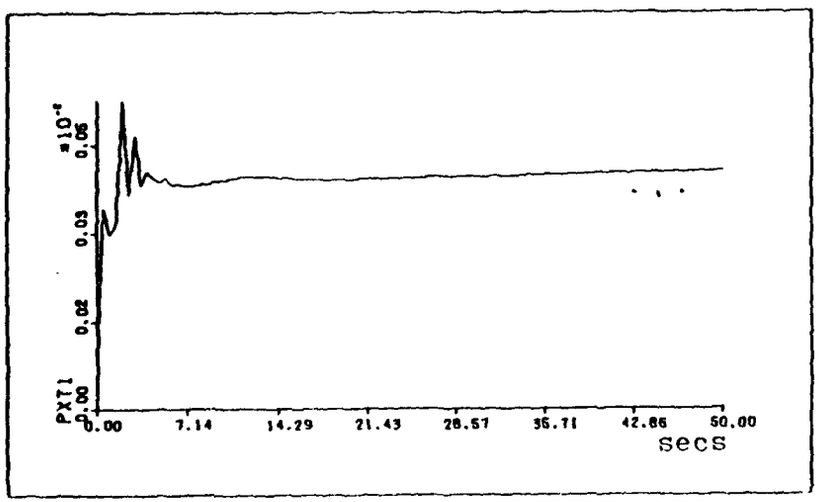
Fig 3.1 *Continuous-time performance with* ω_b^2=100 in the Apollo model



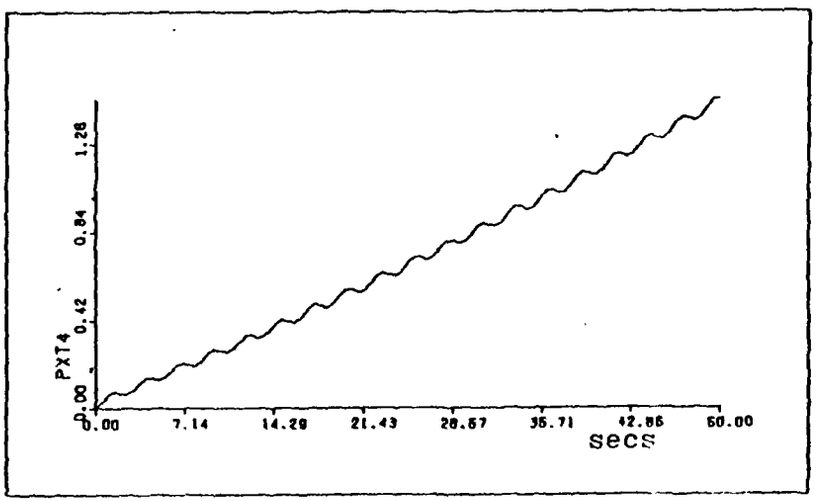
a)mean of state 1



c) mean of state 4



b) variance of state 1



d) variance of state 4

Fig 3.2 Continuous-time performance with $\omega_b^2=400$ in the Apollo model

When the Doyle and Stein technique is applied to Test Cases 1, 2, and 3, the performance is improved in the sense that the density functions are more tightly packed around the mean values (state estimates known more precisely) as q^2 (the Doyle and Stein scalar design parameter) becomes large; see Table 3.1. This is indicated by the fact that surfaces of constant likelihood, planar ellipses in the two-dimensional case, contain less area as q^2 increase where the area is directly proportional to the product of the eigenvalues of the steady-state covariance matrix, $\underline{P}$ (Ref 9). (A quick check, applicable in the two dimensional case, is to compute the value of $P_{11}P_{22} - P_{12}^2$ since this value is the magnitude of the two eigenvalues of $\underline{P}$ multiplied together. Note, in all cases presented here, the P_{12}^2 is a negligible term and thus only values for P_{11} and P_{22} are given in the tables that follow.) By examining Table 3.1, Doyle and Stein's claim that their method moves some of the filter poles toward stable plant zeros and the rest to $-\infty$ (asymptotically as q^2 becomes larger) can be verified.

Note that the Apollo model used in Table 3.1 includes a damping factor, ζ, of 0.001 in the bending mode dynamics (i.e., the 3, 3 term in the $\underline{F}$ matrix of Eq (2.75) is no longer 0 but becomes $-2\zeta\omega_b$). A damping factor was added because no noticeable performance improvement could be obtained with $\zeta = 0$ (which places a set of poles for the bending mode dynamics on the imaginary axis) and it was anticipated that moving the

Table 3.1

Steady-State Performance of Continuous Time Controllers
With Doyle and Stein Technique Applied

Test Case or Model	q_c^2	Steady-State $P_{x_t x_t}$	Steady-State P_{uu}	Filter Poles	System Zeros
1	.01	1.79	.105	-.960	none
1	1	1.78	.119	-1.00	none
1	100	1.59	.896	-3.04	none
1	1000	1.47	10.6	-28.8	none
1	$(1000)^2$	1.46	108	-288	none
2	.01	$4.50(10)^{-5}$, $6.09(10)^{3}$	2.28	-111, -111	none
2	1	$3.15(10)^{-5}$, $5.13(10)^{-3}$	4.13	-36.7, -36.7	none
2	100	$2.00(10)^{-5}$, $4.96(10)^{-3}$	57.0	-22.8, -22.8	none
3	.1	221, 2068	$4.06(10)^{4}$	-7.02, -7.02	-2
3	1	221, 2065	$4.06(10)^{4}$	-7.04, -7.04	-2
3	100	235, 1812	$5.90(10)^{4}$	-13.1, -4.30	-2
3	1000	316, 1192	$7.40(10)^{4}$	-100 , -2.10	-2
Apollo	0	$4.04(10)^{-4}$, .621	$4.60(10)^{4}$	-37.1±j37.8, -10 -6.65±j6.62	-.0044±j6.62
Apollo	1000	$3.8(10)^{-4}$, .410	.451	-10,-.02,-1.12, $-3.6(10)^{4}$,14.1	-.0044±j6.62
Apollo	$(1000)^2$	$(10)^{53}$, $(10)^{53}$	$(10)^{72}$	-1119,-560±j969, -.0044±j6.62	-.0044±j6.02

Table 3.1 (Cont)

Test Case or Model	q^2	Steady-State		Filter Poles	System Zeros
		$P_{x_t x_t}$	P_{uu}		
1[b]	0	∞	∞	-1.21	none
1[b]	1	∞	∞	-1.00	none
1[b]	100	12.1	2.43	-3.1	none
1[b]	$(1000)^2$	7.32	109.5	-288	none
3[b]	0	∞	∞	-7.02, -7.02	-2.0
3[b]	100	∞	∞	-13.1, -4.32	-2.0
3[b]	$(100)^2$	1025, 553	$4.99(10)^6$	-100 , -2.10	-2.0
3[b]	$(1000)^2$	1533, 262	$2.11(10)^6$	-1000, -2.00	-2.0

a) P_{11} and P_{22} only are given for Test Cases 2 and 3, P_{11}, P_{44} only for Apollo.

b) In these cases, the truth model was deliberately mismatched with the controller design model to produce an initially ($q^2 = 0$) unstable controller. For Test Case 1, $\underline{F}_t = [0.1]$. Test Case 3, $F_{t_{2,2}} = -100$. Test Case 2 could not be stabilized using the Doyle and Stein procedure once a destabilizing $\underline{F}$ was chosen.

c) q^2 is the Doyle and Stein scalar design parameter.

poles away from the imaginary axis might allow the Doyle and Stein technique some extra "maneuvering room" in which to bring about steady state performance improvement. Several damping factors between 0.001 and 0.15 were tried to see if steady state performance could be improved but no noticeable improvement was obtained for any value tried. (No values larger than 0.15 were tried since $\zeta = 0.15$ is already 10 times larger than that in the actual Apollo system.)

Table 3.1 shows only marginal improvements in closed-loop stability for the Apollo model as q^2 increases until some critical value is reached (approximately, $q^2 \approx (701)^2$) at which closed-loop stability is lost. Note that one of the filter poles is positive for $q^2 \leq (701)^2$, the stable closed-loop system case, and that the filter poles do not migrate as Doyle and Stein suggest but abruptly, at $q^2 > (701)^2$, change to the configuration where some are co-located with system zeros and the rest have larger negative real parts. At this point closed-loop system stability is lost. It is noted that in the case of the Apollo model, the Doyle and Stein techniques adds white noise in the system before the first order lag (where noise did not previously enter) instead of after it and this may affect the resulting performance of this technique. This phenomenon should be investigated further.

Table 3.2 presents the results for Test Cases 2 and 3 when the strength of the continuous-time noise $\underline{Q}$ is "tuned" by adding a $\underline{\Delta Q}$ which is a simple scalar multiple of Q. ΔQ is

Table 3.2

Steady-State Performance of
Continuous-Time Controllers with Tuning
of $\underline{Q}$ by Adding $\underline{\Delta Q}$

Test Case	Q, ΔQ	Steady-State		Filter Poles
		$P_{x_t x_t}$	P_{uu}	
2	10, 1	$4.4(10)^{-5}$, $6.1(10)^{-3}$	2.3	-9.6±j9.1
2	10, 100	$2.9(10)^{-5}$, $5.0(10)^{-3}$	5.0	-40±j40
2	10, 1000	$1.9(10)^{-5}$, $4.7(10)^{-3}$	79.6	-125±j125
3	1, .1	219 , 2091	$4.3(10)^{4}$	-7.25±j1.7
3	1, 1	210 , 2229	$6.5(10)^{4}$	-11.3, -6.6
3	1, 100	179 , 2784	$6.7(10)^{5}$	-90, -5.9
3	1, 1000	173 , 2914	$7(10)^{6}$	-9000, -5.8
3[b]	1, 10	∞ ∞	∞	-28.0, -5.99
3[b]	1, 10000	∞ ∞	∞	-900, -5.90
3[b]	1, $(10)^6$	∞ ∞	∞	-9000, -5.90

a) Only diagonal terms P_{11}, P_{22} shown.

b) Truth model $\underline{F}$ matrix deliberately mismatched with controller design model; $F_{t_{2,2}} = -100$. Test Case 2 not shown; it also could not be stabilized using this tuning procedure.

chosen such that it approximates the value of noise added by the Doyle and Stein method. Tuning Q in this manner adds noise at the points of entry of $\underline{w}$ into the system dynamics instead of at the point where $\underline{u}$ enters as in the Doyle and Stein technique. (Test Case 1 is not displayed because there is no physical difference between the Doyle and Stein technique and this tuning procedure for this case.) Comparing the results for Test Cases 2 and 3 shows that there is not much improvement from one method to the other for Test Case 2, but that there is a noticeable difference for Test Case 3. Note, when the closed-loop system of Test Case 2 was made unstable (by changing entries for the $\underline{F}$ matrix) in order to show robustness effects, that neither procedure (Doyle and Stein or adding ΔQ) could re-stabilize the closed-loop system. For Test Case 3, the Doyle and Stein technique gives better results in the sense that the surfaces of constant likelihood, contain less area indicating that density functions are more tightly packed around the mean values. This approach was not applied to the Apollo model since the results of the Doyle and Stein approach were poor.

As far as the Test Cases are concerned, the Doyle and Stein technique appears to be a valid approach to improving the performance of continuous-time systems even for values of q^2 less than ∞. For Test Case 3 this technique is superior to tuning the $\underline{Q}$ matrix by adding a $\underline{\Delta Q}$ as a simple scalar multiple of $\underline{Q}$. In this case, the tuning procedure adds noise in both

state equations, but the Doyle and Stein technique adds noise only to the second state equation, where the control input exists. In contrast, either procedure adds noise at the control input for Test Case 2 due to configuration of the B and G matrices in the test case. As for the more complex Apollo model, the results are inconclusive. While some marginal improvement could be seen for large q^2 and $\omega_b^2 = 400$, the total behavior of the system was not as expected. In particular, the behavior of the filter poles, as discussed previously, was not expected, especially since when the poles were in the desired positions, closed-loop stability was lost.

Discretized Continuous-Time Controller

The results of discretizing each controller without first applying the Doyle and Stein technique are presented in Table D.2 of Appendix D. Examination of the entries in this table reveal that the discretization process gives better results for smaller sample periods, as expected since the first order approximations used are valid for small (relative to characteristic times of the basic system) sample periods (see the Doyle and Stein Technique in Discrete-Time Systems - 1 in Chapter II). Sample periods chosen for the robust versions of these controllers were those for which the unmodified controllers had essentially the same steady-state performance as their continuous-time counterparts. Table 3.3 presents the results for the discretized controllers for Test Cases 1, 2,

Table 3.3

Steady-State Performance of
Discretized Robust Controllers

Test Case	Δt	q^2 b	Steady State $P_{x_t x_t}$ a	P_{uu}
1	.01	1	1.78	.120
1	.01	10000	1.48	12.4
1	.01	$(500)^2$	1.47	194
1	.01	$(1000)^2$	∞	∞
2	.001	100	$2.01(10)^{-5}$, $4.79(10)^{-3}$	71.5
2	.001	$(25)^2$	$1.86(10)^{-5}$, $5.55(10)^{-3}$	305
2	.001	$(49)^2$	$1.77(10)^{-5}$, $6.78(10)^{-3}$	953
2	.001	$(50)^2$	∞	∞
3	.001	1	221 , 2122	$4.7(10)^4$
3	.001	100	235 , 1814	$2.9(10)^4$
3	.001	$(100)^2$	316 , 1192	$7.3(10)^4$
3	.001	$(1000)^2$	∞	∞

Table 3.3 (Cont)

Test Case	t	q^2 b	Steady-State $P_{x_t x_t}$ a	Steady-State P_{uu}
1[c]	.001	0	∞	∞
1[c]	.001	1	∞	∞
1[c]	.001	100	12.0	2.44
1[c]	.001	$(1000)^2$	7.33	128
3[c]	.001	0	∞	∞
3[c]	.001	100	∞	∞
3[c]	.001	10000	1025, 554	$4.99(10)^6$
3[c]	.001	$(1000)^2$	1533, 262	$2.16(10)^6$

a) only diagonal terms P_{11}, P_{22} shown.

b) q^2 is the Doyle and Stein scalar design parameter.

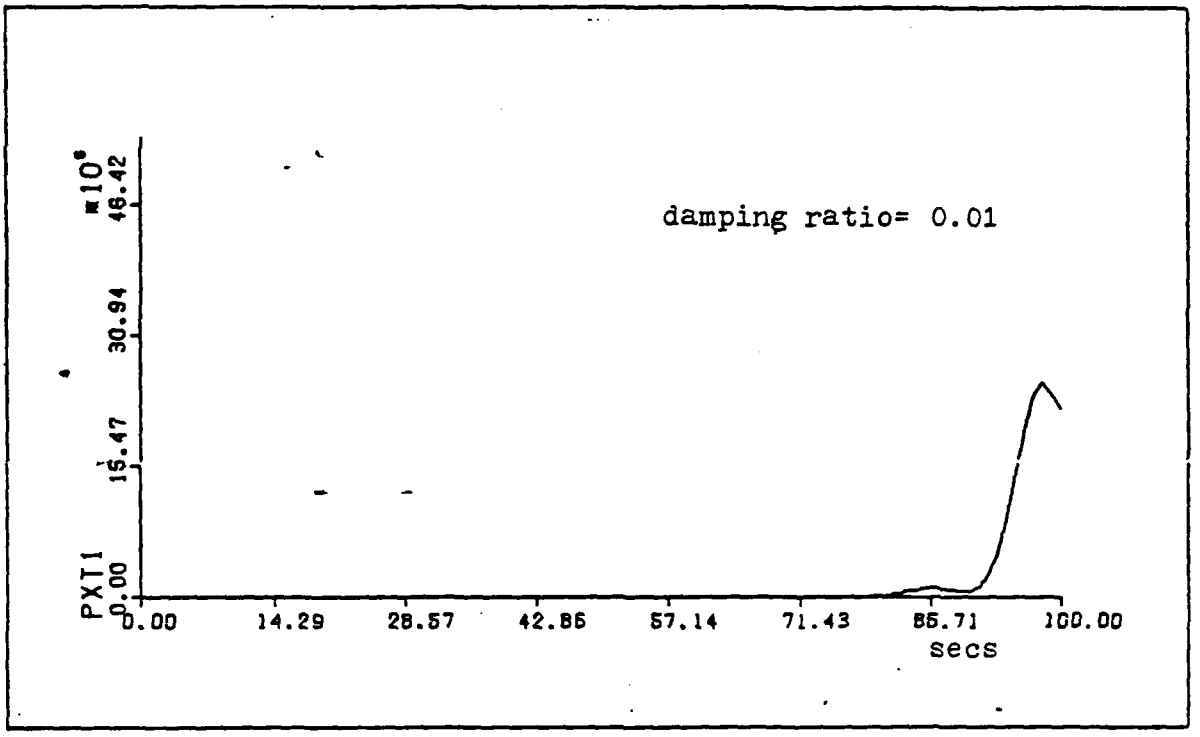
c) $\underline{F}$ matrices changed in truth model description to show robustness. For Test Case 1, $F_t = 0.1$; for Test Case 3 $F_{t_{2,2}} = -100$. Test Case 2 could not be stabilized once a destabilizing $\underline{F}$ was chosen.

and 3 when the Doyle and Stein technique is applied. Comparing the results for Test Cases 1, 2, and 3 shown in Table D.2 and 3.3 shows that discretizing a continuous-time controller to which the Doyle and Stein technique has been applied produces a discrete-time controller with enhanced performance up to a certain value of q^2.

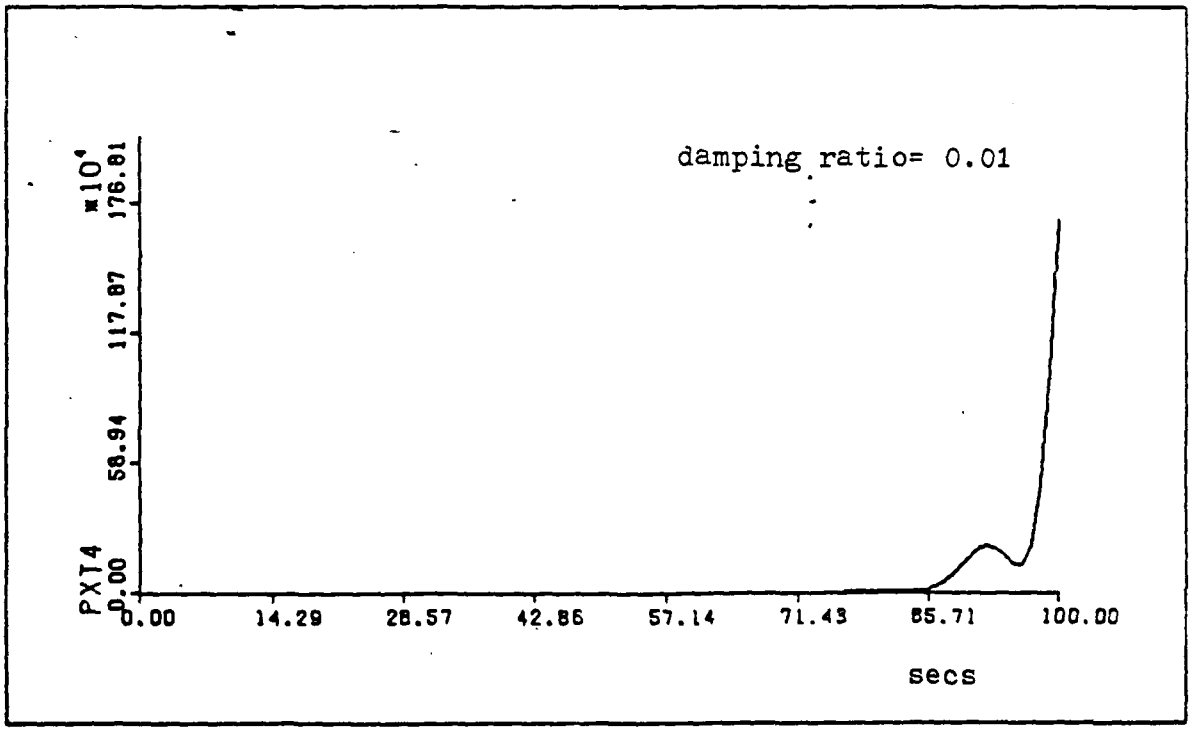
The Apollo model presented difficulties in this case also. In fact, no matter how small the sample time was made (periods down to 0.001 sec were investigated), the resulting closed-loop system was unstable indicating that the first order approximations were invalid. To improve the situation, a damping factor was again added in bending mode dynamics as in the continuous-time case. The initial choise of $\zeta = 0.015$ and sample periods ≤ 0.01 sec allowed the resulting discretized continuous-time controllers to remain stable even though ω_b^2 was varied passed 400 which was the desired value about which to investigate performance. By trial and error and noticing trends in the data (the interactive computer program was well suited to this task) a suitable combination of sample period and damping ratio was found. With a $\zeta = 0.01$ and sample period of 0.01 sec, a discretized continuous-time controller that was initially unstable at $\omega_b^2 = 400$ could be made stable by applying the Doyle and Stein technique before discretizing. Values of q^2 between $(100)^2$ and $(0.001)^2$ were tried and $q^2 = (0.02)^2$ selected as the value that gave the best performance. In fact stable closed-loop operation occurs only for

$(0.005)^2 \leq q^2 \leq (0.05)^2$. The fact that there is an upper limit on the value for q^2 is an aspect that is not seen in the continuous-time case. Fig 3.3 shows the performance of the unstable unmodified discretized continuous-time controller. Figs 3.4 and 3.5a show the performance of the discretized controller after the Doyle and Stein technique is applied where $q^2 = (0.1)^2$. For $q^2 = (0.02)^2$ the results are shown in Figs 3.5b and 3.6. It is quite evident from these Figs that applying the Doyle and Stein technique can cause dramatic improvements in performance.

Thus, applying the Doyle and Stein to a continuous-time controller and then discretizing the result to produce a robust discrete-time controller, appears to be one acceptable approach to extending the Doyle and Stein technique to discrete-time controllers. Note however, that since the discretization process requires appropriately small sample-times, it is anticipated that there will be some cases where this technique cannot be applied readily. The Apollo model is a good example of this since in the actual Apollo system the sample period is 0.1 secs which is 10 times slower than that which produces a stable closed-loop system. One other item to note at this point is that the interactive computer program is well suited to any tuning (trial and error) that may be required. In fact, the author accomplished the tuning described above in several hours of interactive computer-time, whereas it probably would have taken several days using batch processing.

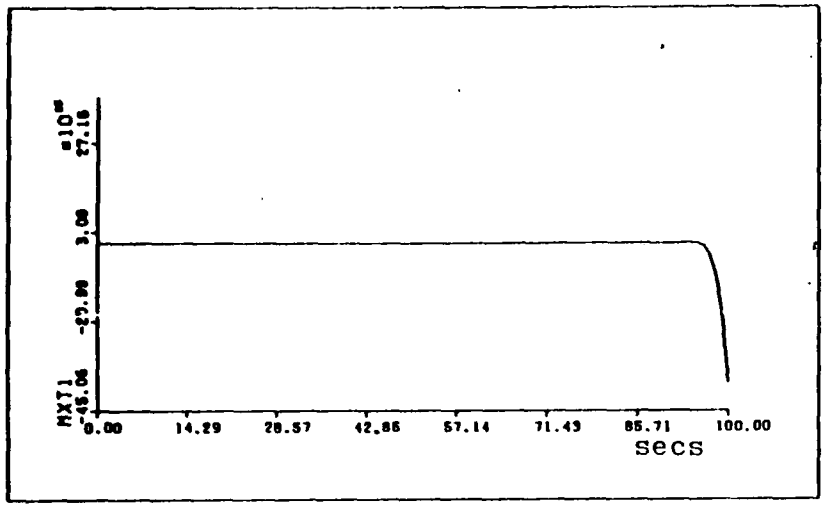


a) variance of state 1

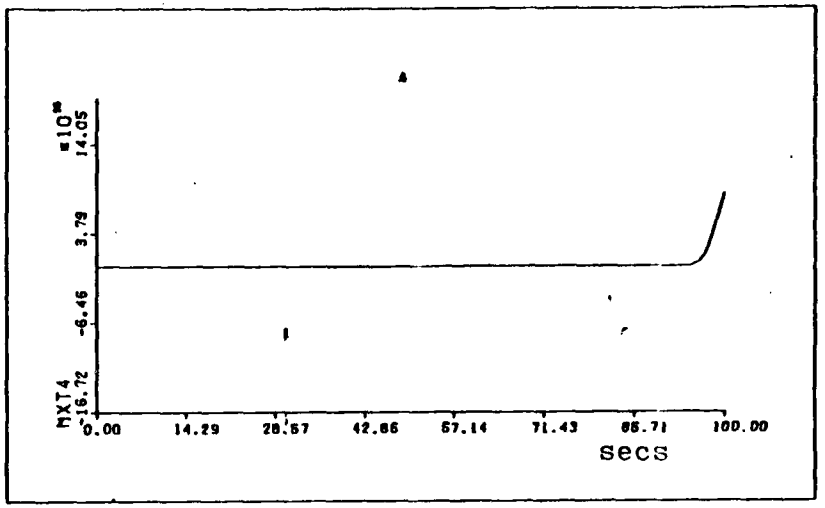


b) variance of state 4

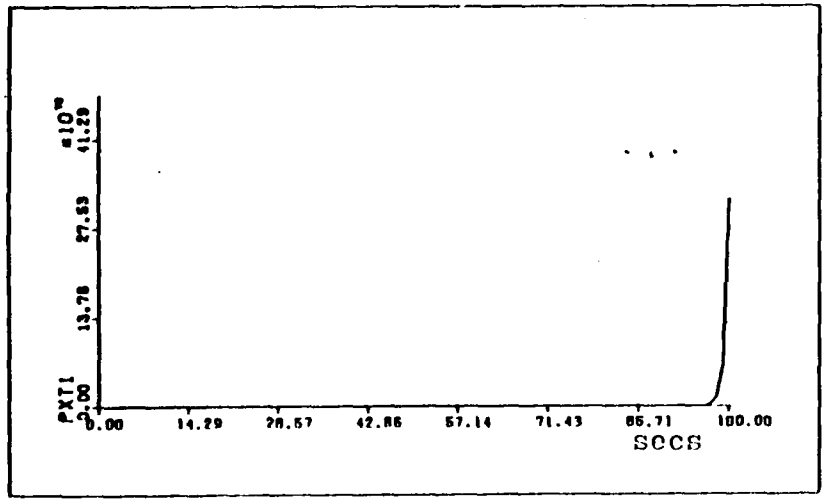
Fig 3.3 Discretized continuous-time performance with ω_b^2=400 in the Apollo model.



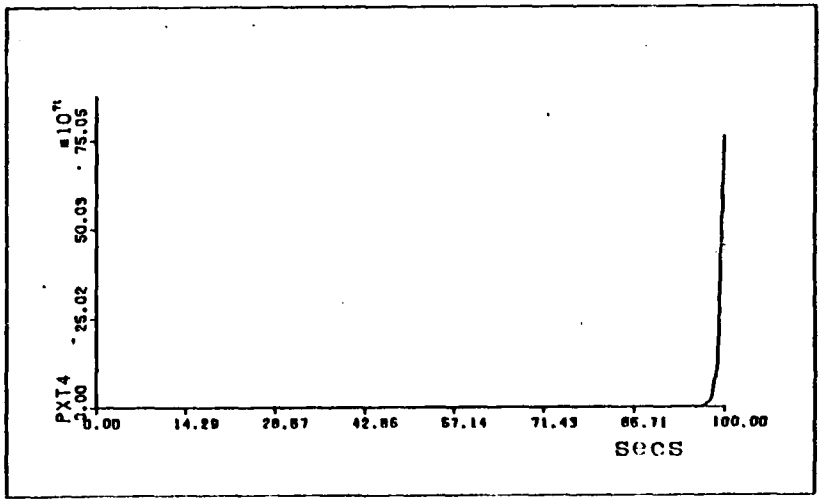
a) mean of state 1



c) mean of state 4

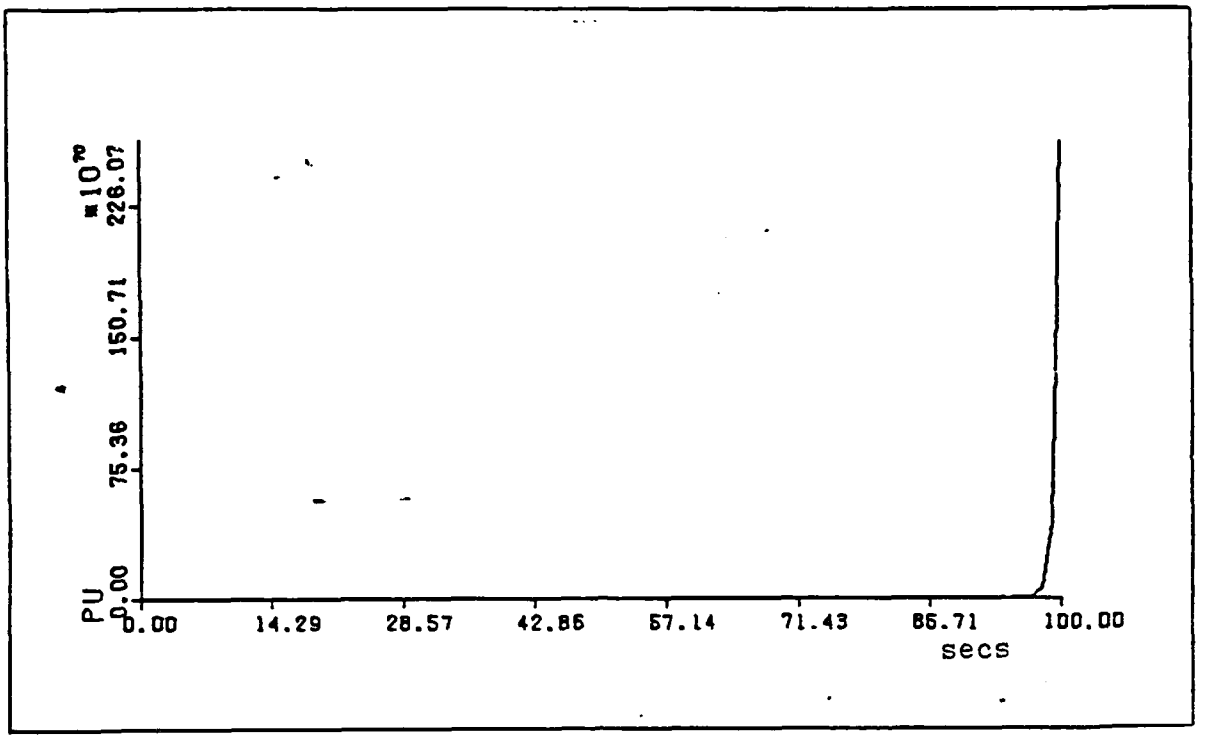


b) variance of state 1

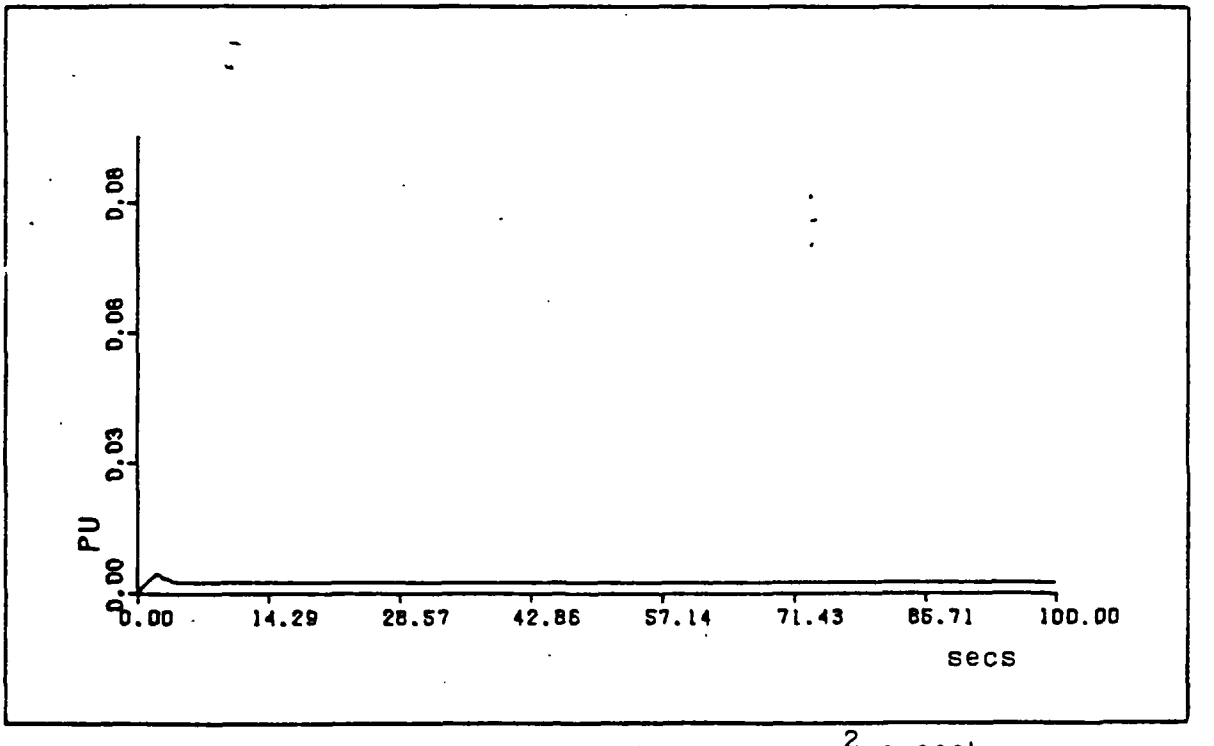


d) variance of state 4

Fig 3.4 Discretized continuous-time performance with $\omega_b^2=400$ and $q^2=0.01$

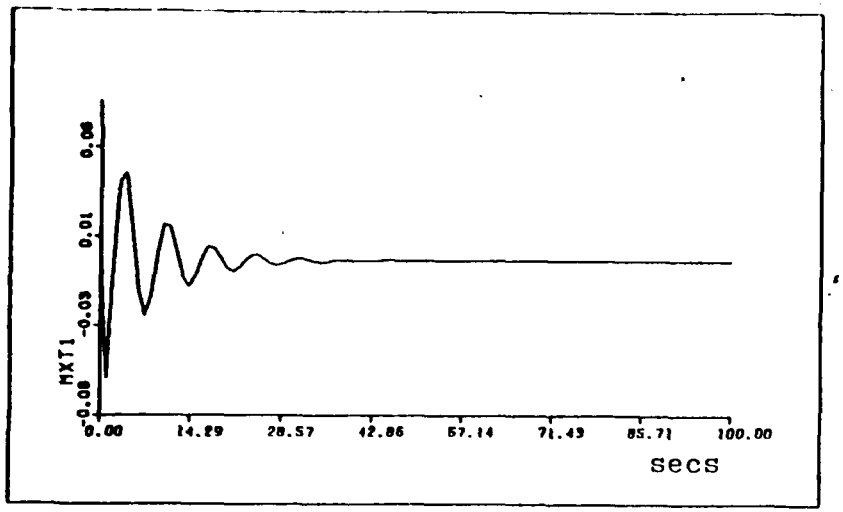


a) control variance when $q^2=0.01$

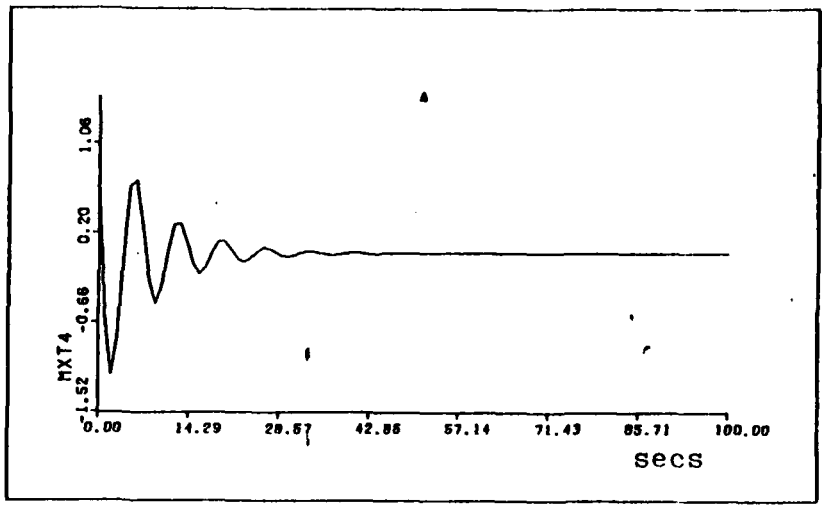


b) control variance when $q^2=0.0004$

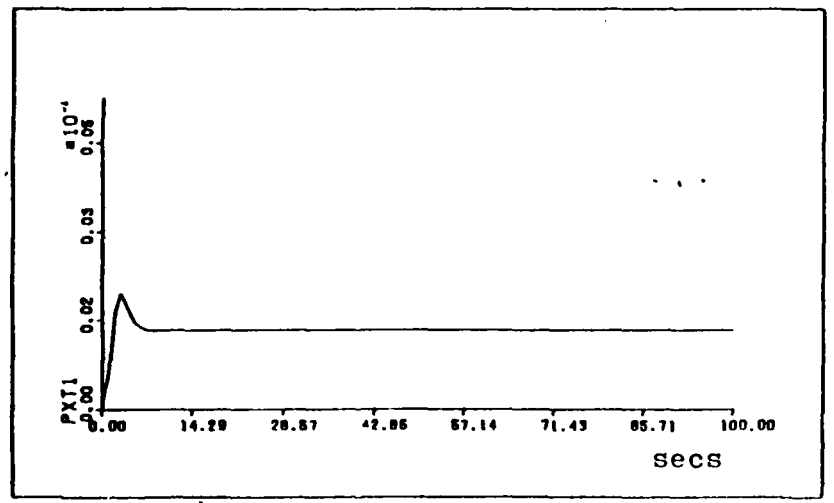
Fig 3.5 Discretized continuous-time performance with $\omega_b^2=400$ and $q^2=0.01$ and 0.0004 (control variance)



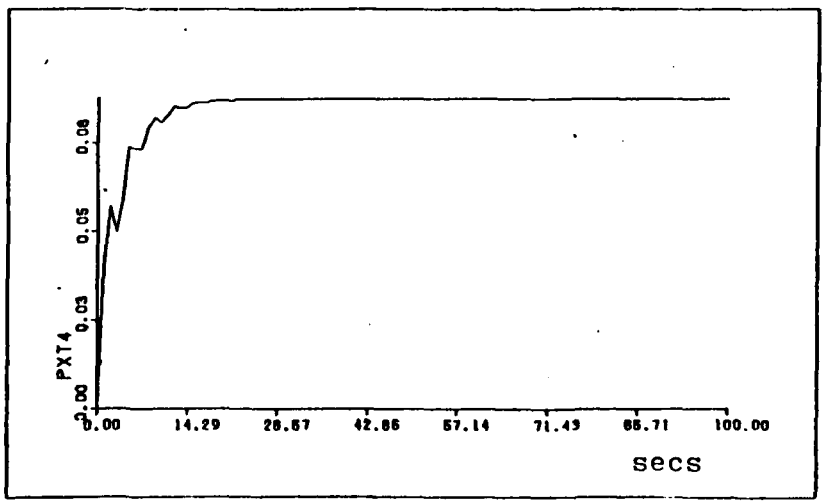
a) mean of state 1



c) mean of state 4



b) variance of state 1



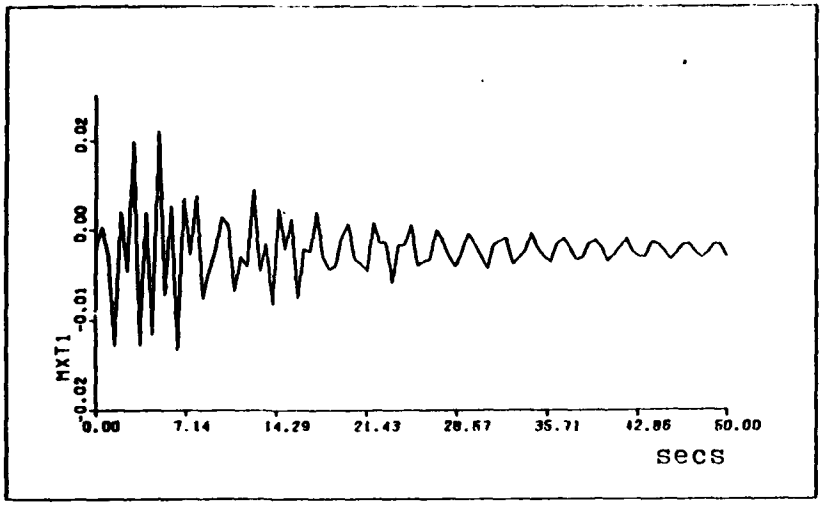
d) variance of state 4

Fig 3.6 Discretized continuous-time performance with ω_b^2=400 and q^2=0.0004

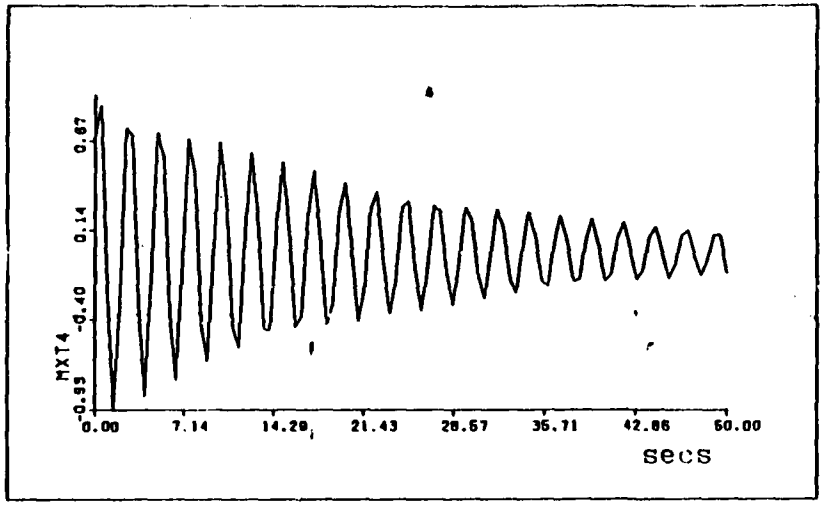
Sampled-Data Controllers

This section presents the results and conclusions concerning the performance of the robust sampled-data controllers developed in this thesis. A discussion in the effect of applying the Doyle and Stein technique (as extended in the Enhancing Robustness of Discrete-Time Controllers - 2 section of Chapter II) appears first, followed by comparison with the effects of tuning the $\underline{Q}$ matrix (by adding a $\Delta\underline{Q}$ as in the continuous-time discussion). Last there is a discussion of the results for picking the Kalman filter gain, $\underline{K}$, directly (see appropriate section of Chapter II).

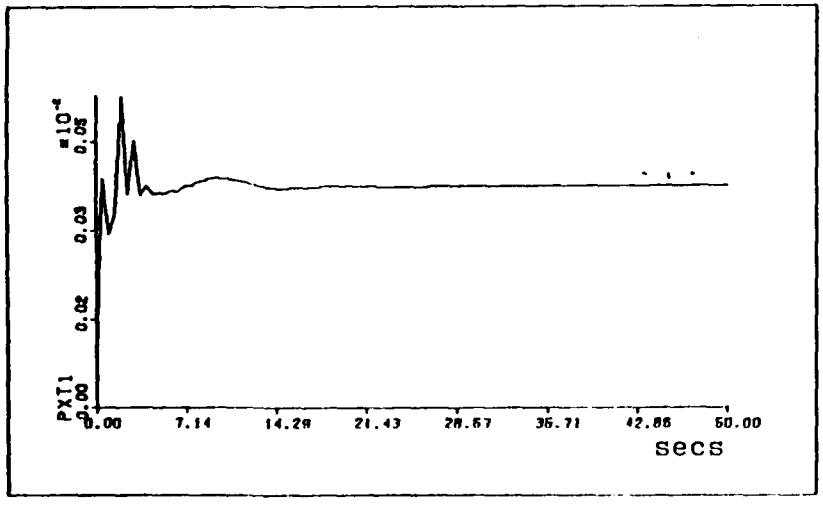
Doyle and Stein Techniques Extended to Sampled-Data Controllers. The steady-state performance of the sampled-data controllers for Test Cases 1, 2, and 3, with no robustness enhancement, is shown in Table D.3 of Appendix D. Note here that, as for discretized continuous-time controllers, the sample period can greatly affect the steady-state performance. The performance of the sampled-data controller for the Apollo model, with no robustness enhancement and for several different values of ω_b^2, is presented in Figs 3.7 and 3.8 and Figs F.4 through F.6 of Appendix F. As in the continuous-time case, the closed-loop system is unstable for $\omega_b^2 \leq 50$ (although this is not shown in the above Figs). In the sampled-data case instability also occurs when $\omega_b^2 \geq 200$. (Note 0.1 secs is the chosen sample period for the Apollo model since this value gives good performance and is the same as that used in the actual Apollo controller).



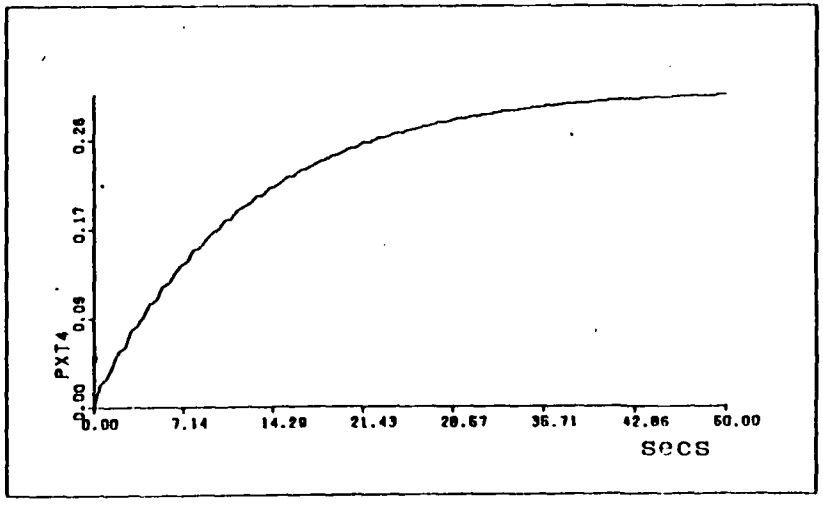
a) mean of state 1



c) mean of state 4

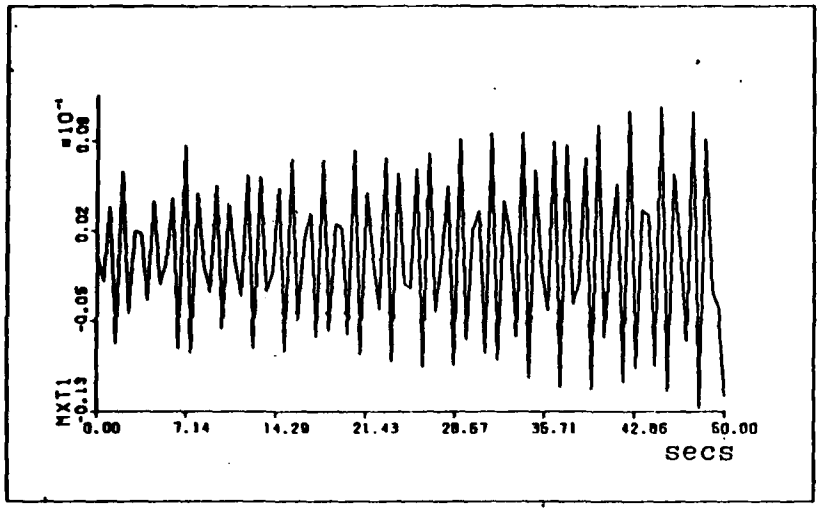


b) variance of state 1

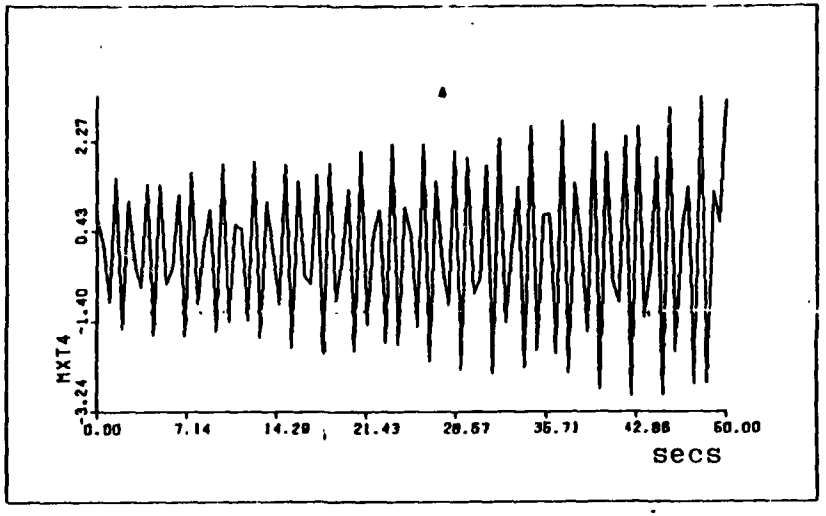


d) variance of state 4

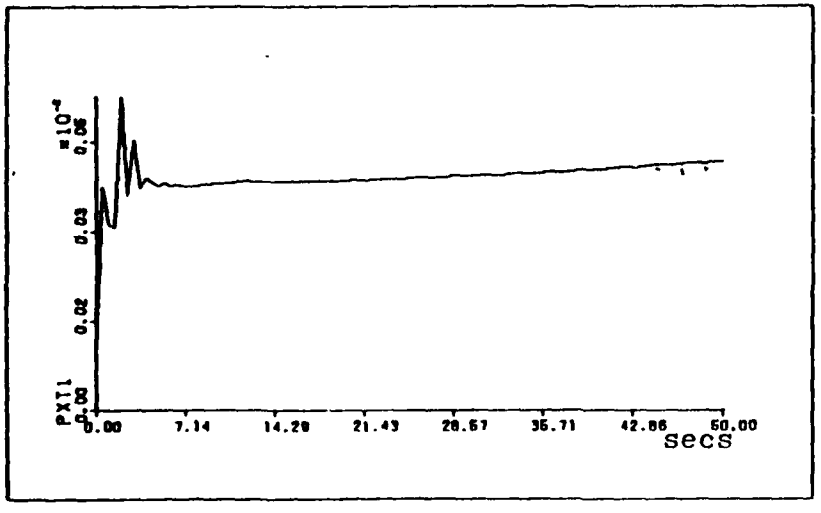
Fig 3.7 Sampled-data performance with ω_b^2=100 in the Apollo model



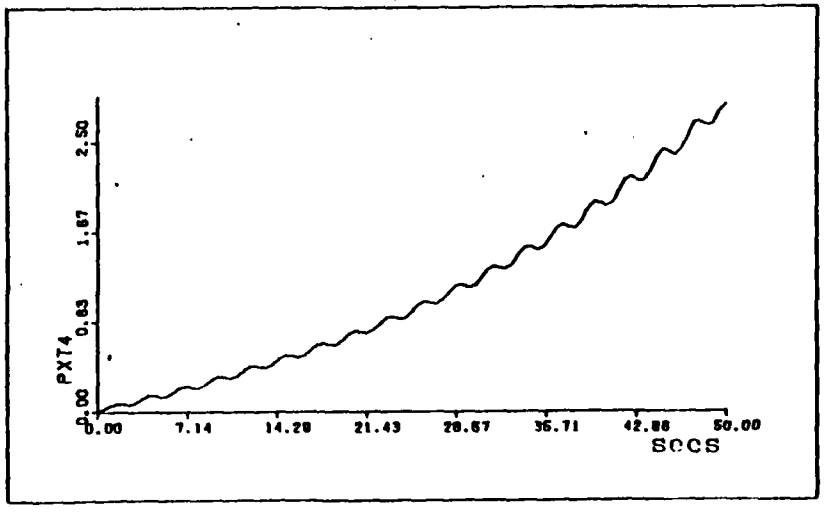
a) mean of state 1



c) mean of state 4



b) variance of state 1



d) variance of state 4

Fig 3.8 Sampled-data performance with ω_b^2=400 in the Apollo model

The steady state closed-loop performance with the Doyle and Stein technique applied is presented in Table 3.4 for Test Cases 1, 2, and 3 and in Figs 3.9, 3.10, and F.7 through F.10 for the Apollo model. Comparison of the results presented in Tables 3.3 and 3.4 shows, for Test Cases 1, 2, and 3, that the steady state performance is enhanced as q^2 increases in the sense that areas of the ellipse representing the surfaces of constant likelihood become smaller (see related discussion of these surfaces in the Continuous-Time Controller section of this chapter). For the Apollo model the results are similar. Figs 3.9 and 3.10 show stable closed-loop operation for $\omega_b^2 = 400$ and 700, respectively, while Figs 3.7, 3.8 and F.6 indicate that the closed-loop system with an unmodified controller is unstable for $\omega_b^2 > 300$. By examining Figs 3.9, F.7 and F.8, one can see that there is a value of q^2 (in this case between $(0.1)^2$ and 1.0) beyond which increases in q^2 produce no noticeable difference.

As in the continuous-time case, a comparison is made between the Doyle and Stein technique benefits and those benefits obtained by tuning the $\underline{Q}$ matrix by adding a scalar multiple of $\underline{Q}$, $\underline{\Delta Q}$. Also as in the continuous-time case ΔQ is chosen so as to approximate values of q^2 used. When this technique is applied to Test Cases 1, 2, and 3, the performance could be made similar to that obtained by applying the Doyle and Stein technique as shown by comparing the results in Tables 3.4 and 3.5. In contrast, Figs 3.11 and 3.12 show the performance

Table 3.4

Steady-State Performance of Sampled-Data Controllers with Doyle and Stein Technique Applied

Test Case	Δt	q^2 [b]	Steady State	
			$P_{x_t x_t}$ [a]	P_{uu}
1	.01	.01	1.79	.105
1	.01	1	1.78	.119
1	.01	100	1.58	.900
1	.01	$(1000)^2$	1.46	61.5
2	.01	.01	$4.00(10)^{-5}$, $6.0(10)^{-3}$	2.28
2	.01	1	$2.70(10)^{-5}$, $5.1(10)^{-3}$	5.11
2	.01	100	$1.75(10)^{-5}$, $5.4(10)^{-3}$	51.8
2	.01	10000	$1.65(10)^{-5}$, $6.4(10)^{-3}$	110
2	.01	$(1000)^2$	$1.65(10)^{-5}$, $6.4(10)^{-3}$	113
3	.01	.01	341 , 1108	655
3	.01	1	341 , 1108	652
3	.01	100	349 , 1078	473
3	.01	10000	316 , 1193	774

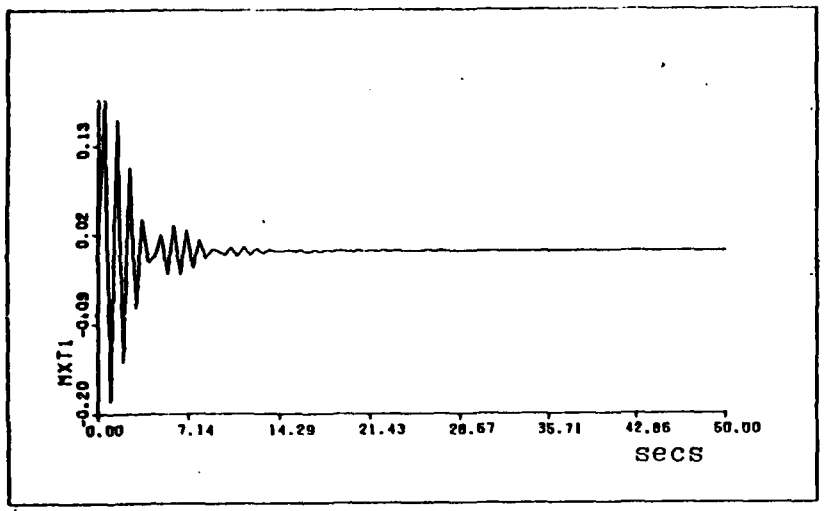
Table 3.4 (Cont)

Test Case	Δt	q^2 b	Steady-State $P_{x_t x_t}$ a	Steady-State P_{uu}
1[c]	.01	0	∞	∞
1[c]	.01	1	∞	∞
1[c]	.01	100	12.0	2.43
1[c]	.01	$(1000)^2$	7.35	63.0
3[c]	.01	0	∞	∞
3[c]	.01	100	∞	∞
3[c]	.01	$(100)^2$	996, 551	$4.96(10)^6$
3[c]	.01	$(1000)^2$	1554, 257	$2.03(10)^6$

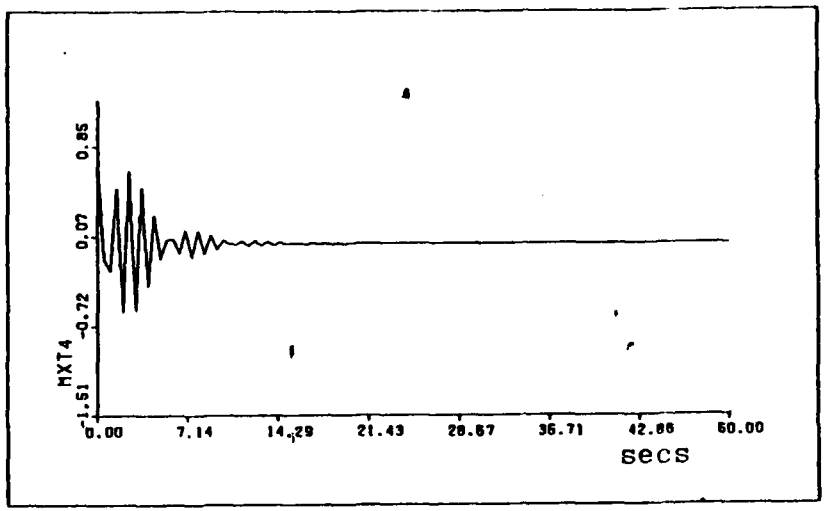
a) only the diagonal terms P_{11}, P_{22} are shown

b) q^2 is the Doyle and Stein scalar design parameter

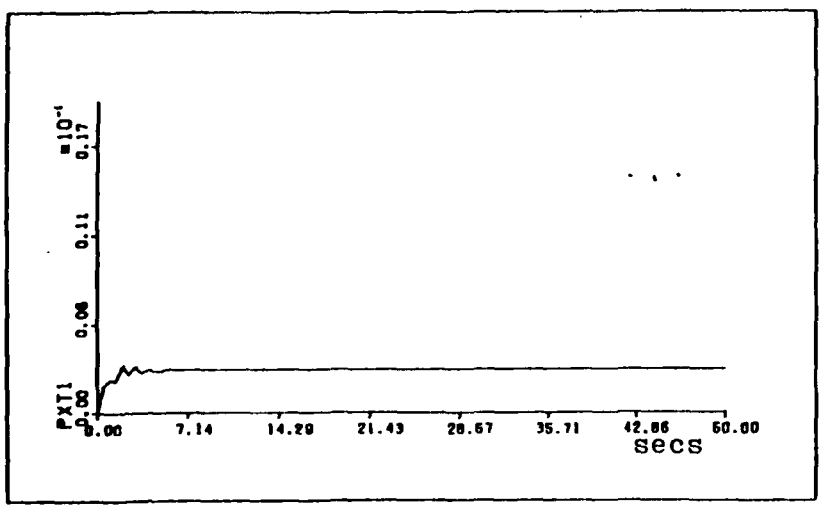
c) $\underline{F}$ matrices changed in truth model descriptions to show robustness. For Test Case 1, $F_t = .1$; for Test Case 3, $F_{t_{2,2}} = -100$. Test Case 2 chould not be stabilized once a destabilizing $\underline{F}$ was chosen.



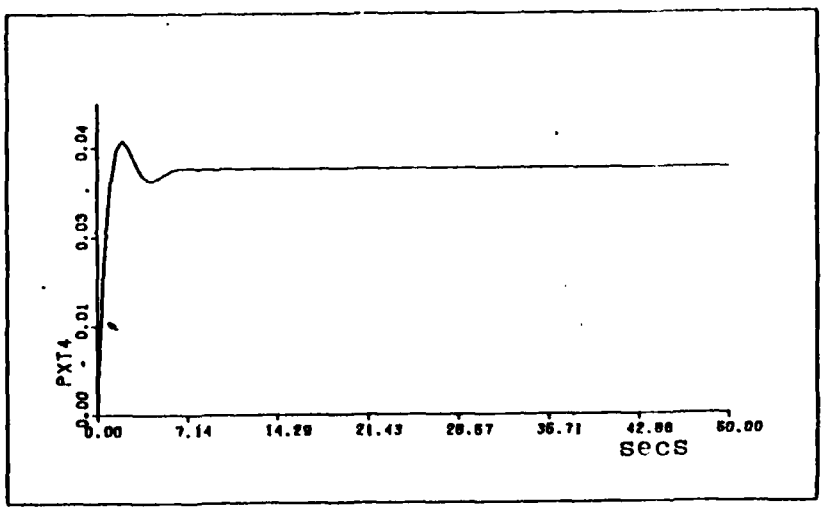
a) mean of state 1



c) mean of state 4

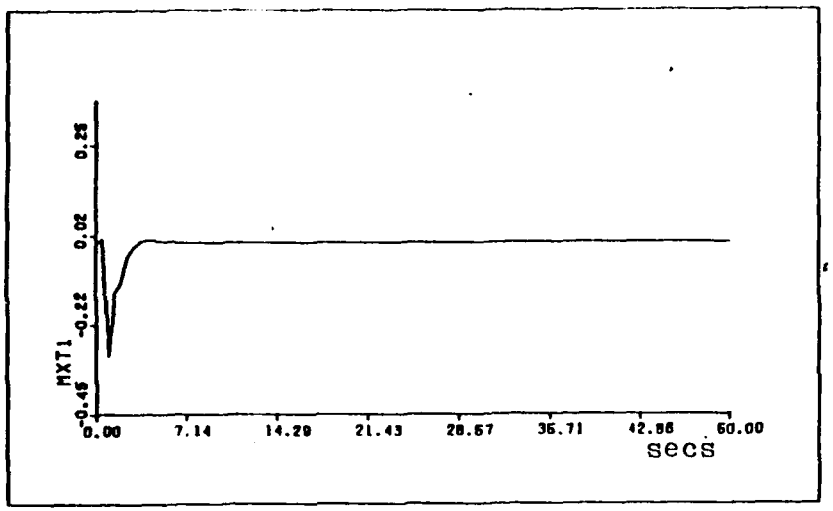


b) variance of state 1

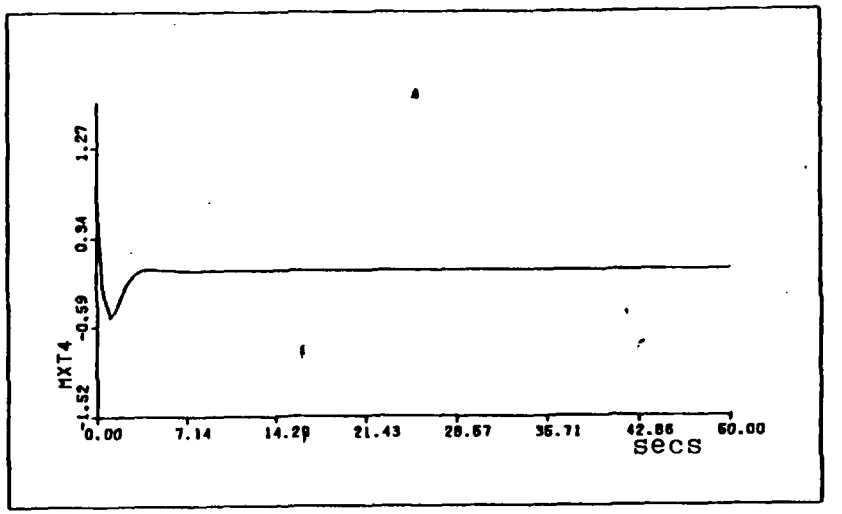


d) variance of state 4

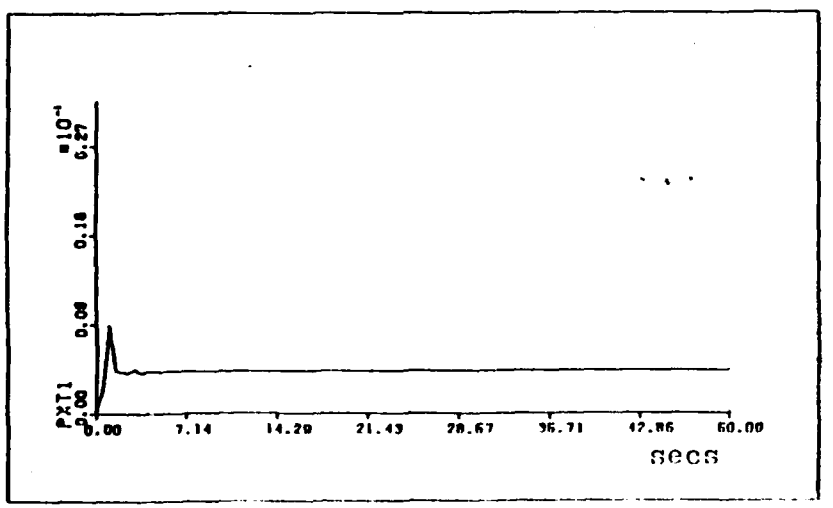
Fig 3.9 Sampled-data performance with ω_b^2=400 and q^2=0.01



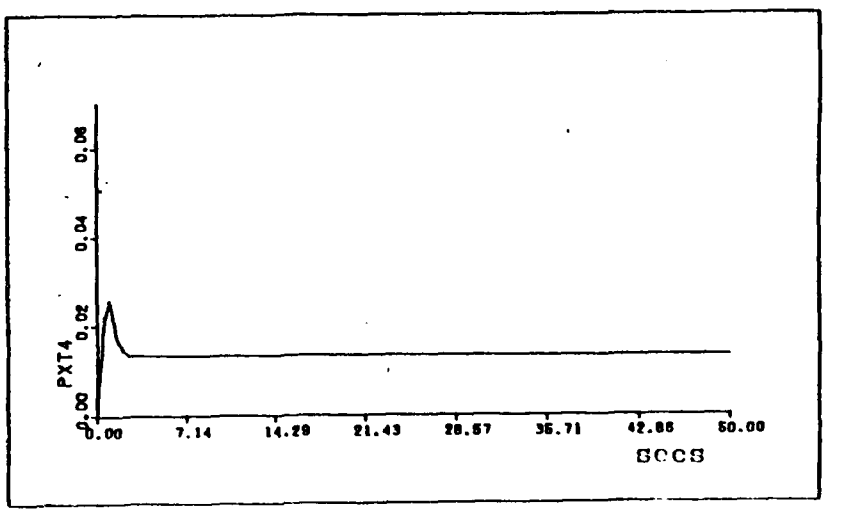
a) mean of state 1



c) mean of state 4



b) variance of state 1



d) variance of state 4

Fig 3.10 Sampled-data performance with $\sigma_b^2=700$ and $q^2=(100)^2$

Table 3.5

Steady-State Performance of Sampled-Data Controllers When $\underline{Q}$ is Tuned by Adding ΔQ

Test Case	Δt	Q , ΔQ	Steady-State $P_{x_t x_t}$ a	Steady-State P_{uu}
1	.01	-2 , -1	1.73	.156
1	.01	2 , 100	1.53	1.97
2	.01	10, 1	$3.92(10)^{-5}$, $5.69(10)^{-3}$	2.68
2	.01	10, 100	$2.52(10)^{-5}$, $5.02(10)^{-3}$	6.22
2	.01	10, $(10)^4$	$1.67(10)^{-5}$, $5.90(10)^{-3}$	80.3
2	.01	10, $(10)^6$	$1.56(10)^{-5}$, $8.47(10)^{-3}$	231
3	.01	1 , .1	340 , 1110	672
3	.01	1 , 1	337 , 1122	782
3	.01	1 , 100	325 , 1163	2390
3	.01	1 , $(10)^4$	324 , 1169	4800
3	.01	1 , $(10)^6$	323 , 1169	4900

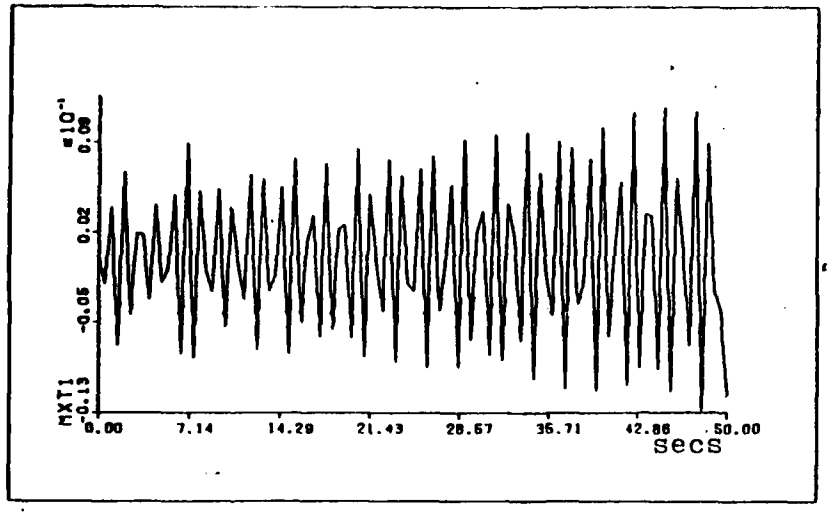
Table 3.5 (Cont)

Test Case	Δt	Q , ΔQ	Steady-State $P_{x_t x_t}$ [a]	Steady-State P_{uu}
1[b]	.01	2, 0	∞	∞
1[b]	.01	2, 1	Large[c]	Large[c]
1[b]	.01	2, 100	9.33	3.43
1[b]	.01	2, $(1000)^2$	7.34	72.1
3[b]	.01	1, 0	∞	∞
3[b]	.01	1, 100	∞	∞
3[b]	.01	1, $(10)^4$	∞	∞
3[b]	.01	1, $(10)^8$	∞	∞

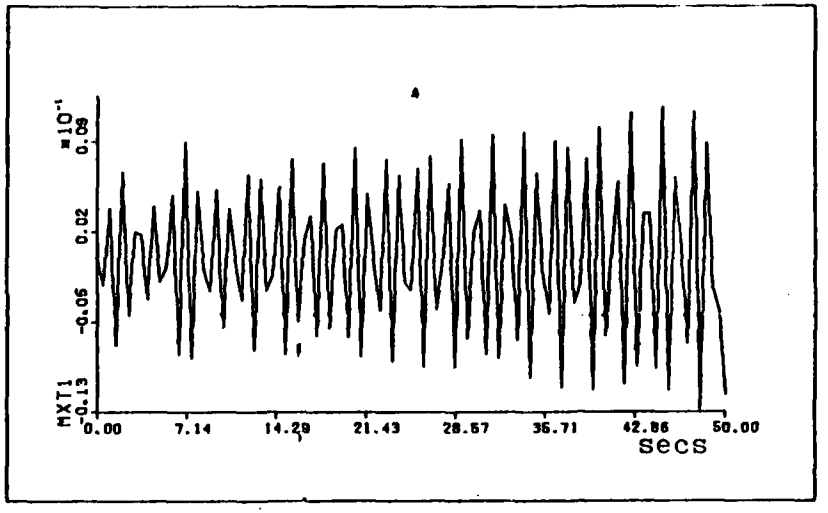
a) only the diagonal terms P_{11}, P_{22} are shown

b) $\underline{F}$ matrices changed in truth model descriptions to show robustness. For Test Case 1, $F_t = .1$; for Test Case 3 $F_{t_{2,2}} = -100$. Test Case 2 not shown, but it could not be stabilized using this technique once a destabilizing $\underline{F}$ matrix was chosen.

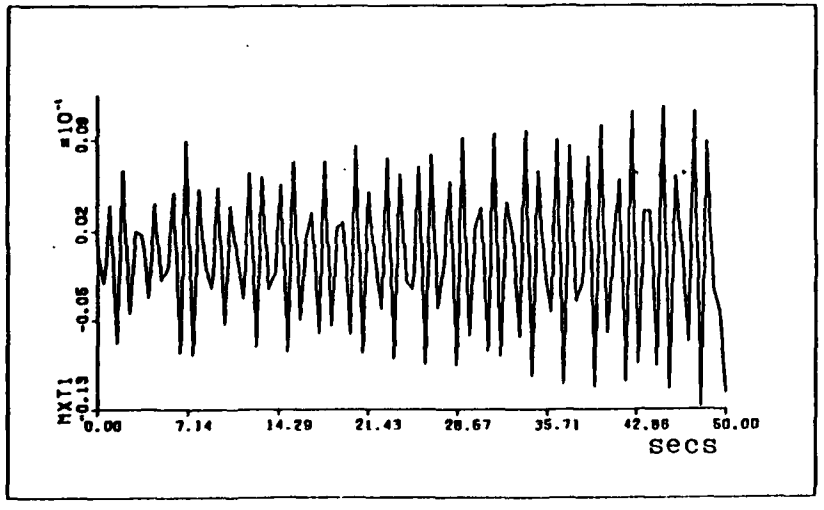
c) Steady-state not rached during length of simulation, but eigenvalues are within the unit circle.



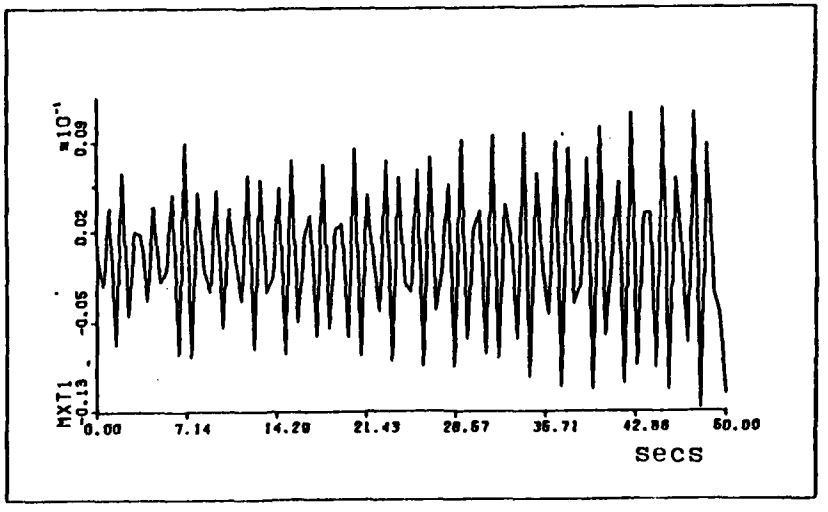
a) mean of state 1, $\underline{Q}$=0.0029



b) mean of state 1, $\underline{Q}$=0.0104

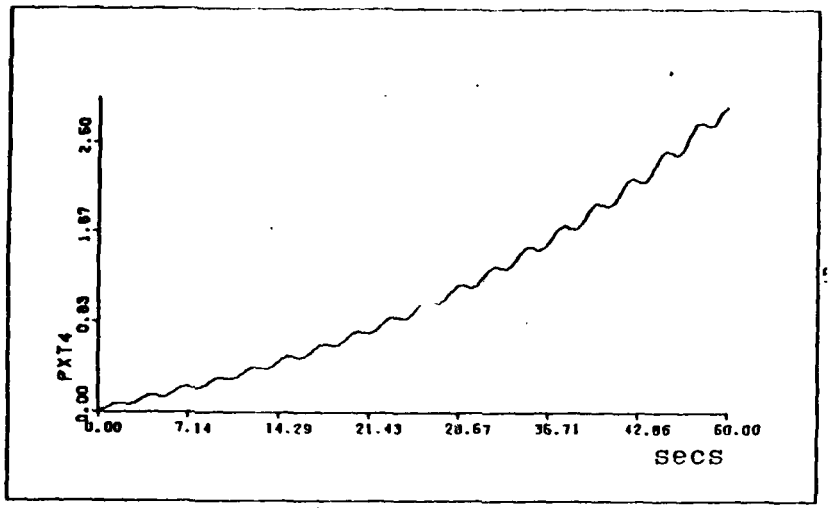


c) mean of state 1, $\underline{Q}$=1

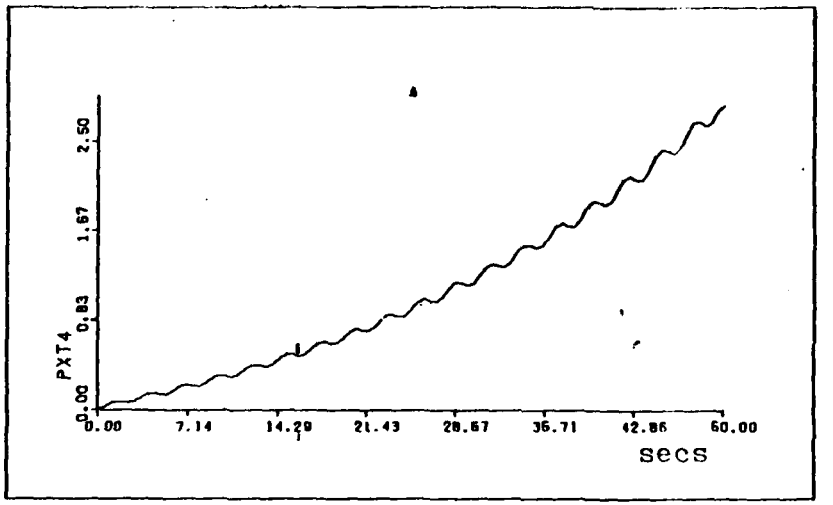


d) mean of state 1, $\underline{Q}$=100

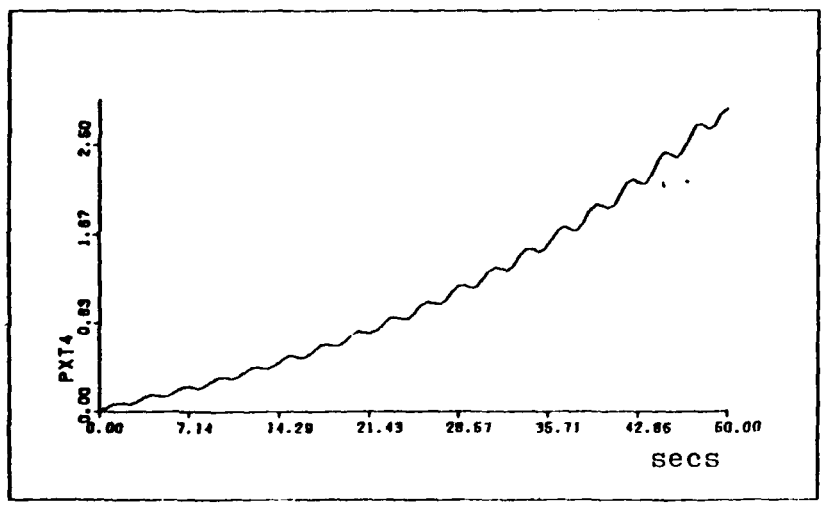
Fig 3.11 Sampled-data performance with ω_h^2=400 and tuned $\underline{Q}$ matrix (means)



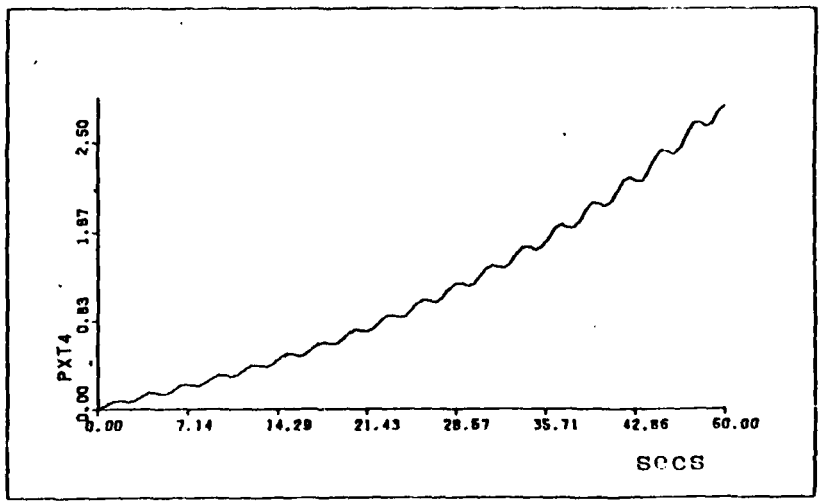
a) variance of state 4, $\underline{Q}$=0.0029



b) variance of state 4, $\underline{Q}$=0.0104



c) variance of state 4, $\underline{Q}$=1



d) variance of state 4, $\underline{Q}$=100

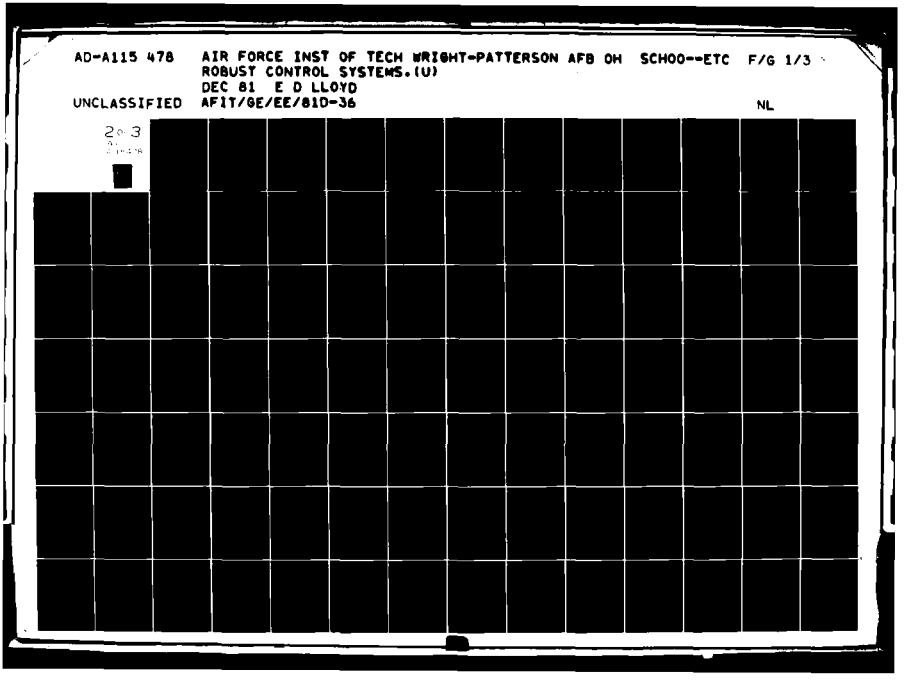
Fig 3.12 Sampled-data performance with ω_b^2=400 and tuned $\underline{Q}$ matrix (variances)

of the Apollo system when the $\underline{Q}$ and $\underline{\Delta Q}$ tuning procedure is used. Note only the mean of state 1 and variance of state 4 are shown but all states exhibit similar behavior. Note, from Figs 3.11 and 3.12, that no matter what value of $\underline{\Delta Q}$ was added, there was little or no performance benefit. While the largest $\underline{\Delta Q}$ shown in Figs 3.11 and 3.12 is 10,000, larger values were tried but the closed loop system eigenvalues rapidly moved outside and away from the unit circle resulting in even more unstable closed-loop operation.

Based on the above comparisons and discussion, the extension of the Doyle and Stein technique to sampled-data systems provides a valuable tool for performance enhancement in the face of uncertain parameters in the controller design model. For the Apollo model, values of q^2 as small as 0.0001 effectively created a stable closed-loop system as shown in Fig F-9. (For $q^2 > 0.01$ the increase in stability appears more dramatic at first glance, Figs 3.10 and F.9, but this is partly due to a much larger initial transient that occurs when q^2 increases.) Tuning $\underline{Q}$, by adding a $\underline{\Delta Q}$, that is a scalar multiple of $\underline{Q}$, had little effect on the Apollo model since this did not add additional noise at the control input as the Doyle and Stein technique did.

Robustness Enhancement by Directly Picking $\underline{K}$. In general the results from this technique were unsatisfactory. For Test Case 2 and the Apollo model, there were no combinations of the scalar design parameter q (see the applicable section of

AD-A115 478 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 1/3
ROBUST CONTROL SYSTEMS.(U)
DEC 81 E D LLOYD
UNCLASSIFIED AFIT/GE/EE/81D-36 NL



Chapter II) and sample period that produced stable closed-loop systems. The results for Test Cases 1 and 3 are presented in Table D.3 of Appendix D. Test Case 1 is the only case used in this study for which this technique provided performance enhancement. For Test Case 3, using the areas of the ellipsoids representing the surfaces of constant likelihood as the basis of comparison, shows that the performance actually degraded with increasing q. (Recall that the product of P_{11} and P_{22} as given in Table D.3 is directly related to these areas.)

One possible cause for the failure of this technique to produce stable closed-loop systems is that the suboptimal control law $\underline{u}(t_i) = -\underline{G}_C^* \hat{\underline{x}}(t_i^-)$ was chosen as the basis of the controller instead of the optimal control law $\underline{u}(t_i) = -\underline{G}_C^* \hat{\underline{x}}(t_i^+)$. The case for the optimal control law should be investigated. Another possibility is that there is an as yet undetected error in software used to implement this controller despite thorough testing and validation. In any event it is noted at this point that there are no stability guarantees for this method of choosing $\underline{K}$ as there are for the LQG controller when $\underline{K}$ is obtained as a result of solving the appropriate matrix Riccati equation (Eq C.18 of Appendix C for example).

Remarks

Closed-loop system eigenvalues were determined for each control system designed and were the primary basis of claims about closed-loop stability. Although these eigenvalues indicate

stability, they do not directly provide performance characteristics versus time of each controlled system. This information is obtained from the mean and covariance analyses developed in Chapter II. Both performance measures have been presented here to provide an adequate portrayal of system characteristics.

Another item of a general nature and applicable to all the modified controllers, whether continuous or discrete-time, is that as the noise level is artifically increased in the system (via Doyle and Stein technique or by tuning $\underline{Q}$ by adding $\underline{\Delta Q}$) the control variable variance generally increases (implying increased control costs are incurred for increased robustness, which is not surprising). This fact should be considered in any design attempting to apply the techniques for enhancing robustness that are discussed in this thesis.

One final note involves the performance of Test Case 2. This system did not behave "nicely" during the course of this investigation. It could not be made stable when picking the Kalman filter gain directly and it could not be stabilized (using either of the two procedures tried here) once the $\underline{F}$ matrix was changed to destabilize the closed-loop system for robustness studies. The performance of this Test Case should be investigated further and characterized.

IV Recommendations

The recommendations fall into two main categories. The first deals with the basic nature of robust control systems and the second deals primarily with increasing the utility of the interactive computer program developed to support this study.

Basic Investigation

The first item that requires further investigation and explanation is the fact that the Doyle and Stein technique did not provide noticeable performance enhancement for the continuous-time Apollo system. There are two aspects about this particular model that may have affected the results. The first is that there is no damping in the bending mode dynamics, which places a pair of poles on the imaginary axis. An attempt to allow the Doyle and Stein technique more "maneuvering room" by moving these poles off the imaginary axis (non-zero damping factor) was tried but was not successful. The second is that for the unmodified continuous-time controller, noise enters only states 2 through 5 but when the Doyle and Stein technique is used, noise now enters before the first order deterministic lag and thus into all 5 states. The effects of this aspect on the Doyle and Stein technique need to be characterized.

A second item is that, for Test Case 3, one of the LQG controller poles (eigenvalues of $\underline{F} - \underline{B}\ \underline{G}_c^* - \underline{K}\ H$)

is positive but the closed-loop poles are all negative and thus the closed-loop system is stable. This phenomenon has been observed before by C.D. Johnson ("State-variable Design Methods May Produce Unstable Feedback Controllers", International Journal of Control, 1979, Vol 29, No. 4, 607-619) and should be investigated and characterized. Another phenomenon that should be investigated is the fact that there are upper limiting values on q^2 (in the Doyle and Stein technique) beyond which the discretized continuous-time and sampled-data controllers no longer cause closed-loop system stability, something not observed for the purely continuous-time case.

The results for picking $\underline{K}$ directly were unsatisfactory and this approach whould be investigated further. The investigation should include the possibility of using the optimal control law $\underline{u}(t_i) = -\underline{G}_c^* \hat{\underline{x}}(t_i^+)$ as the basis of the formulation. As mentioned previously, there are no guarantees of closed-loop stability (even with no mismatch between the design and truth models) when this method is used, unlike the case of using the optimal gains from LQG controller synthesis methods (when there is no mismatch of design and truth models).

For the cases where good results were obtained, extensions should be attempted. The possibility of adding time-correlated noise in a fashion similar to that of Doyle and Stein's technique for adding white noise should be investigated. The results would have potential applications in cases where the noise process is known to be frequency limited. Thus the controller would only expend extra control energy over a

specific frequency bandwidth to achieve robustness instead of over the entire frequency spectrum as is the case for white noise. In addition, the results of extending these techniques to a more general purpose controller, (as opposed to the simple regulator used here) such as a Command Tracker Generator/ Proportional Plus Integral Plus Filter Controller, should be investigated (Ref 10). Also the performance benefits for other specific applications, such as in aircraft flight control systems, should be investigated.

Program Improvements

Since the program is interactive and the CYBER remote terminals limit the amount of memory that can be used at AFIT, the size of the systems that the program can currently handle is restricted to less than eighth order systems. Two approaches can be taken to alleviate this problem. One would be to rework the current interactive program to use several of many available techniques to streamline/optimize the source code so as to reduce core memory requirements. Use of FORTRAN overlay structure, reworking the system model storage in conjunction with limiting the allowable number of inputs and outputs are several of the potential techniques that could be used. (See Appendix C.) A second approach would be to produce a non-interactive version to handle high order systems.

There are several other items that could be changed and/or added to make the program more useful. One is to provide a better discretization technique for producing the

discretized continuous-time controller in order to remove some of the negative effects that the discretization process has on controller performance when the assumption used in the first order approximation (small sample period compared to system characteristic times) is not valid. A potentially useful option would be to provide plots of the time histories of the mean tracking errors versus just plotting the means of the states. This could be accomplished relatively easily since the performance analysis software currently propagates the state estimates; it just does not store them for plotting. Note this would be useful in making comparisons with data available in the literature since tracking error data is used widely for comparisons. Another potentially useful option would be to provide time histories of the sampled-data mean and covariance between sample-time which would provide additional insight into system performance. The current time history data provides no information about system behavior between sample periods and thus the possibility of aliasing errors exists. Finally, an option to compute the eigenvalues of the steady-state covariance matrix, $\bar{P}_{x_t x_t}$, would be useful in evaluating the closed-loop system performance of various controllers as design parameters are changed since they are directly related to the areas of the surfaces of constant likelihood.

Bibliography

1. Ackerman, Juergen E., A Robust Control System Design. AFOSR-TR-79-0743 Report on research supported under Grant AFOSR-78-3633. Urbana, Illinois: Coordinated Sciences Laboratory, University of Illinois, June 1979. (AD 071 162).

2. Doyle, John C. and G. Stein. "Robustness with Observers," IEEE Transactions on Automatic Control, AC-24 (4): 607-611 (August 1979).

3. Doyle, John C. "Guaranteed Margins for LQG Regulators," IEEE Transactions on Automatic Control, AC-23 (4): 601 (August 1978).

4. Floyd, Richard M. Doctoral Student (personal interviews). Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio. June 1981 to December 1981.

5. Kleinman, David L. A Description of Computer Programs Useful in Linear Systems Studies. Technical Report TR-75-4. Storrs, Connecticut: University of Connecticut, October 1975.

6. Krogh, Bruce and J. B. Cruz, Jr. "Design of Sensitivity Reducing Compensators Using Observers," IEEE Transactions on Automatic Control, AC-23 (6): 1058-1062 (December 1978).

7. Kwakernaak, Huibert and Raphael Swan. Linear Optimal Control Systems. New York, New York: John Wiley and Sons, Inc., 1972.

8. Maybeck, Peter S. "Combined Estimation of States and Parameters for On-line Applications," (Doctoral Dissertation). Massachusetts Institute of Technology, Cambridge, Massachusetts 1972.

9. Maybeck, Peter S. Stochastic Models, Estimation, and Control, Volume I. New York, New York: Academic Press, 1979.

10. Maybeck, Peter S. Stochastic Models, Estimation, and Control, Volume 2 (tentative title), unpublished text. Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio.

11. Maybeck, Peter S. Professor of Electrical Engineering (personal interviews). Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, April 1981 to December 1981.

12. Maybeck, Peter S. Spacecraft Autopilot Development Memo #17-68. Cambridge, Massachusetts: Massachusetts Institute of Technology Instrumentation Laboratory, October 17, 1968.

13. Nuzman, Dwayne W. and N. R. Sandell, Jr. "An Inequality Arising in Robustness Analysis of Multivariable Systems," IEEE Transactions on Automatic Control, AC-24 (3): 472-493 (June 1979).

14. Safonov, Michael G. and Michael Athans. "Gain and Phase Margin for Multiloop LQG Regulators," IEEE Transactions on Automatic Control, AC-22 (2): 173-179 (April 1977).

15. Safonov, Michael G. "Frequency-Domain Design of Multivariable Control Systems for Insensitivity to Large Plant Model Errors," Proceedings of the 18th Annual IEEE Conference on Decision and Control. 247-249. New York, New York: IEEE Control Systems Society, December 1979.

16. Widnall, William S. Applications of Optimal Control Theory to Computer Controller Design. Cambridge, Massachusetts: The M.I.T. Press, 1968.

Appendix A

Software Flowcharts

This appendix contains flowcharts of the software developed during the course of this thesis. In this appendix, acronyms such as LQGRP, CHPTR and DDTCON are the subroutine names as they appear in the FORTRAN source code. When these acronyms appear beneath the lower right corner of a block in the flowchart, this indicates that the functions in that box are performed by the given subroutines. All such subroutines have their own flowcharts and descriptions. Flowcharts begin with a subroutine or program name and end with a return, end or stop. Each subroutine description has a reference to the corresponding flowchart number in parentheses next to the subroutine title.

LQGRP (Fig A.1)

LQGRP is the main program name. LQGRP sequences the three primary program modes. The first primary mode is for input. All matrices/vectors associated with the controller model and truth model are entered when LQGRP is in the input mode. The second primary mode develops and formats for performance analysis, either a continuous-time or sampled-data controller depending on a user input. The last primary mode in LQGRP is the performance analysis. The performance analysis is a covariance analysis as described in the body of the thesis for continuous-time and sampled-data systems.

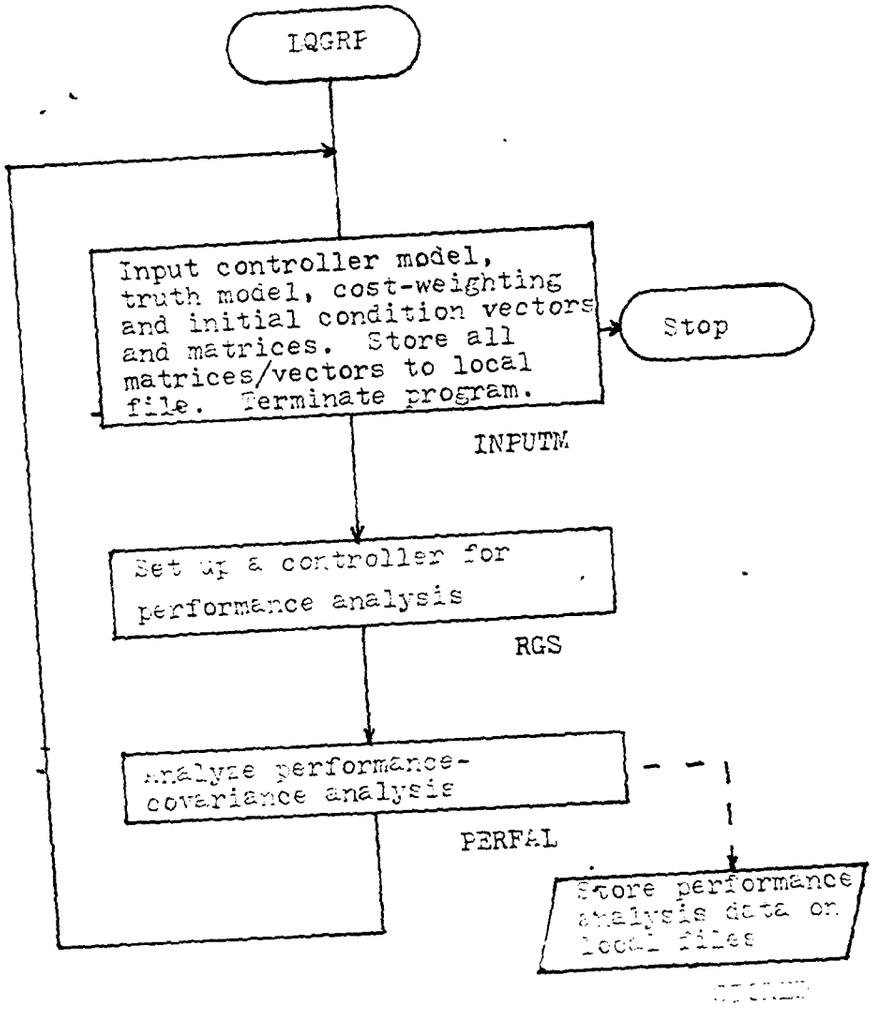
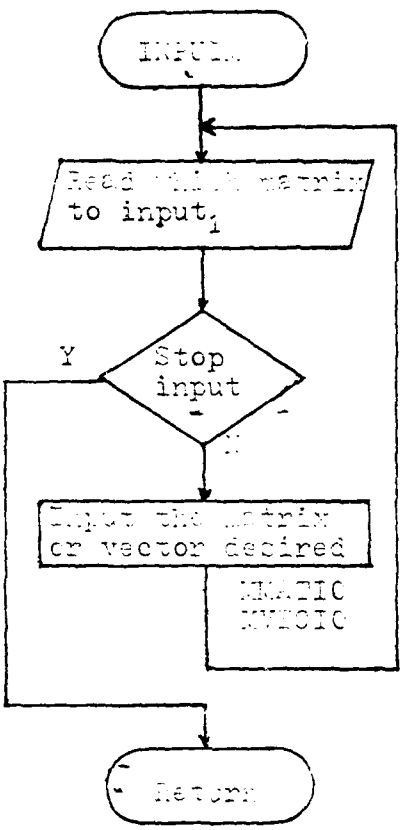

IQGRF
Input controller model, truth model, cost-weighting and initial condition vectors and matrices. Store all matrices/vectors to local file. Terminate program.
Stop
INPUTM
Set up a controller for performance analysis
RGS
Analyze performance-covariance analysis
PERFAL
Store performance analysis data on local files

Fig A.1 IQGRF

INPUTM (Fig A.2)

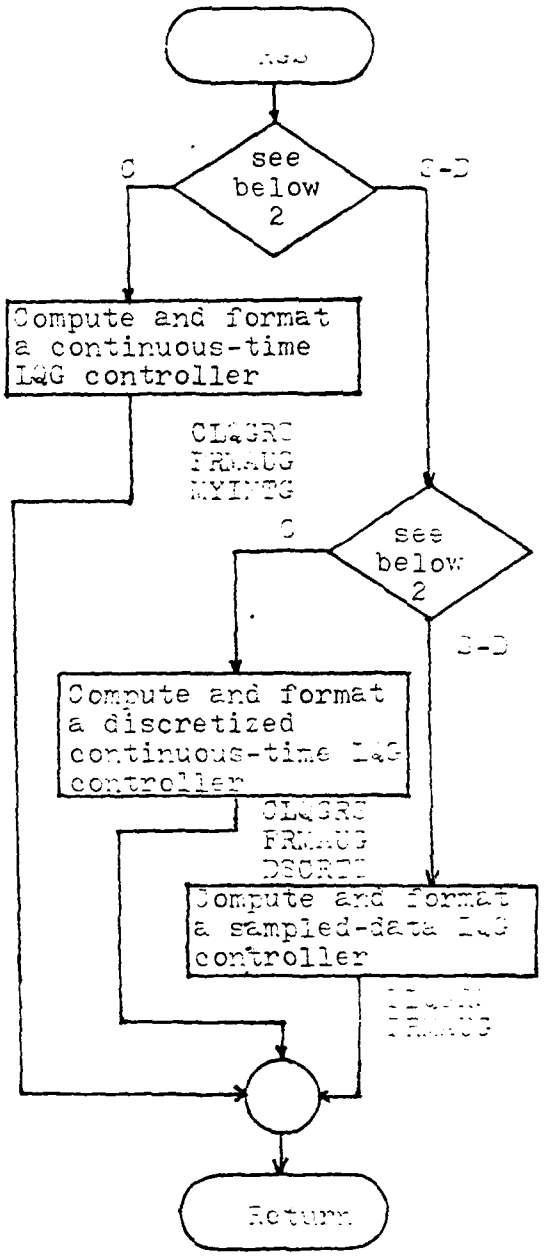
The subroutine INPUTM, the first primary mode of LQGRP, allows the user to input or output system model matrices for the truth model and controller model. In addition, it allows the user to specify cost-weighting matrices needed for optimal LQG controller computation. The subroutine INPUTM is entered at the beginning of each simulation run. Each time it is entered, the user can select to change and/or print any, all or none of the matrices or portions of any or all matrices by choosing appropriate options. One set of options specifies a particular matrix or vector, the other set specifies what to do with that matrix or vector. The options are listed in Table E.1 of Appendix E.

Two additional options in this routine allow the user to store all matrices, and subsequently retrieve them from a local file that can be stored as a permanent file upon program termination. This option is especially useful for systems of large dimension that will be used during many different sessions of running the program. The user merely has to attach the permanent file created during a previous run and execute one option to recover the entire set system matrices. Normal program termination can only be accomplished when in the input mode. Program termination is accomplished by specifying any matrix/vector option and then specifying an input/output option of zero.



There are 21 input options, one for each matrix of the truth and controller models, the cost-weighting functions, one to terminate the input routine, and two for storing and retrieving all matrices/vectors and dimensions from a local file (See Table E.1)

Fig A.2 INPUT



2 User selects which controller to compute and format, continuous-time (C) or sampled-data (S-D)

Fig A.3 RGS

RGS (Fig A.3)

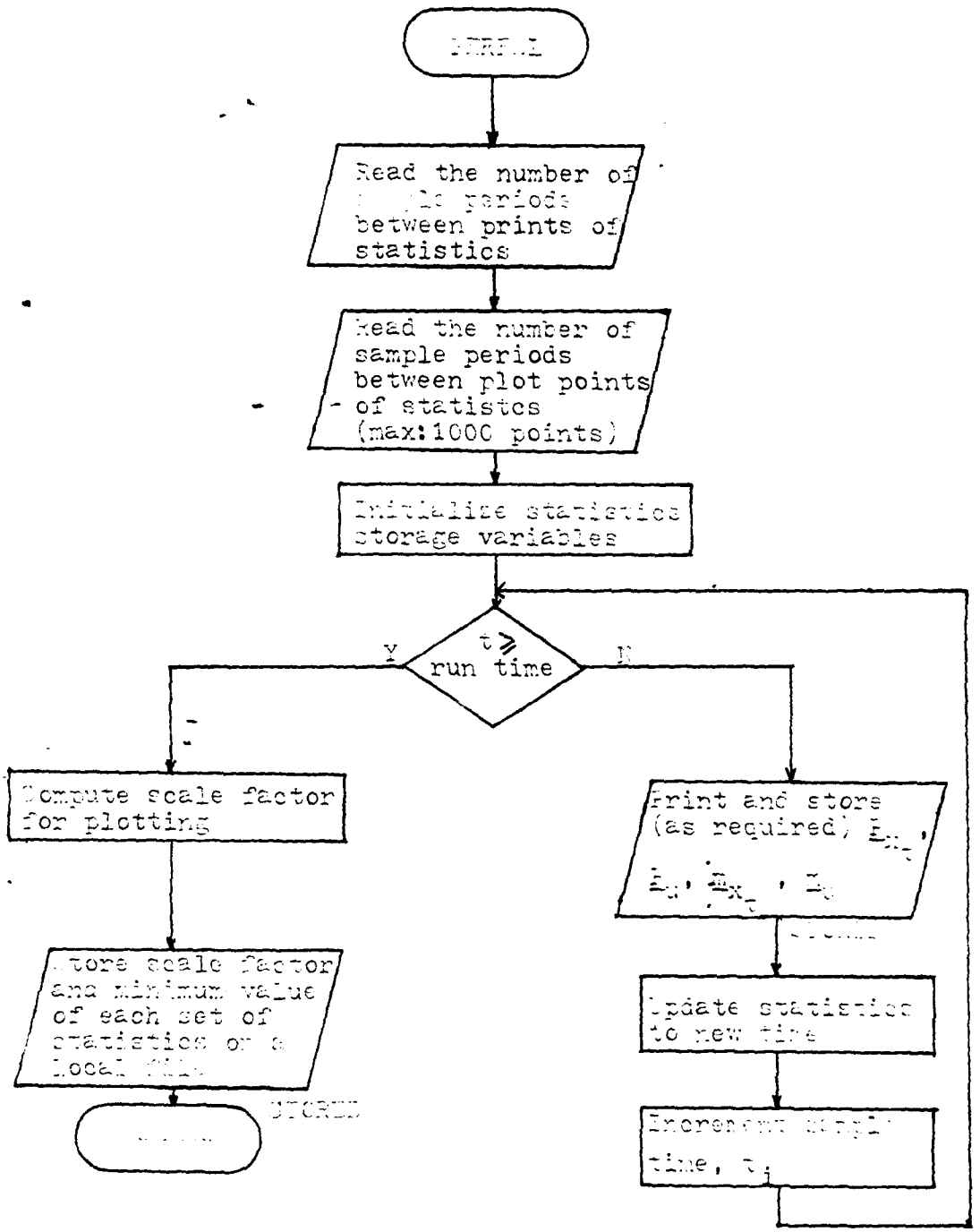
The subroutine RGS, directs the development and formatting of LQG controllers for performance analysis by subroutine PERFAL. Based on a user input to select either a continuous-time, discretized continuous-time or a sampled-data LQG regulator, RGS calls the appropriate subroutines CLQGRS or DLQGRS necessary to compute the various quantities associated with each type controller. After the desired controller is properly formatted, RGS calls subroutine FRMAUG to form the augmented system matrices as described in the performance analysis sections of this thesis. Eigenvalues of the closed-loop system are computed (See subroutine MEIGN) if the user wants to see them.

PERFAL (Fig A.4)

The subroutine PERFAL performs the performance analysis as described in both performance analysis sections of this thesis. PERFAL uses the flag, IFLGSD, which is set in the subroutine RGS, to determine whether to perform a continuous-time or discrete-time system performance analysis.

CLQGRS (Fig A.5)

The subroutine CLQGRS, called by RGS, performs the computations to specify a continuous-time LQG regulator and format it for performance analysis. CLQGRS calls subroutine CKFTR to calculate steady-state Kalman filter gain $\bar{K}$, and calls CDTCON to compute the optimal steady-state


Read the number of
between prints of
statistics
Read the number of
sample periods
between plot points
of statistcs
(max:1000 points)
Initialize statistics
storage variables
t ≥
run time
Y
N
Compute scale factor
for plotting
Update statistics
to new time

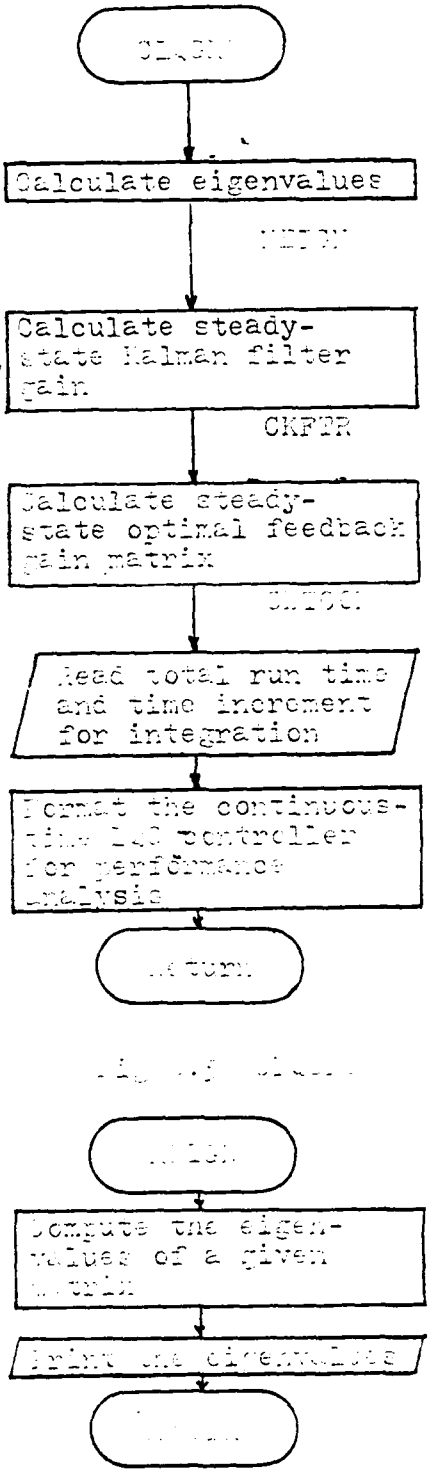
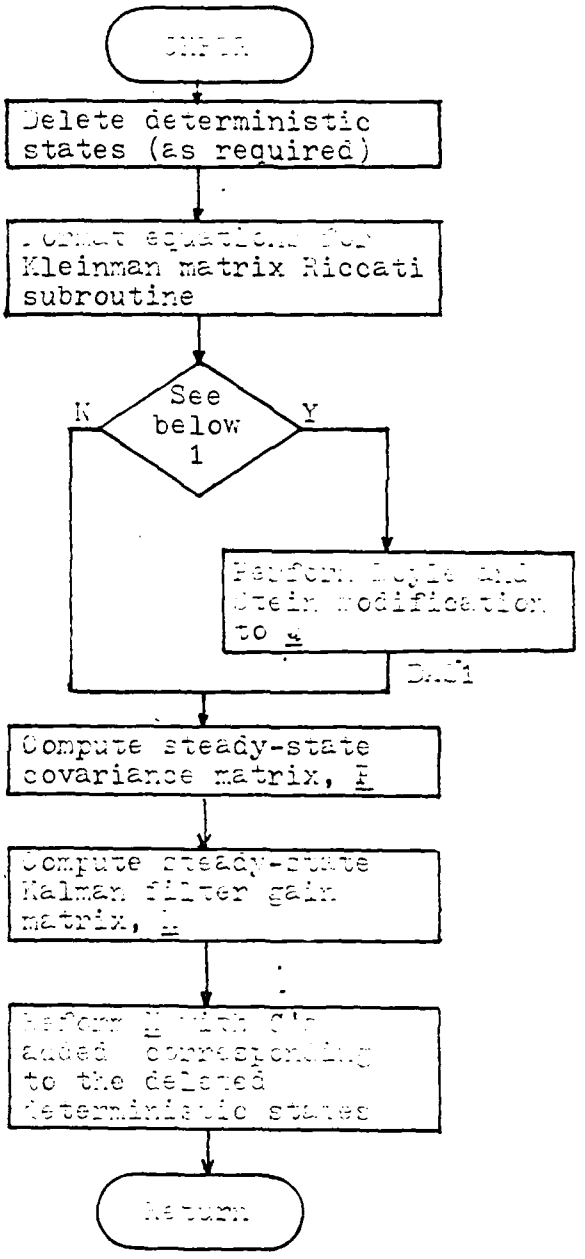


Fig A.6



1 User specifies whether or not perform Doyle and Stein modification

feedback gain matrix, $\bar{G}_c^*$. Eigenvalues of the truth model $\underline{F}$ matrix, the controller model $\underline{F}$ matrix, $[\underline{F}_f - K_f H_f]$ (filter poles), $[\underline{F}_f - B_f G_c^*]$ (LQ controller poles) and $[\underline{F}_c - B_f G_c^* - K_f H_f]$ (LQG controller poles) are calculated if the user wants to see them (MEIGN).

MEIGN (Fig A.6)

The subroutine MEIGN is called by CLQGRS to compute and then print the eigenvalues of any given square matrix.

CKFTR (Fig A.7)

The subroutine CKFTR is called by CLQGRS to perform the computations for determining the steady-state continuous-time Kalman filter gain $\bar{K}$ as in Eq (2.13). This subroutine deletes deterministic states from the filter equations as described in the Deterministic State Augmentation section of this thesis. In addition, the user may select to use the Doyle and Stein method to enhance robustness of the controller. This is done through a call to subroutine DAS1.

DAS1 (Fig A.8)

The subroutine DAS1, when called by CKFTR, gives the user the necessary design options to perform modifications to the Kalman filter gain calculations that are described in the Enhancing Robustness in Continuous-Time LQG Controllers section of this thesis. The user selects the scalar design parameter, q, and the matrix design parameter, $\underline{V}$, and then must select to calculate the modified $\underline{Q}(q)$ matrix. After observing the

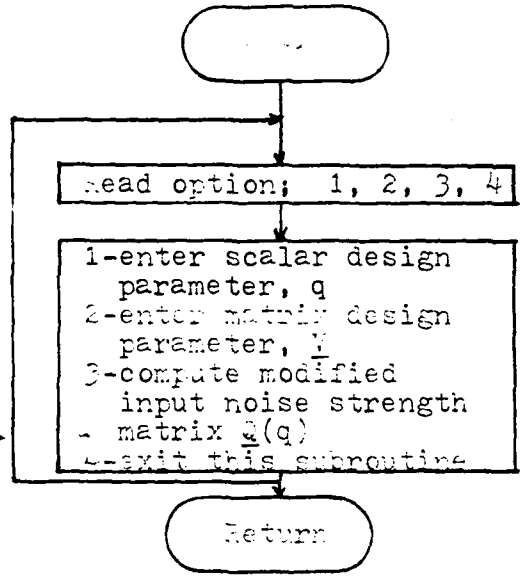


Fig A.8 DAS1

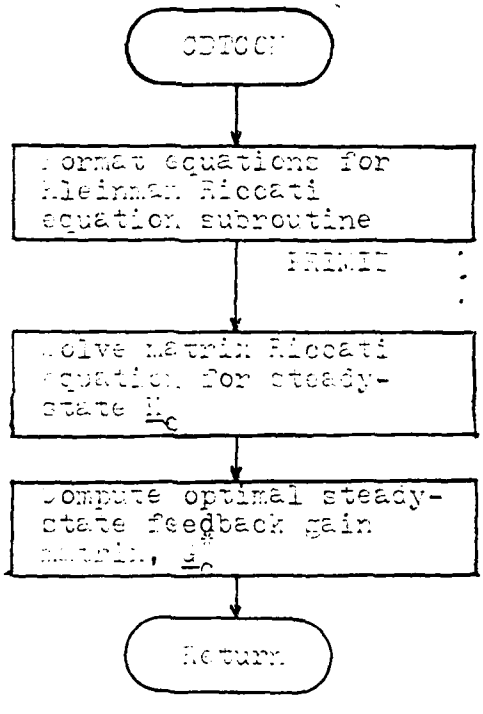


Fig A.9

modified $\underline{Q}(q)$ matrix the user can select either to exit the routine or recalculate $\underline{Q}(q)$ with new parameters. Note, the user may chose any option at any time in this routine and therefore must insure that values are chosen for q and $\underline{V}$ prior to calculating $\underline{Q}(q)$ and that $\underline{Q}(q)$ is calculated prior to leaving the routine.

CDTCON (Fig A.9)

The subroutine CDTCON is called by CLQGRS to perform the necessary calculations to obtain the continuous-time optimal steady-state feedback gain matrix, $\underline{\bar{G}}_c^*$ as described by Eqs (2.10) and (2.14). This subroutine makes provision for non-zero cross cost-weighting matrices by calling the subroutine PRIMIT. PRIMIT supplies an equivalent transformed system of equations in which the cross cost-weighting terms are zero. This transformation is necessary because the matrix Riccati solver can handle only those systems of equations with zero cross cost-weighting matrices. A discussion of the transformation is in Appendix C.

MYINTG (Fig A.10)

The subroutine MYINTG is called by RGS and DLQGRS. It provides an equivalent discrete-time representation of any given set of $\underline{F}$, $\underline{B}$, $\underline{G}$ and $\underline{Q}$ matrices. That is, it computes and returns the state transition matrix $\underline{\Phi}$ based on $\underline{F}$. It computes and returns $\underline{Q}_d$, the discrete-time representation of $\underline{Q}$, which is an integral function of $\underline{\Phi}$, $\underline{G}$ and $\underline{Q}$. In addition,

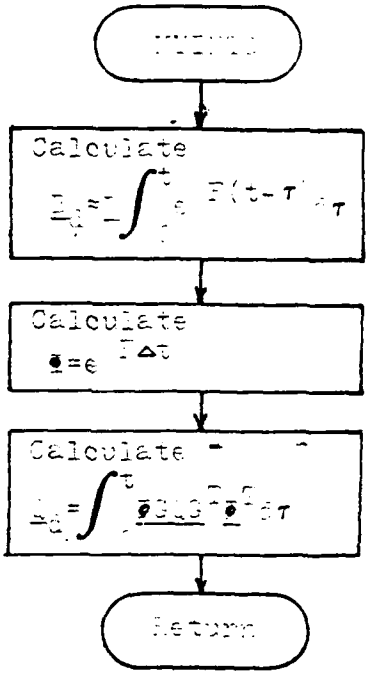

Calculate
Calculate
Calculate
Return

Fig ..11 [illegible]

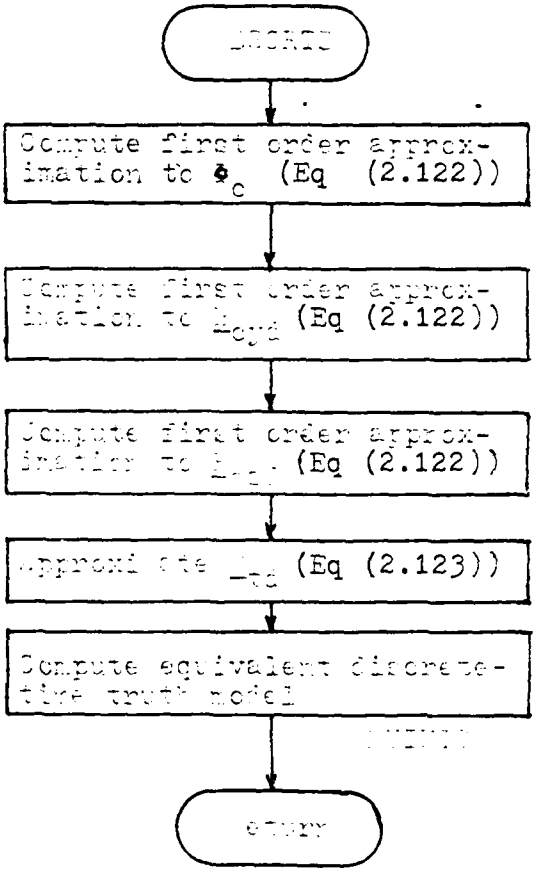

Compute first order approximation to Φ_c (Eq (2.122))
Compute first order approximation to $\underline{L}_{cyd}$ (Eq (2.122))
Compute first order approximation to $\underline{L}_{cu}$ (Eq (2.122))
Approximate $\underline{L}_{cd}$ (Eq (2.123))
Compute equivalent discrete-time truth model

Fig ..11 [illegible]

it computes and returns $\underline{B}_d$, which is a function of $\underline{B}$ and the integral of $\underline{\Phi}$. See Fig A.10 for the equations used in this calculation.

DSCRTZ (Fig A.11)

The subroutine DSCRTZ takes an appropriately formatted (from CIQGRS) continuous-time LQG controller and discretizes it using the first order approximations described in the Doyle and Stein-Technique in Discrete-Time Systems - 1 section of this thesis. These approximations are then properly formatted for the performance analysis routine. In addition, this subroutine computes and then formats for performance analysis, an equivalent discrete-time truth model and, if appropriate, the discrete-time approximation $\underline{R}_d = \underline{R}_c/\Delta t$ where Δt is the sample time and $\underline{R}_d$ is the discrete-time approximation to $\underline{R}_c$, the strength of the continuous-time measurement noise (Ref 10).

DLQGRS (Fig A.12)

The subroutine DLQGRS is called by RGS to perform all necessary computations to specify a sampled-data controller and then formats the controller for performance analysis as discussed in the Sampled-Data LQG Controller section of this thesis. DLQGRS calls XSU and then DDTCON to compute the steady-state optimal feedback gain matrix $\bar{\underline{G}}_c^*$. It then computes a steady-state Kalman filter gain $\bar{\underline{K}}$, using the subroutine DKFTR or the subroutine PKDIRC depending on whether the user wishes to compute $\bar{\underline{K}}$ from the matrix Riccati equation (DKFTR) or to pick $\bar{\underline{K}}$ directly (PKDIRC) (as in the section

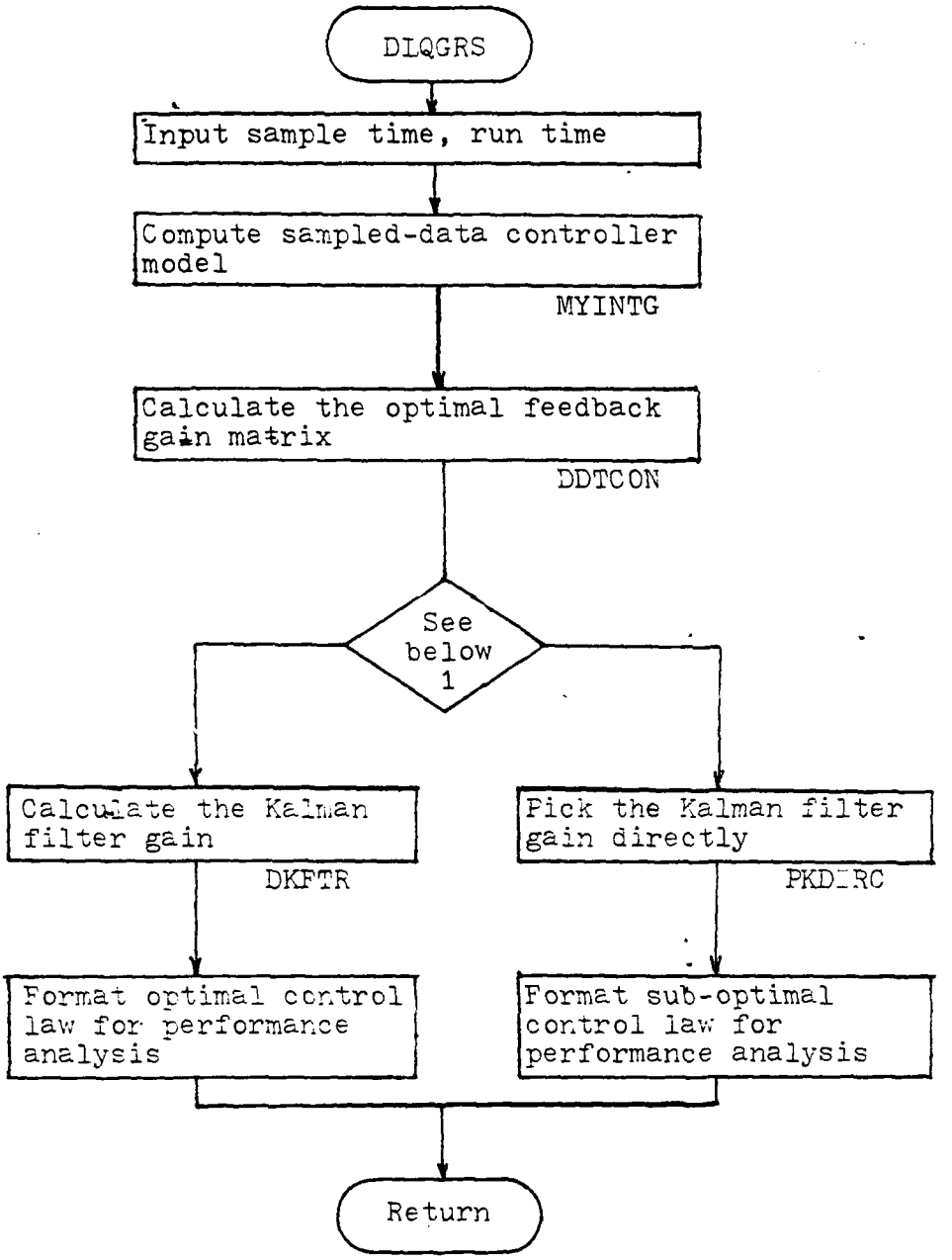


Fig A.12 DLQGRS

1 User selects method of obtaining Kalman filter gain

of this thesis titled Enhancing Robustness of Discrete-Time Systems by Directly Choosing K.) Filter poles, LQ controller poles and LQG controller poles are calculated (MEIGN) if the user wants to see them.

XSU (Fig A.13)

The subroutine XSU is called by DLQGRS to compute the matrices $\underline{X}(t_i)$, $\underline{S}(t_i)$ and $\underline{U}(t_i)$ as described in Eqs (2.95) through (2.97). Note, XSU does not directly solve Eq (2.95) through (2.97) but solves an approximation to their solution forms as discussed in Appendix C, Programming Considerations.

DDTCON (Fig A.14)

The subroutine DDTCON is called by DLQGRS to compute the steady state optimal feedback gain matrix, $\bar{G}_C^*$, for a sampled data controller. Some data formatting complexities involved in using Kleinman's matrix Riccati equation solver to compute the $\bar{G}_C^*$ are discussed in detail in Appendix C. One involves computing values for the integral definitions of $\underline{X}$, $\underline{S}$ and $\underline{U}$ as described in the Sampled-Data Controller section of this thesis (See subroutine XSU). Another involves converting a system with a non-zero cross cost-weighting matrix, $\underline{S}$, to an equivalent one with a zero cross cost-weighting term since the Kleinman matrix Riccati solver does not have provisions for non-zero $\underline{S}$ (this conversion is done by subroutine PRIMIT).

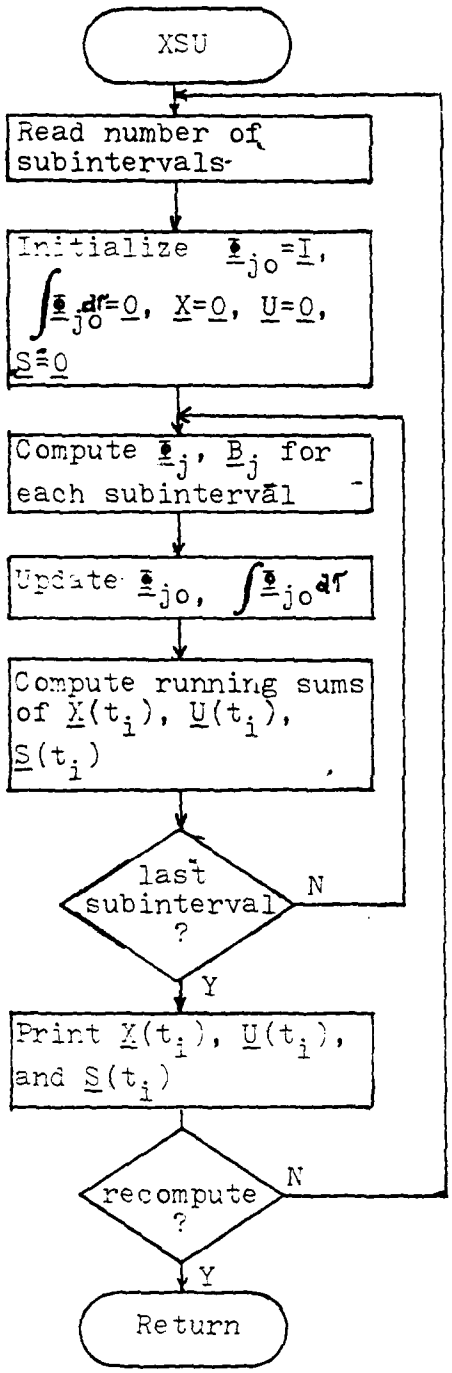


Fig A.13 XSU

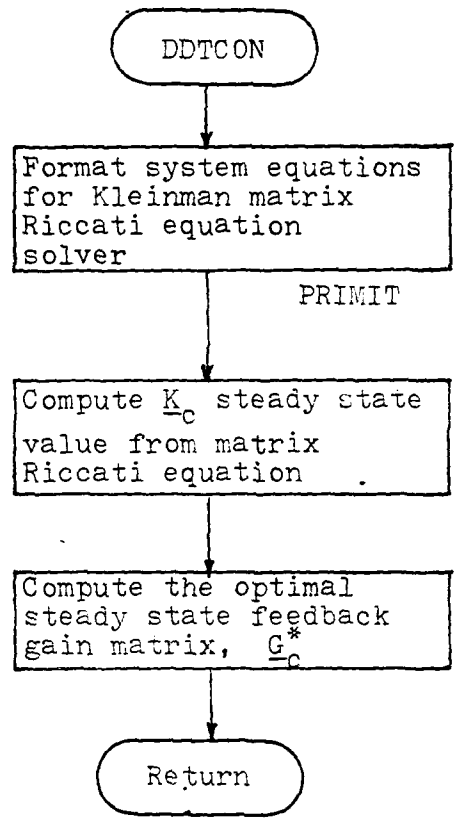


Fig A.14 DDTCON

DKFTR (Fig A.15)

The subroutine DKFTR, when called by DIQGRS, computes the steady-state Kalman filter gain, $\bar{K}$, for a sampled-data LQG regulator as discussed in the Sampled-Data LQG Controller section of this thesis. There is a provision to modify the computed value of $\bar{K}$ by altering Q_d (the covariance on input noise matrix) that appears in the matrix Riccati equation for $\bar{K}$. The modification is performed, when requested by the user, by calling subroutine DAS2.

DAS2 (Fig A.16)

The subroutine DAS2, when called by DKFTR, performs a modification to Q_d (the covariance of the input noise) that is a modification to the Doyle and Stein technique that is applicable to sampled-data systems. See section The Doyle and Stein Technique in Discrete-Time Systems - 2 in the body of this thesis, for a description of this modification. DAS2 is very similar to DAS1 and as in DAS1, the user chooses options in order to enter a scalar design parameter, to enter a matrix design parameter, to calculate the modified Q_d and to exit the routine.

PKDIRC (Fig A.17)

The subroutine PKDIRC, when called by DIQGRS, performs the necessary input options and computations to derive a Kalman filter gain $\bar{K}$ in the manner described in the Enhancing Robustness of Discrete-Time Systems by Directly Choosing K

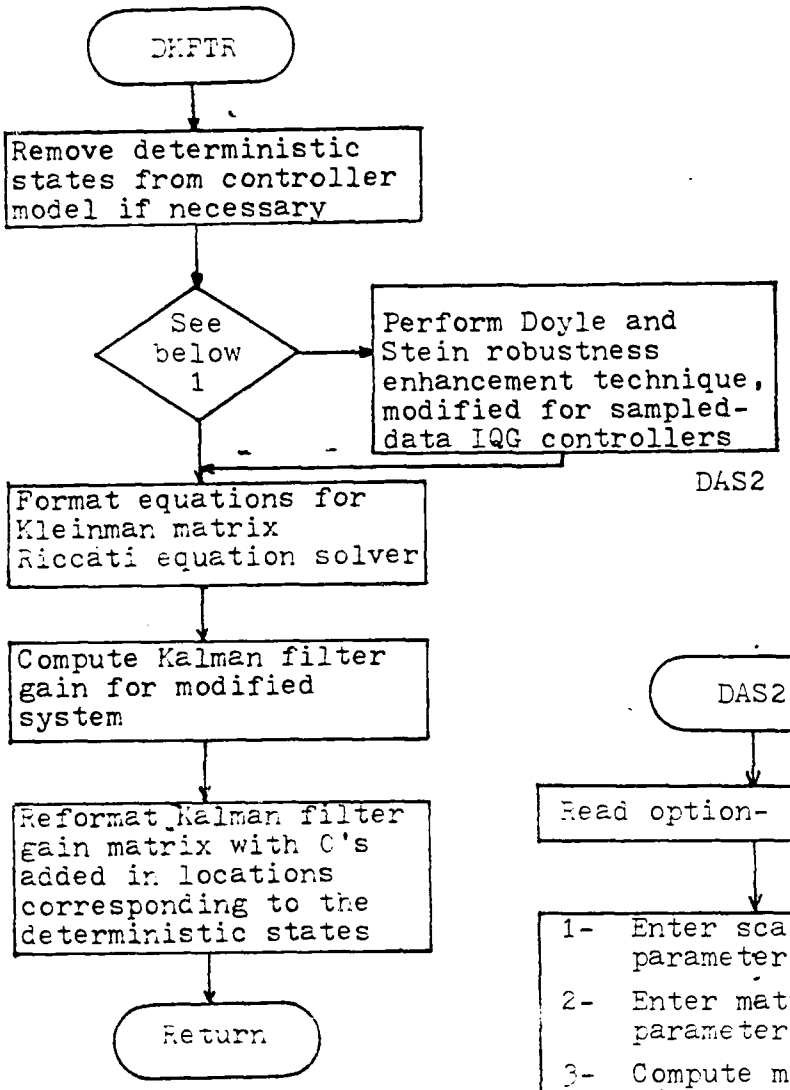


Fig A.15 DKFTR

1 User decides whether or not to perform Doyle and Stein modification

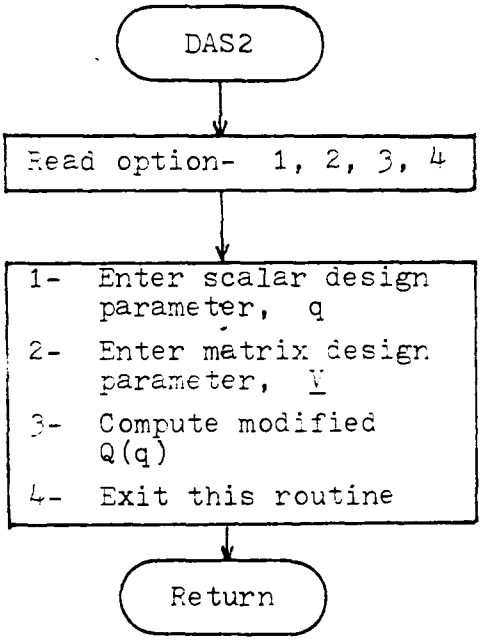


Fig A.16 DAS2

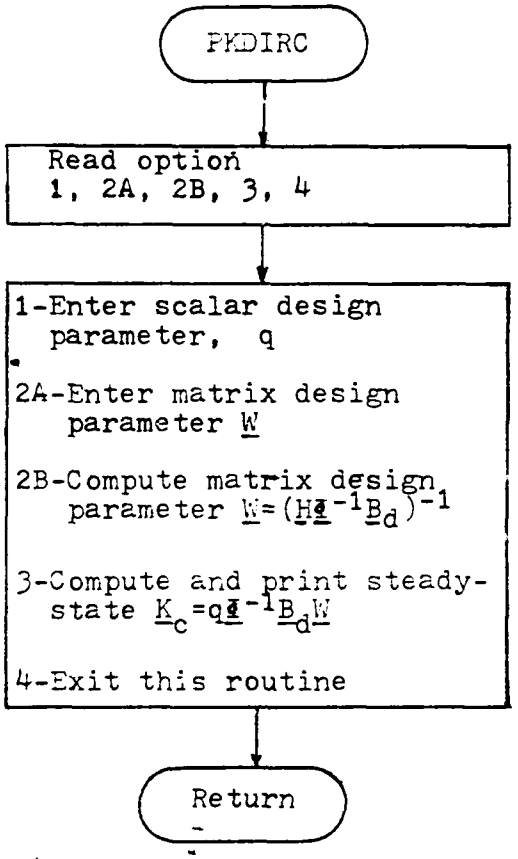


Fig A.17 PKDIRC

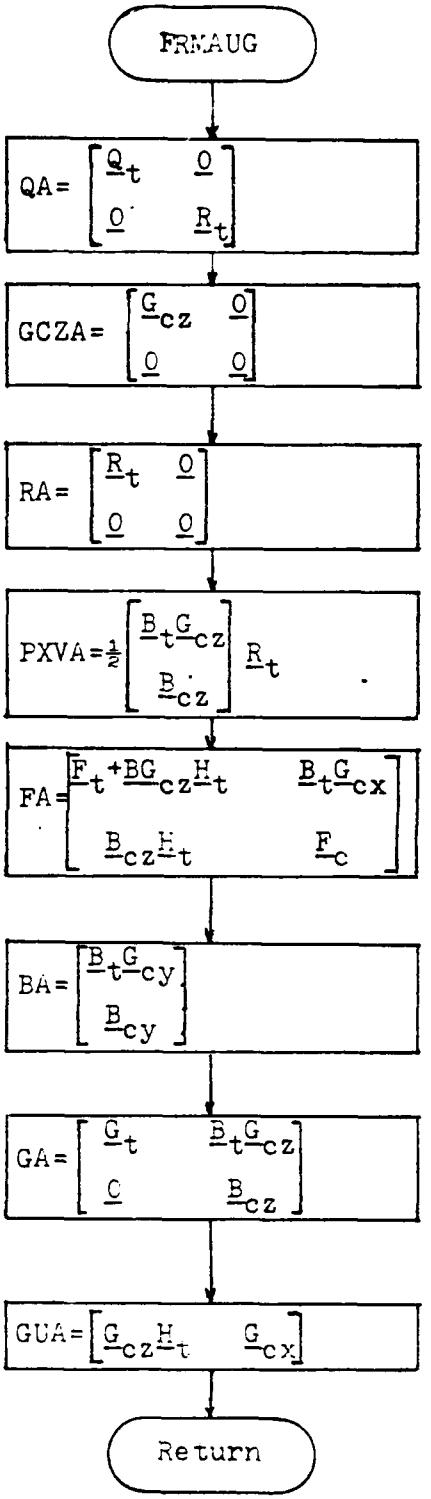


Fig A.18 FRMAUG

section of this thesis. The user must specify the scalar design parameter, q, and one of two methods for choosing the matrix design parameter, $\underline{W}$. $\underline{W}$ can either be chosen directly or can be calculated as in Eq (2.129).

FRMAUG (Fig A.18)

The subroutine FRMAUG receives data in either the continuous-time or sampled-data LQG controller format and forms the augmented matrices required by the performance analysis algorithms described in both the Continuous-Time Performance Analysis and the Sampled-Data Performance Analysis sections of this thesis. That is, it forms $\underline{F}_a$ or $\underline{\Phi}_a$, $\underline{B}_a$ or $\underline{B}_{da}$, $\underline{G}_a$ or $\underline{G}_{da}$, $\underline{Q}_a$ or $\underline{Q}_{da}$, and $\underline{P}_{x_a v_t}$ for continuous-time systems. In addition, it also forms $\underline{R}_a$, $\underline{G}_{cza}$, $\underline{G}_{ua}$. $\underline{R}_a$ and $\underline{G}_{cza}$ are simply $\underline{R}_t$ and $\underline{G}_{cz}$ in the upper left partitions of $\underline{R}_a$ and $\underline{G}_{cza}$ respectively. All other elements in $\underline{R}_a$ and $\underline{G}_{cza}$ are zero. It is necessary to form $\underline{R}_a$ and $\underline{G}_{cza}$ in order to use the Kleinman matrix routines which require that all matrix arguments be of the same declared dimension. $\underline{G}_{ua}$ is formed out of expediency. It is the lower partition of the matrix.

$$\begin{bmatrix} \underline{I} & \underline{0} \\ \underline{G}_{cz} \underline{H}_t & \underline{G}_{cz} \end{bmatrix}$$

which appears in Eqs (2.47), (2.49) and (2.50). With $\underline{G}_{ua}$ thus defined, $\underline{P}_{uu} = \underline{G}_{ua} \underline{P}_{x_a x_a} \underline{G}_{ua}^T$. This eliminates the unnecessary multiplications associated with the upper partition of the matrix above.

Note that $\underline{P}_{x_a v_t}$ as defined in Eq (2.57) is calculated only for continuous-time systems and then only when $\underline{G}_{cz}$ in Eq (2.27A) is not equal to zero.

STORED (Fig A.19)

The subroutine STORED is called by PERFAL to format data from the performance analysis subroutine, PERFAL, for storage on local files. This allows the user to store data as a permanent file upon program termination and, later, to recall the permanent files for printing or plotting as required.

PRIMIT (Fig A.20)

PRIMIT is an auxiliary routine used by DDTCON and CDTCON for optimal deterministic controller gain calculations. It transforms a given system of equations with non-zero cross cost-weighting matrices to an equivalent system of equations with zero cross cost-weighting matrices. The requirement for and details of this modification are discussed in Appendix C. CDTCON and DDTCON transform the resulting optimal feedback gain matrix back to a form consistent with the original system equations.

MMATIO (Fig A.21), MVECIO (Fig A.22)

These subroutines are auxiliary routines that are called by most other subroutines to perform input and output of matrices and vectors respectively. Both routines perform actions on the desired array/vector based on the value of

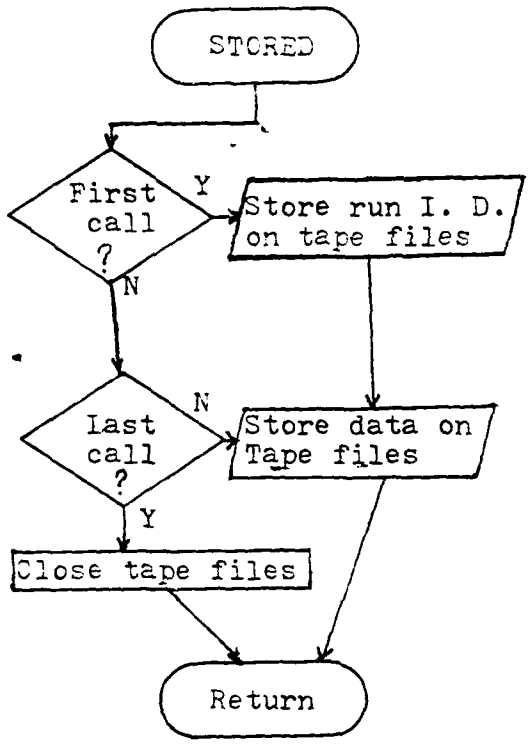


Fig A.19 STORED

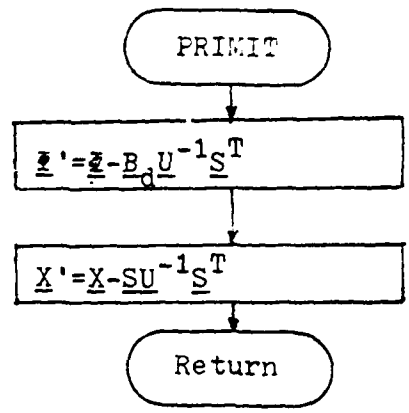


Fig A.20 PRIMIT

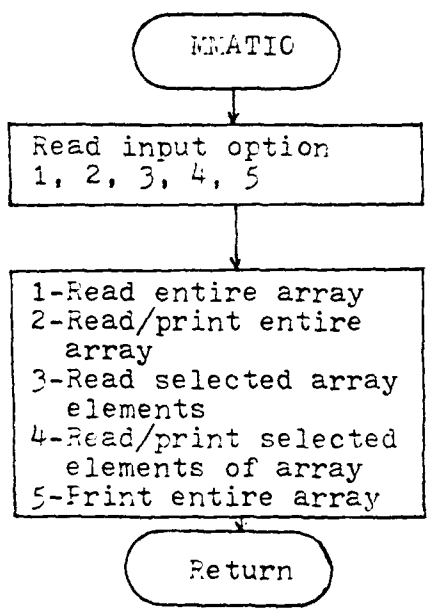


Fig A.21 MMATIO

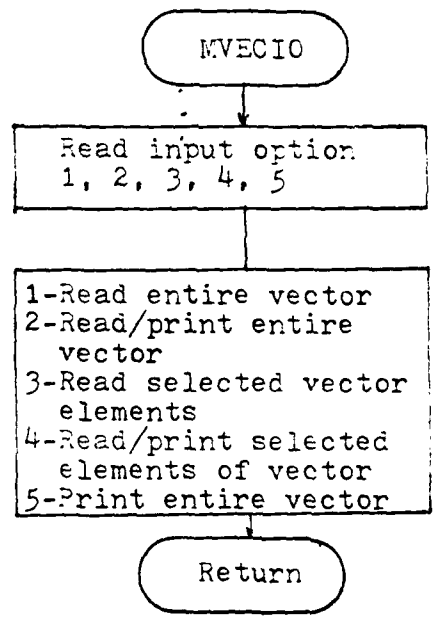


Fig A.22 MVECIO

three parameters, IO, KIN, KOUT. KIN and KOUT tell the routines which file to read from or write to, respectively. IO can have any of 5 values; 1, 2, 3, 4, 5. If IO= 1, the routines read all elements in list directed (and row by row for arrays) format. IO= 2 performs the same read but also writes out the entire array/vector. IO= 3 lets the routine read selected elements in the array/vector. IO= 4 performs the same read function as IO= 3, but it then prints the values read. IO= 5 causes the routine to print out the entire array/vector. Note that if IO is any number than those listed above, a call to these subroutines produces no action other than a return to the calling program.

AUGMAT (Fig A.23)

The subroutine AUGMAT is an auxiliary routine used to form augmented matrices. Based on the flag IFORM, AUGMAT either forms

$$\begin{bmatrix} A \\ B \end{bmatrix} \text{ or } \begin{bmatrix} A & B \end{bmatrix}$$

when given the two matrices A and B and their dimensions.

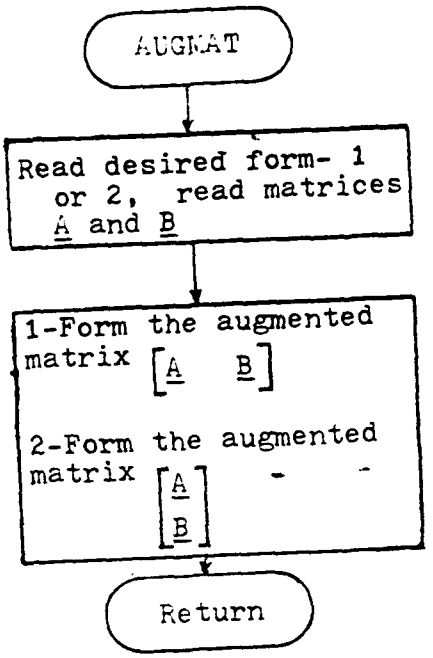


Fig A.23 AUGMAT

Appendix B

Software Source Code

This appendix contains the FORTRAN V software written as a result of this thesis. The FORTRAN V source code implements the flowcharts that are specified and discussed in Appendix A.

Table B.1 contains a listing of the variables as used in the various sections of this thesis along with their FORTRAN V counterparts as used in the software.

TABLE B.1

Correspondence Between Variables in Thesis and FORTRAN Source Code

Variables as in Thesis	FORTRAN Counterparts	Variables as in Thesis	FORTRAN Counterparts
$\underline{F}_t$	FT	$\exp(-\underline{F}\,\Delta t)$	EAT
$\underline{B}_t$	BT	$\int_o^t \underline{\Phi} G Q G^T \underline{\Phi}^T d\tau$	INTGA
$\underline{G}_t$	GT	$\int_o^t \underline{I} B d\tau$	INTBA
$\underline{H}_t$	HT	$\underline{P}_{x_a v_t}$	PXVA
$\underline{Q}_t$	QT	$\underline{G}_{cx}$	GCX
$\underline{R}_t$	RT	$\underline{G}_{cy}$	GCY
$\underline{F}_f$	FM	$\underline{G}_{cz}$	GCZ
$\underline{B}_f$	BM	$\underline{B}_{cy}$	BCY
$\underline{G}_f$	GM	$\underline{B}_{cz}$	BCZ
$\underline{H}_f$	HM	$\underline{F}_c$	FC
$\underline{Q}_f$	QM	$\underline{\Phi}_t$	PHIT
$\underline{R}_f$	RM	$\underline{\Phi}_m$	PHIM
$\underline{W}_{xx}$	WXX	$\underline{\Phi}_c$	FC
$\underline{W}_{uu}$	WUU	Δt	DELTIM
$\underline{W}_{xu}$	WXU	$\underline{X}$	X
$\underline{G}_c^*$	GCSTR	$\underline{S}$	S
$\underline{K}_f$	RKFSS	$\underline{U}$	U
$\underline{F}_a$	FA	$\underline{\Phi}_j$	PHIJ
$\underline{B}_a$	BA	$\underline{B}_j$	BJ
$\underline{G}_a$	GA	$\underline{G}_{ua}$	GUA
$\underline{Q}_a$	QA	$\underline{W}$	W
$\underline{M}_{x_a}$	MXA	$\underline{V}$	V
$\underline{P}_{x_a}$	PXA	q	SQ
$\underline{M}_{x_t}$	MXT		
$\underline{P}_{x_t}$	PXT		

TABLE B.1 (con't)

Notes:

a. A "D" appended to the right side of the Fortran variable indicates that it is an equivalent discrete-time representation of the variable. For example, BTD is the equivalent discrete-time counterpart of $\underline{B}_{td}$.

b. A "T" appended to the right side of the Fortran variable or expression that designates a matrix/vector, indicates that the matrix/vector is transposed. For example, HTT is $\underline{H}_t^T$.

c. An "I" appended to the end of a Fortran variable or expression that designates a matrix/vector indicates that the matrix/vector is inverted. For example, PHIMI is $\underline{\Phi}_m^{-1}$.

```
*DECK LQGRP
      PROGRAM LQGRP(OUTPUT=64,TAPE6=OUTPUT,TAPE12=64,TAPE13=64,
     1TAPE14=64,TAPE15=64,INPUT=64,INPUT,TAPE8=64,TAPE7)
C  THIS PROGRAM PERFORMS A PERFORMANCE ANALYSIS FOR THE LINEAR QUADRATIC
C  GAUSSIAN CONTROLLERS DESCRIBED BY D(X)/DT=FX+BU+GU AND A QUADRATIC
C  COST FUNCTION.      IF DIFFERENT CONTROL ALGORITHMS ARE SUPPLIED THE
C PROGRAM WILL STILL DO A PERFPRMANCE ANALYSIS.  THE METOHODOLOGY IS
C  BASED ON THE PERF ANAL. SECTION IN CHAPTER  14 OF P. S. MAYBECKS
C  TO BE PUBLISHED VOLUME 2 OF STOCH,MODELS,EST AND CONTROL
C
C  MANY OF THE SUBROUTINES USED FOR MATRIX MANIPULATION COME FROM THE
C  ROUTINES  COMPILED BY D. KLIEMAN(TR-75-4, ONR CONTRACT #
C  N00014-75-C1067)
C
C  IN THE FOLLOWING PROGRAM TRUTH MODEL MATRICES ARE TWO LETTERS WITH
C  THE LAST LETTER BEING  -T- .  CONTROLLER MODEL MATRICES ARE TWO
C  LETTERS ENDING IN -M-.  -T- FOLLOWING APARTICULAR MATRIX NAME
C  INDICATES THE MATRIX IS TRANSPOSED.  -I- FOLLOWING A PARTICULAR
C  MATRIX NAME INDICATES THE INVERSE OF THE MATRIX
C
C
C  COMMON BLOCKS MAIN1,MAIN2,INOU ARE REQUIRED BY THE KLIEMAN ROUTINES
C-----INPUT FROM TAPE10    ----OUTPUT TO TAPE11

      CHARACTER MSG*60
      REAL FT(5,5),BT(5,5),GT(5,5),HT(5,5),RM(5,5),
     1    FM(5,5),BM(5,5),GM(5,5),XO(5),HM(5,5),QM(5,5),
     1 PO(5,5),QT(5,5),RT(5,5),COM1(10,10),COM2(10,10),GCSTR(5,5
     7),UXU(5,5),RKFSS(5,5),UU1(5),UU2(5),
     1UM1(5,5),UM2(5,5),UM3(5,5),UM4(5,5),UM5(5,5),UM6(5,5),
     1UMA(10,10),UMB(10,10),UMC(10,10),UMD(10,10),UME(10,10),UMF(10,10),
     1    UUU(5,5),UXX(5,5),FA(10,10),BA(10,10),GA(10,10),RXA(10),
     1    GCX(5,5),GCZ(5,5),QA(10,10),PXA(10,10),PXUA(10,10),
     1    RUOUT(5),RXTOUT(5),PXTOUT(5),PUOUT(5),YD(1000),
     1    GUA(10,10),RXAMIN(5),RXAMAX(5),RUMIN(5),RUMAX(5),
     1  PXTMIN(5),PXTMAX(5),PUMIN(5),PUMAX(5),GCZA(10,10),
     1 RA(10,10),BCY(5,5),BCZ(5,5),GCY(5,5),FC(5,5),
     1UM7(5,5),UM8(5,5),UM9(5,5),UM10(5,5),UU3(10),UU4(10)
      INTEGER IFLGCZ,IRY
      REAL RU(5)
      COMMON /RNTIM/ RNTIME,DELTIM
      COMMON /MAIN2/COM2
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON / INOU/ KIN,KOUT,KPUNCH
      COMMON /MAIN4/NDIM2,NDIM3
      COMMON /MAUN5/ MSG
      COMMON /MAIN6/ ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1   IRHT,IRQA,IO,LOG,IRHM,NUMDTS
C
C
      DATA UMA,UMB,UMC,UMD,UME,UMF  /600*0.0/
      DATA QA,COM1,COM2,FA,BA,GA,PXA,PXUA /800*0.0/
      DATA UU3,UU4 /20*0.0/
      DATA UM1,UM2,UM3,UM4,UM5,UM6,FT,BT,GT,HT /250*0.0/
      DATA UUU,UXX,FM,BM,GM,PO,QT,UXU /200*0.0/
      DATA RT,GCSTR,RKFSS,GCX,GCZ  /125*0.0/
      DATA UU1,UU2,XO,RXA,RU  /30*0.0/
      DATA RM,QM,HM /75*0.0/
      DATA IFLGCZ /0/
C
C
C
      KIN=10
      KOUT=6
      KPUNCH=7
      NDIM=5
      NDIM1=6
      NDIM2=10
      NDIM3=1000
C
C
C
C   ZZZZZZZZZMAIN PROGRAM FOLLOWS ZZZZZZZ
C----  IRXX, ICXX INDICATE ROWS,COLUMNS OF MATRIX XX
C
CZZZZZTHIS PROGRAM CAN HANDLE UP TO 1000 DIFFERENT COMBINATIOS OF
C                  RUNTIME,DELTIM,AND UNSPECIFIED PARAMETERS
      DO 2932 LOG=1,1000
      WRITE(KOUT,11)' '
```

```
11    FORMAT(A1,/)
12    FORMAT(A1)
      WRITE(KOUT,2)'THIS IS RUN NUMBER ',LQG
      CALL INPUTM(FT,BT,GT,HT,FM,BM,GM,HM,PO,QT,QM,RT,RM,XO,
     1  UUU,UXX,UXU)
      IF (IO.EQ.0) THEN
      GO TO 2933
      END IF
      CALL RGS1(GCSTR,RKFSS,GCX,GCY,GCZ,BCY,BCZ,FC,YD,
     1UM1,UM2,UM3,UM4,UM5,UM6,UM7,UM8,UM9,UM10,UU1,UU2,
     1UMA,UMB,UMC,UMD,UME,UMF,FT,BT,GT,HT,QT,RT,
     1 FM,BM,GM,HM,QM,RM,XO,PO,UXX,UUU,UXU,FA,BA,GA,QA,GUA,
     1 MXA,PXA,RA,PXUA,GCZA,IRY,IFLGCZ,IFLGSD,
     1MU,PUMAX,PUMIN,PXTMAX,PXTMIN,MUMAX,MUMIN,MXAMAX,MXAMIN,
     1 PUOUT,PXTOUT,MXTOUT,MUOUT,UU3,UU4)
      WRITE(KOUT,2)' '
      WRITE(KOUT,2)'DO YOU WISH TO CALCULATE THE EIGENVALUES OF THE CLOS
     1ED-LOOP STATE TRANSITION MATRIX USED IN THE PERFORMANCE ANALYSIS
     1(APPLICABLE TO BOTH THE CONTINUOUS-TIME AND SAMPLED-DATA CASE)
     1  Y  OR  N>'
      READ(KIN,12)MSG
      IF (MSG.EQ.'Y') THEN
      WRITE(KOUT,2)'THE CLOSED-LOOP STATE TRANSITION MATRIX EIGENVALUES
     1 ARE...'
      NSAV=NDIM
      NDIM=NDIM2
      NDIM1=NDIM2+1
      CALL MEIGN(UMA,UU3,UU4,IRFA,UME)
      NDIM=NSAV
      NDIM1=NSAV+1
      END IF
      WRITE(KOUT,2)' '
      WRITE(KOUT,2)'TYPE Y  TO   PERFORM THE COVARIANCE ANALYSIS.  TYPE
     1 N TO SKIP IT>'
      READ(KIN,12)MSG
      IF (MSG.EQ.'N')THEN
      GO TO 2932
      END IF
      CALL PERFAL( IRY,IFLGCZ,MXA,GCY,GUA,PXA,PXUA,IFLGSD,
     1 RA,GCZA,YD,UMA,UMB,UME,UMF,UMD,UM1,MUOUT,MXTOUT,PUOUT,PXTOUT,
     1  MXAMIN,MXAMAX,PXTMIN,PXTMAX,MUMIN,MUMAX,PUMIN,PUMAX,MU,UMC,
     1 UU3,UU4)
2932  CONTINUE
2933  WRITE(KOUT,2)'PROGRAM TERMINATED, NO MORE INPUT DATA'
      END


*DECK STORED
      SUBROUTINE STORED(IUCHRN,RNTIME,DELTIM,IFSTCL,IS12SZ,
     1   IS13SZ,IS14SZ,IS15SZ,STOR12,STOR13,STOR14,STOR15,NDIM6)
C THIS SUBROUTINE STORES PLOT DATA TO LOCAL FILES,  TAPE12,TAPE13,
C                  TAPE14 TAPE15 FOR PERMANENT STORAGE.  ISXXSZ IS THE SIZE IF THE
C                  DATA ARRAYS,STORXX.  NDIM IS THE CALLING PROGRAM DIMENSION OF
C                  THE ARRAYS.          NOTE THAT   RUNTIME/DELTIM+2  GIVES THE TOTA
C                  NUMBER OF DATA POINTS IN THE RUN.  THE LAST ENTRY IN EACH
C                  DATA FILE IS THE SCALE FACTOR FOR THE DATA ON THE FILE.  THE
C                  NEXT TO LAST ENTRY IS THE MINIMMUM VALUE OF THE DATA IN THE FIL
C
      DIMENSION STOR12(NDIM6),STOR13(NDIM6),STOR14(NDIM6),STOR15(NDIM6)
      COMMON /INOU/ KIN,KOUT,KPUNCH
      IF (IFSTCL.EQ.0) THEN
C     PUT A RUN HEADER ON THE TAPE
      OPEN(UNIT=12,ERR=10,FILE='TAPE12',RECL=80)
      OPEN(UNIT=13,ERR=10,FILE='TAPE13',RECL=80)
      OPEN(UNIT=14,ERR=10,FILE='TAPE14',RECL=80)
      OPEN(UNIT=15,ERR=10,FILE='TAPE15',RECL=80)
      WRITE(12,101) IUCHRN,RNTIME,DELTIM,IS12SZ
      WRITE(13,101) IUCHRN,RNTIME,DELTIM,IS13SZ
      WRITE(14,101) IUCHRN,RNTIME,DELTIM,IS14SZ
      WRITE(15,101) IUCHRN,RNTIME,DELTIM,IS15SZ
101   FORMAT(' ',I10,1X,2E19.6,I10)
      END IF
      IF (IFSTCL.LT.0) THEN
C     CLOSE FILES AND PUT END OF FILE MARKER ON THEM.
      CLOSE(12,ERR=10)
      CLOSE(13,ERR=10)
      CLOSE(14,ERR=10)
      CLOSE(15,ERR=10)
      RETURN
      END IF
      WRITE(12,102)(STOR12(I),I=1,IS12SZ)
      WRITE(13,102)(STOR13(I),I=1,IS13SZ)
      WRITE(14,102)(STOR14(I),I=1,IS14SZ)
      WRITE(15,102)(STOR15(I),I=1,IS15SZ)
```

```
      RETURN
10    WRITE(KOUT,*)'AN ERROR HAS OCCURRED IN THE STORED ROUTINE'
102   FORMAT(3(4(' ',E19.6,1),/))
      END
*DECK INPUTM
      SUBROUTINE INPUTM(FT,BT,GT,HT,FM,BM,GM,HM,PO,QT,QM,RT,RM,XO
     1,UUU,UXX,UXU)
      CHARACTER MSG*60,MSG1*50
      REAL FT(NDIM,NDIM),BT(NDIM,NDIM),GT(NDIM,NDIM),HT(NDIM,NDIM),HM(ND
     1IM,NDIM),QM(NDIM,NDIM),RM(NDIM,NDIM),XO(NDIM),
     1    UUU(NDIM,NDIM),UXX(NDIM,NDIM),FM(NDIM,NDIM),BM(NDIM,NDIM),
     1 GM(NDIM,NDIM),UXU(NDIM,NDIM),
     1 PO(NDIM,NDIM),QT(NDIM,NDIM),RT(NDIM,NDIM),COM1(1),COM2(1)
      COMMON /MAIN4/NDIM2,NDIM3
      COMMON /MAIN2/COM2
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON / INOU/ KIN,KOUT,KPUNCH
      COMMON /MAUN5/ MSG
      COMMON /MAIN6/ ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1    IRHT,IRQA,IO,LQG,IRHM,NUMDTS
      MSG1='*****-----*****-----*****-----*****-----*****'
C                 ICFT=IRFT,ICFM=IRFM,IRBT=IRFT,IRBM=IRFM,IRGT=IRFT,IRGM=IRFM
C                 IRX=IRF,IRQ=ICQ=ICGM,IRR=ICR=IRHT,IRUXX=ICUXX=IRFM,
C                -IRUUU=ICUUU=ICBM,ICHT=IRFT,ICHM=IRFM,IRGCZ=ICBT,ICGCZ=IRHT,
C     IO IS A INPUT ROUTINE PARAMETER--1=READ,2=READ&PRINT,
C     3= PRINT ONLY, 4=PUNCH
C
      NSAV=NDIM1
      NDIM1=NDIM

C                 IRX=IRF,IRQ=ICQ=ICGM,IRR=ICR=IRHT,IRUXX=ICUXX=IRFM,
C                 IRUUU=ICUUU=ICBM,ICHT=IRFT,ICHM=IRFM,IRGCZ=ICBT,ICGCZ=IRHT,
C     IO IS A INPUT ROUTINE PARAMETER--1=READ,2=READ&PRINT,
C     3= PRINT ONLY, 4=PUNCH
C
      NSAV=NDIM1
      NDIM1=NDIM
      IF (LQG.EQ.1) THEN
      WRITE(KOUT,*)MSG1
      WRITE(KOUT,*)'THE I/O OPTIONS ARE 0,1,2,3,4,5,6,.....'
      WRITE(KOUT,*)'1-READ ENTIRE ARRAY/VECTOR,2-READ AND PRINT'
      WRITE(KOUT,*)'ENTIRE ARRAY/VECTOR, 3-READ,AND 4-READ AND PRINT'
      WRITE(KOUT,*)'SELECTED ARRAY/VECTOR ELEMENTS, 5 PRINT ENTIRE'
      WRITE(KOUT,*)'ARRAY/VECTOR.  6 OR GREATER NO MORE INPUT TO'
      WRITE(KOUT,*)'BE MADE.'
      WRITE(KOUT,*)MSG1
      WRITE(KOUT,*)'SELECT WHICH MATRIX YOU WISH TO ENTER.'
      WRITE(KOUT,*)'BY ENTERING THE APPROPRIATE NUMBER. 1-FT,2-BT'
      WRITE(KOUT,*)'3-GT,4-HT,5-FM,6-BM,7-GM,8-HM,9-PO,10-QT,11-RT,)'
      WRITE(KOUT,*)'12-QM,13-RM,14-XO,15-UUU,16-UXX,17-UXU,18-EQUATE ALL
     1'
      WRITE(KOUT,*)'CONTROLLER MODEL MATICES TO THEIR '
      WRITE(KOUT,*)'TRUTH MODEL COUNTERPARTS,19----- NO MORE DATA'
      WRITE(KOUT,*)'ENTRIES TO BE MADE, 20--STORE ALL MATRICES ON TAPE7'
      WRITE(KOUT,*)'21-  READ ALL MATRICES FROM TAPE8'
      WRITE(KOUT,*)MSG1
      WRITE(KOUT,*)MSG1
      WRITE(KOUT,*)' '
      WRITE(KOUT,*)'FOR SAMPLED DATA MEASUREMENTS,ENTER EITHER A CONTINU
     1OUS RM  TO USE TO APPROXIMATE THE DISCRETE TIME RMD(RMD=RM/SAMPLE
     1TIME) OR ENTER THE DISCRETE TIME RMD'
      WRITE(KOUT,*)MSG1
      WRITE(KOUT,*)MSG1
      END IF
      DO 982 INPUT=1,1000
      WRITE(KOUT,33333)' '
33333 FORMAT(A10,/)
      WRITE(KOUT,*)'ENTER CODE FOR WHICH ARRAY/VECTOR TO BE INPUT)'
      READ(KIN,*)IWCHMA
      GO TO(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21)IWCHMA
C
C***TRUTH MODEL INPUT
1     WRITE(KOUT,*)'ENTER-I/O OPTION, FT MATRIX SIZE>'
      READ(KIN,*,END=27)IO,IRFT
      MSG='TRUTH MODEL F MATRIX ENTRIES'
      CALL MMATIO (FT,IRFT,IRFT,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
      GO TO 982
```

```
2     WRITE(KOUT,*)'ENTER-I/O OPTIONS, COLUMN SIZE OF BT)'
      READ(KIN,*,END=27)IO,ICBT
      MSG='TRUTH MODEL  B  MATRIX ENTRIES'
      CALL MMATIO (BT,IRFT,ICBT,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
       GO TO 982
3     WRITE(KOUT,*)'ENTER-I/O OPTION, COLUMN SIZE OF GT)'
      READ(KIN,*,END=27)IO,ICGT
      MSG='TRUTH MODEL G MATRIX ENTRIES'
      CALL MMATIO (GT,IRFT,ICGT,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
      GO TO 982
4     WRITE(KOUT,*)'ENTER- I/O OPTION, ROW SIZE OF HT)'
      READ(KIN,*,END=27)IO,IRHT
      MSG='H MATRIX ENTRIES'
      CALL MMATIO (HT,IRHT,IRFT,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
      GO TO 982
C***INPUT CONTROLLER MODEL
5     WRITE(KOUT,*)'ENTER-I/O OPTION, FM MATRIX SIZE)'
      READ(KIN,*,END=27)IO,IRFM
      MSG='CONTROLLER MODEL F MATRIX ENTRIES'
      WRITE(KOUT,*)'ENTER THE NUMBER OF DETERMINISTIC STATES IN THIS MOD
     1EL)'
      READ(KIN,*,END=27)NUMDTS
      CALL MMATIO (FM,IRFM,IRFM,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
      GO TO 982
6      WRITE(KOUT,*)'ENTER-I/O OPTION, COLUMN SIZE OF BM)'
      READ(KIN,*,END=27)IO,ICBM
      MSG='CONTORLLER MODEL B MATRIX ENTRIES'
      CALL MMATIO (BM,IRFM,ICBM,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
      GO TO 982
 7     WRITE(KOUT,*)'ENTER-I/O OPTION, COLUMN SIZE OF GM)'
      READ(KIN,*,END=27)IO,ICGM
      MSG='CONTROLLER MODEL G MATRIX ENTRIES'
      CALL MMATIO (GM,IRFM,ICGM,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
      GO TO 982
8     WRITE(KOUT,*)'ENTER I/O OPTION,ROW SIZE OF HM)'
      READ(KIN,*,END=27)IO,IRHM
      MSG='THE CONTROLLER MODEL MEASUREMENT MATRIX, HM, IS'
      CALL MMATIO(HM,IRHM,IRFM,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      ENDIF
      GO TO 982
9     WRITE(KOUT,*)'FT MUST BE ENTERED THRU OPTION 1 PRIOR TO USING THIS
     1 OPTION.DO YOU WISH TO ABORT THIS OPTION, Y  OR  N)'
      READ(KIN,97,END=27)MSG
      IF (MSG.EQ.'Y') THEN
      GO TO 982
      END IF
      WRITE(KOUT,*)'ENTER I/O OPTION,IRFT IS ASSUMED SIZE OF PO)'
      READ(KIN,*,END=27)IO
      MSG='THE INITIAL COVARIANCE MATRIX,PO, IS'
      CALL MMATIO (PO,IRFT,IRFT,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
      GO TO 982
10      WRITE(KOUT,*)'ENTER I/O OPTION, ICGT IS ASSUMED SIZE OF QT)'
      READ(KIN,*,END=27)IO
      MSG='THE INPUT NOISE STRENGTH MATRIX QT IS'
      CALL MMATIO (QT,ICGT,ICGT,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0) THEN
      RETURN
      END IF
      GO TO 982
```

```
11      WRITE(KOUT,*)'ENTER I/O OPTION,IRHT IS ASSUMED SIZE OF RT)'
        READ(KIN,*,END=27)IO
        MSG='THE MEASUREMENT NOISE STRENGTH MATRIX RT IS'
        CALL MMATIO (RT,IRHT,IRHT,IO,KIN,KOUT,NDIM,NDIM1)
        IF (IO.EQ.0) THEN
        RETURN
        END IF
        GO TO 982
12      WRITE(KOUT,*)'ENTER I/O OPTION,ICGM IS ASSUMED SIZE OF QM)'
        READ(KIN,*,END=27)IO
        MSG='CONTROLLER MODEL INPUT NOISE STRENGTH MATRIX, QM'
        CALL MMATIO(QM,ICGM,ICGM,IO,KIN,KOUT,NDIM,NDIM1)
        IF (IO.EQ.0) THEN
        RETURN
        END IF
        GO TO 982
13      WRITE(KOUT,*)'ENTER I/O OPTION, IRMM IS ASSUMED SIZE OF RM)'
        READ(KIN,*,END=27)IO
        MSG='CONTROLLER MODEL MEASUREMENT NOISE STRENGHT MATRIX, RM'
        CALL MMATIO(RM,IRMM,IRMM,IO,KIN,KOUT,NDIM,NDIM1)
        IF (IO.EQ.0) THEN
        RETURN
        END IF
        GO TO 982
14      WRITE(KOUT,*)'FT MUST BE ENTERED THRU OPTION 1 PRIOR TO USING TH IS
     1 OPTION.  DO YOU WISH TO ABORT THIS OPTION, Y  OR  N)'
        READ(KIN,97,END=27)MSG
        IF (MSG.EQ.'Y') THEN
        GO TO 982
        END IF
          WRITE(KOUT,*)'ENTER I/O OPTION,IRFT IS ASSUMED SIZEOF XO)'
        READ(KIN,*,END=27)IO
        MSG='THE INITIAL STATE VECTOR, XO, IS'
        CALL MVECIO (XO,IRFT,IO,KIN,KOUT,NDIM)
        IF (IO.EQ.0) THEN
        RETURN
        END IF
        GO TO 982
15      WRITE(KOUT,*)'BM MUST BE ENTERED THRU OPTION 6 OR 17 PRIOR TO USIN
     1G THIS OPTION.  DO YOU WISH TO ABORT THIS OPTION,  Y  OR  N)'
        READ(KIN,97,END=27)MSG
        IF (MSG.EQ.'Y') THEN
        GO TO 982
        END IF
         WRITE(KOUT,*)'ENTER I/O OPTION,ICBM IS ASSUMED SIZE OF WUU)'
        READ(KIN,*,END=27)IO
        MSG='THE CONTROL FUNCTION COST WEIGHTING MATRIX, WUU'
        CALL MMATIO(WUU,ICBM,ICBM,IO,KIN,KOUT,NDIM,NDIM1)
        IF (IO.EQ.0) THEN
        RETURN
        END IF
        GO TO982
16      WRITE(KOUT,*)'FM MUST BE ENTERED THRU OPTION 5 OR 17 PRIOR TO USIN
     1G THIS OPTION.  DO YOU WISH TO ABORT THIS OPTION, Y  OR N)'
        READ(KIN,97,END=27)MSG
        IF (MSG.EQ.'Y') THEN
        GO TO 982
        END IF
        WRITE(KOUT,*)'ENTER I/O OPTION,IRFM IS ASSUMED SIZE OF WXX)'
        READ(KIN,*,END=27)IO
        MSG='THE STATE COST WEIGHTING MATRIX, WXX'
        CALL MMATIO(WXX,IRFM,IRFM,IO,KIN,KOUT,NDIM,NDIM1)
        IF (IO.EQ.0) THEN
        RETURN
        END IF
        GO TO 982
18      WRITE(KOUT,*)'ALL CONTROLLER MODEL MATRICES HAVE BEEN SET EQUAL TO
     1 THEIR TRUTH MODEL COUNTERPARTS'
        WRITE(KOUT,*)'FT,BT,GT,HT,QT,RT MUST BE ENTERED PRIOR TO USING THI
     1S OPTION.  THE NUMBER OF DETERMINISTIC STATES  MUST BE ENTERED IN
     1 THIS OPTION(FOR CONTROLLER MODEL).'
        WRITE(KOUT,*)' DO YOU WISH TO ABORT THIS OPTION,Y OR N)'
        READ(KIN,97,END=27)MSG
        IF (MSG.EQ.'Y') THEN
        GO TO 982
        END IF
        WRITE(KOUT,*)'ENTER THE NUMBER OF DETERMINISTIC STATES IN THE CONT
```

```
     1ROLLER MODEL)'
      READ(KIN,*,END=27)NUMDTS
      IRFM=IRFT
97    FORMAT(A1)
      ICBM=ICBT
      ICGM=ICGT
      IRHM=IRHT
      CALL EQUATE(FM,FT,IRFT,IRFT)
      CALL EQUATE(BM,BT,IRFT,ICBT)
      CALL EQUATE(GM,GT,IRFT,ICGT)
      CALL EQUATE(HM,HT,IRHT,IRFT)
      CALL EQUATE(QM,QT,ICGT,ICGT)
      CALL EQUATE(RM,RT,IRHT,IRHT)
      GO TO 982
17    WRITE(KOUT,*)'NOTE THAT FM AND BM MUST BE ENTERED THROUGH APPROPRI
     1ATE OPTIONS PRIOR TO EXECUTING THIS OPTION. DO YOU WISH TO ABORT T
     1HIS OPTION, Y  OR  N)'
      READ(KIN,97,END=27)MSG
      IF (MSG.EQ.'Y')THEN
      GO TO 982
      END IF
      WRITE(KOUT,*)'ENTER I/O OPTION, IRFM X ICBM ASSSUMED SIZE WXU)'
      READ(KIN,*,END=27)IO
      MSG='THE CROSS (STATE-CONTROL) COST WEIGHTING MATRIX, WXU'
      CALL MMATIO(WXU,IRFM,ICBM,IO,KIN,KOUT,NDIM,NDIM1)
      IF (IO.EQ.0)THEN
      RETURN
      END IF
      GO TO 982
20    WRITE(KOUT,*)' THIS OPTION STORES ALL MATRICES ON TO TAPE7, DO YOU
     1 WISH TO ABORT THIS OPTION,Y  OR N)'
      READ(KIN,97)MSG
         IF (MSG.EQ.'Y')THEN
         GO TO 982
          END IF
      WRITE(7,*)IRFT,ICBT,ICGT,IRHT,IRFM,ICBM,ICGM,IRHM,NUMDTS
      WRITE(7,*)((FT(I,J),J=1,IRFT),I=1,IRFT),((BT(I,J),J=1,ICBT),I=1,IR
     1FT),((GT(I,J),J=1,ICGT),I=1,IRFT),((HT(I,J),J=1,IRFT),I=1,IRHT),
     1((FM(I,J),J=1,IRFM),I=1,IRFM),((BM(I,J),J=1,ICBM),I=1,IRFM)
      WRITE(7,*)((GM(I,J),J=1,ICGM),I=1,IRFM),((HM(I,J),J=1,IRFM),I=1,IR
     1HM),((PO(I,J),J=1,IRFT),I=1,IRFT),(XO(I),I=1,IRFT),((WUU(I,J),
     1=J=1,ICBM),I=1,ICBM),((WXX(I,J),J=1,IRFM),I=1,IRFM)
      WRITE(7,*)((QT(I,J),J=1,ICGT),I=1,ICGT),((RT(I,J),J=1,IRHT),I=1,IR
     1HT),((QM(I,J),J=1,ICGM),I=1,ICGM),((RM(I,J),J=1,IRHM),I=1,IRHM)
      WRITE(7,*)((WXU(I,J),J=1,ICBM),I=1,IRFM)
      GO TO 982
21      WRITE(KOUT,*)' THIS OPTION READS ALL MATRICES FROM TAPE8, DO YOU
     1 WISH TO ABORT THIS OPTION,  Y  OR  N)'
      READ(KIN,97)MSG
      IF (MSG.EQ.'Y') THEN
           GO TO 982
      END IF
      WRITE(KOUT,*)'DO YOU WISH TO REWIND TAPE8 BEFORE THE READ,Y OR N)'
      READ(KIN,97)MSG
      IF (MSG.EQ.'Y') THEN
         REWIND(8)
      END IF
      READ(8,*,END=10033)IRFT,ICBT,ICGT,IRHT,IRFM,ICBM,ICGM,IRHM,NUMDTS
      READ(8,*,END=10033)((FT(I,J),J=1,IRFT),I=1,IRFT),((BT(I,J),J=1,ICB
     1T),I=1,IRFT),((GT(I,J),J=1,ICGT),I=1,IRFT),((HT(I,J),J=1,IRFT),I=1
     1,IRHT),((FM(I,J),J=1,IRFM),I=1,IRFM),((BM(I,J),J=1,ICBM),I=1,IRFM)
      READ(8,*,END=10033)((GM(I,J),J=1,ICGM),I=1,IRFM),((HM(I,J),J=1,IRF
     1M),I=1,IRHM),((PO(I,J),J=1,IRFT),I=1,IRFT),(XO(I),I=1,IRFT),((WUU(
     1 I,J),J=1,ICBM),I=1,ICBM),((WXX(I,J),J=1,IRFM),I=1,IRFM)
      READ(8,*,END=10033)((QT(I,J),J=1,ICGT),I=1,ICGT),((RT(I,J),J=1,IRH
     1T),I=1,IRHT),((QM(I,J),J=1,ICGM),I=1,ICGM),((RM(I,J),J=1,IRHM),
     1 I=1,IRHM)
      READ(8,*)((WXU(I,J),J=1,ICBM),I=1,IRFM)
      GO TO 982
10033  WRITE(KOUT,*)'END OF, FILE ENCOUNTERED DURING READ OF TAPE8'
982           CONTINUE
19         CONTINUE
      NDIM1=NSAV
      RETURN
27    IO=0
C
      END
```

```
*DECK MEIGN
      SUBROUTINE MEIGN(A,AREV,AIEV,CURSZA,WM1)
      CHARACTER MSG*60
      DIMENSIONA(NDIM,NDIM),AREV(NDIM),AIEV(NDIM),WM1(NDIM,NDIM)
      DIMENSION COM1(1),COM2(1)
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON / INOU/ KIN,KOUT,KPUNCH
      COMMON /MAIN2/ COM2
      COMMON /MAUNS/ MSG
C    THE  CALLING ROUTINE MUST SUPPLY A WORKING MATRX NDIM X NDIM --WM1
C***FIND THE EIGENVALUES OF A ,NR=0 TELLS THE ROUTINE TO CALCULATE
C                  EIGENVALUES ONLY
C  NDIM MUST BE THE DIMENSION OF A IN THE CALLING PROGRAM
C      CURSZA IS THE CURRENT SIZE OF A
       NR=0
       C1=1.0
       CALL IDNT(CURSZA,WM1,C1)
C WM1 =I    CURSZA X CURSZA
       CALL EIGEN(CURSZA,A,AREV,AIEV,WM1,NR)
       IO=5
       NSAV=NDIM1
       NDIM1=NDIM
       MSG='REAL PARTS OF THE EIGENVALUES '
       CALL MVECIO (AREV,CURSZA,IO,KIN,KOUT,NDIM)
       MSG='IMAG PARTS OF THE EIGENVALUES '
       CALL MVECIO (AIEV,CURSZA,IO,KIN,KOUT,NDIM)
C
       NDIM1=NSAV
       END
*DECK CDTCON
       SUBROUTINE CDTCON(FM,BM,WXX,WUU,GCSTR,IRFM,ICBM,WM1,WM2,
      1WM3,WM4,WM5,WM6,WXU,FPRIM,WXXPRM)
       CHARACTER MSG*60
       DIMENSIONFM(NDIM,NDIM),WXX(NDIM,NDIM),BM(NDIM,NDIM),WUU(NDIM,NDIM)
      1,GCSTR(NDIM,NDIM)
       DIMENSION WM1(NDIM,NDIM),WM2(NDIM,NDIM),WM3(NDIM,NDIM),WM4(NDIM,ND
      1IM),WM5(NDIM,NDIM),WM6(NDIM,NDIM),WXU(NDIM,NDIM),
      1 WXXPRM(NDIM,NDIM),FPRIM(NDIM,NDIM)
       DIMENSION COM1(1),COM2(1)
       COMMON /MAIN2/COM2
       COMMON /MAIN1/NDIM,NDIM1,COM1
       COMMON / INOU/ KIN,KOUT,KPUNCH
       COMMON /MAUNS/ MSG
C
C***DETERMINISTIC CONTROLLER GAIN CALCULATION---MODULE # 1
C
C                  THIS MODULE COMPUTES THE STEADY STATE DETERMINISTIC CONTROLLER
C                  GAIN MATRIX,GCSTAR=(WUUI)(BMT)(KCSSPM).  WUUI IS THE INVERSE OF T
C                  CONTROL COST WEIGHTING MATRIX, KCSSPM IS THE STEADY STATE
C                  SOLUTION TO  D(KC)/DT=(FMT)(KC)+(KC)(FM)+WXX-(KC)(BM)(WUUI)(BMT)*
C                  (KC),  WXX IS THE COST WEIGHTING MATRIX ON THE STATES
C
C                  KLIENMAN ROUTINES ARE EXTENSIVELY USED IN THIS MODULE
C  TRANSFORM SYSTEM SO THAT WXU NOT=0 CAN STILL BE HANDLED BY
C KLEINMAN ROUTINES,  SEE KWAKERNAAK AND SIVAN'S BOOK, PAGE 322
       CALL PRIMIT(WM1,WUU,ICBM,GCSTR,WXU,IRFM,BM,WM2,FPRIM,WXX,WXXPRM,
      1 FM)
C NOW HAVE FPRIM, WXXPRM CAN USE RICCATI SOLVER FOR KCSSPM
       MT=1
       CALL TRANS2(IRFM,ICBM,BM,WM2)
C WM2=BMT   ICBM X IRFM
       CALL EQUATE(WM1,WUU,ICBM,ICBM)
C  GMINV DESTROYS THE CALLING ARRAY
       CALL GMINV(ICBM,ICBM,WM1,WM3,MR,MT)
C WM3=WUUI  ICBM X ICBM------MR IS AN ERROR INDICATOR
       IF (MR.NE.ICBM) THEN
       PRINT*,'AN ERROR OCCURRED IN INVERTING WUU, MR=',MR,'ICBM=',ICBM
       END IF
       CALL EQUATE(WM1,WM3,ICBM,ICBM)
C WM1=WUUI  SAVE FOR LATER COMPUTATIONS
       CALL MAT1(WM3,WM2,ICBM,ICBM,IRFM,WM4)
C WM4=(WUUI)(BMT)  ICBM  X  IRFM
C      NOW CALL RICCATI EQUATION SOLVER
       CALL MAT1(BM,WM4,IRFM,ICBM,IRFM,WM3)
C WM3=BM(WUUI)(BMT)  IRFM  X  IRFM
       CALL MRIC(IRFM,FPRIM,WM3,WXXPRM,WM2,WM6)
C WM2=KCSSPM   IRFM X IRFM
C WM6=FM-BM(WUUI)(BMT)(KCSSPM)---I DONT USE THIS RESULT
       MSG='KCSSPM FOR THE DETERMINISTIC CONTROLLER IS'
       IO=5
```

```
      NSAV=NDIM1
      NDIM1=NDIM
      CALL MMATIO(UM2,IRFM,IRFM,IO,KIN,KOUT,NDIM,NDIM1)
C                 NOW CALCULATE OPTIMAL GAIN MATRIX GCSTAR,  NOTE  I NEED THE
C                 NEGATIVE OF GCSTAR FOR THE CONTROL LAW GENERATION FROM AN LQG
C                 CONTROLLER, AND THIS WILL BE THE GCX REQUIRED IN THE PERFORMANCE
C                 ANALYSIS ROUTINE
      NDIM1=NSAV
C NOW HAVE KCSSPM , CALCULATE GCSTR=UUUI(BMT(KCSSPM)+UXUT)
C  RECALL UUUI IN UM1
      C1=1.0
      CALL MAT4A(BM,UM2,ICBM,IRFM,IRFM,UM4)
      CALL TRANS2(IRFM,ICBM,UXU,UM3)
      CALL MADD1(ICBM,IRFM,UM4,UM3,UM2,C1)
      CALL MAT1(UM1,UM2,ICBM,ICBM,IRFM,GCSTR)
      NSAV=NDIM1
      NDIM1=NDIM
      IO=5
C GCSTR  ICBM  X  IRFM
C
      MSG='THE OPTIMAL STEADY STATE FEEDBACK GAIN MATRIX, GCSTR'
      CALL MMATIO(GCSTR,ICBM,IRFM,IO,KIN,KOUT,NDIM,NDIM1)
      NDIM1=NSAV
C
      END
*DECK CKFTR
      SUBROUTINE CKFTR(FM,GM,R,HM,NUMDTS,RKFSS,Q,UM1,UM2,
     1UM3,UM4,UM5,UM6,IRFM,IRHM,ICGM,F2,H2,BM,ICBM)
C  CALLING PROGRAM MUST SUPPLY EIGHT WORK SPACE ARRAYS
C
CTHIS ROUTINE CALCULATES  THE KALMAN FILTER GAINS WHEN
C GIVEN THE FM,HM, GM AND R MATRICES AND THE NUMBER OF
C  DETERMINISTIC STATES.   THE CONTROLLER MODEL MUST BE
C  SPECIFIED SUCH THAT ALL THE DETERMINISTIC STATES APPEAR
C  FIRST AND TOGETHER,   THAT IS
CD/DT(X1,X2,...XK,XL,XM,...XN)T=[F1  0 ; 0  F2][B1 ; B2]U+[0 ; G2]W
C
C
C  WHERE X1 THROUGH XK ARE THE DETERMINISTIC STATES AND THE
C  REMAINING STATES ARE STOCHASTIC.  F1 IS K  X  K, AND F2 IS N-K  X
C  N-K,  AND B1,B2,AND G2 ARE PARTITIONED ACCORDINGLY.
C  THIS ROUTINE FIRST STRIPS OFF THE DETERMINSTIC STATES THEN COMPUTES
CAND RETURNS  KALMAN FILTER GAINSFOR THE REMAINING STATES.
C  THE KALMAN FILTER GAINS FOR THE DETERMINSTIC STAES ARE SET TO ZERO
C AND THE KALMAN FILTER GAIN THAT IS RETURNED IS   0
C
C WHERE THE DIMENSION OF THE ZERO VECTOR IS K AND THE RKFSS IS THE
C  STEADY STATE KALMAN FILTER GAIN MATRIX FOR THE N-K STOCHASTIC STATES.
C  THIS AUGMENTED MATRIX IS RETUTNED IN RKFSS
C  ALSO NOTE THAT IN ORDER TTO GENERATE THE KALMAN FILTER, ONLY
C MEASUREMENTS OF STACHASTIC STATES ARE NEEDED SO THE H MATRIX IS
C REDUCED ACCORDDINGLY.
      CHARACTER MSG*60,MSG1*1
      DIMENSION F2(NDIM,NDIM),H2(NDIM,NDIM),FM(NDIM,NDIM),GM(NDIM,NDIM),
     1  R(NDIM,NDIM),HM(NDIM,NDIM),UM1(NDIM,NDIM),Q(NDIM,NDIM),
     1  UM2(NDIM,NDIM),UM3(NDIM,NDIM),UM4(NDIM,NDIM),UM5(NDIM,NDIM)
     1  ,UM6(NDIM,NDIM)
      REAL RKFSS(NDIM,NDIM),BM(NDIM,NDIM)
      DIMENSION COM1(1),COM2(1)
      COMMON /MAIN2/COM2
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON / INOU/ KIN,KOUT,KPUNCH
      COMMON /MAUN5/ MSG
C
CXXXKALMAN FILTER STEADY STATE GAIN----MODULE # 2
C
C     RKFSS=(PMSS)(HMT)(RI), WHERE PM IS THE STEADY STATE SOLUTION TO TH
C     RICCATI EAUATION D(PM)/DT=FM(PM)+PM(FMT)+GM(Q)(GMT)-
C                 PM(HMT)(RI)(HM)(PM)
C
C
C  XXXXXXXDELETE DETERMINISTIC STATES,  AND ---TRANSPOSE THE F2
C  MATRIX FOR THE RICATTI SOLVER SINCE IT TRANSPOSES THE CALLING ARRAY)
      WRITE(KOUT,X)'IF YOU PLAN TO USE THE DOYLE AND STEIN TECHNIQUE FO
     1R THIS RUN YOU MAY WISH TO MODIFY THE VALUE OF NUMDTS,  THE NUMBER
     1 OF DETERMINISTIC STATES.  DO YOU WANT TO CHANGE NUMDTS?  Y  OR  N
     1)'
      READ(KIN,11)MSG1
      NUMSAV=NUMDTS
      IF (MSG1.EQ.'Y') THEN
      WRITE(KOUT,X)'ENTER THE NEW VALUE OF NUMDTS FOR THIS RUN)'
      READ(KIN,X)NUMDTS
      END IF
```

```
      IDS=NUMDTS+1
      DO 2112 I=IDS,IRFM
      II=I-NUMDTS
      DO 2112 J=IDS,IRFM
      JJ=J-NUMDTS
2112  F2(JJ,II)=FM(I,J)
      IRF2=IRFM-NUMDTS
      DO 2113 I=1,IRHM
      DO 2113 J=IDS,IRFM
      JJ=J-NUMDTS
2113  H2(I,JJ)=HM(I,J)
C  NOW FORM B2,G2
      DO 2114 I=IDS,IRFM
      II=I-NUMDTS
      DO 2114 J=1,ICGM
2114  UM1(II,J)=GM(I,J)
C UM1 =G2    IRF2  X  ICGM
      DO 2115 I=IDS,IRFM
      II=I-NUMDTS
      DO 2115 J=1,ICBM
2115  UM4(II,J)=BM(I,J)
C UM4 =B2  IRF2  X  ICBM
      IRH2=IRHM
      CALL MAT3(IRF2,ICGM,UM1,Q,UM2)
C UM2=GM(Q)(GMT)  IRF2 X IRF2 --USED AS 'Q'IN KLIENMAN RICCATI ROUTINE
      WRITE(KOUT,1)'DO YOU WISH TO MODIFY Q BY THE DOYLE AND STEIN TECHNI
     1QUE,  Y  OR N>'
      READ (KIN,11)MSG1
11     FORMAT(A1)
      IF (MSG1.EQ.'Y') THEN
        CALL DAS1(UM2,UM4,UM1,ICBM,UM3,IRFM)
      END IF
      MT=1
      CALL EQUATE(UM1,R,IRH2,IRH2)
C  GMINV DESTROYS THE CALLING ARRAY
      CALL GMINV(IRH2,IRH2,UM1,UM3,MR,MT)
      IF (MR.NE.IRH2) THEN
      WRITE(KOUT,1)'MR=',MR,'IRH2=',IRH2
      WRITE(KOUT,1)'R-INVERSE IS FOULED UP'
      END IF
C UM3= RI   IRH2 X IRH2
      CALL TRANS2(IRH2,IRF2,H2,UM4)
C UM4= H2T IRF2 X IRH2
      CALL MAT1(UM4,UM3,IRF2,IRH2,IRH2,UM5)
C UM5= H2T(RI)    IRF2 X IRH2
      CALL MAT1(UM5,H2,IRF2,IRH2,IRF2,UM3)
C UM3= H2T(RI)(H2)    IRF2 X IRF2
      CALL MRIC(IRF2,F2,UM3,UM2,UM6,UM4)
C  NOW CALL RICCATI EQUATION SOLVERTO GET PMSS
C UM6=PMSS    IRF2 X IRF2
      CALL MAT1(UM6,UM5,IRF2,IRF2,IRH2,UM1)
C UM1=RKFSS  IRF2  X IRH2
      IO=5
C  FORM RKFSS WITH ZEROS ADDED FOR DETER. STATES.
      PRINT*, 'NUMDTS=',NUMDTS
      IF (NUMDTS.NE.0)THEN
      DO 2119 J=1,IRHM
      DO 2118 I=1,NUMDTS
2118  RKFSS(I,J)=0
      DO 2119 I=IDS,IRFM
      II=I-NUMDTS
2119  RKFSS(I,J)=UM1(II,J)
      ELSE
      CALL EQUATE(RKFSS,UM1,IRFM,IRHM)
      END IF
      MSG='STEADY STATE KALMAN FILTER GAIN MATRIX,RKFSS'
      CALL MMATIO(RKFSS,IRFM,IRHM,IO,KIN,KOUT,NDIM,NDIM)
      NUMDTS=NUMSAV
C
      END
*DECK FRMAUG
      SUBROUTINE FRMAUG(Q,R,FT,BT,GCZ,HT,GCX,BCZ,FC,GCV,BCV,GT,XO,PO,
     1 FA,BA,GA,GA,GUA,UM1,UM2,UMA,UMB,UMC,UMD,UME,UMF,
     1  MXA,PXA,RA,PXUA,GCZA,IRY,IFLGCZ,IFLGSD)
C  THIS ROUTINE FORMS A SET OF AUGMENTED MATRICES NEEDED BY THE
C   PERFORMANCE ANALYSIS ROUTINES
      DIMENSION Q(NDIM,NDIM),R(NDIM,NDIM),FT(NDIM,NDIM)
```

```
     1,BT(NDIM,NDIM),GCZ(NDIM,NDIM),HT(NDIM,NDIM),GCX(NDIM,NDIM),
     1  BCZ(NDIM,NDIM),FC(NDIM,NDIM),BCY(NDIM,NDIM),GT(NDIM,NDIM),
     1  XO(NDIM),PO(NDIM,NDIM),UM1(NDIM,NDIM),UM2(NDIM,NDIM)
      DIMENSION FA(NDIM2,NDIM2),BA(NDIM2,NDIM2),GA(NDIM2,NDIM2),
     1 QA(NDIM2,NDIM2),UMA(NDIM2,NDIM2),UMB(NDIM2,NDIM2),
     1  UMC(NDIM2,NDIM2),UMD(NDIM2,NDIM2),UME(NDIM2,NDIM2),UMF
     1 (NDIM,NDIM),GCY(NDIM,NDIM),PXA(NDIM2,NDIM2),RA(NDIM2,NDIM2),
     1 -GCZA(NDIM2,NDIM2),PXUA(NDIM2,NDIM2),GUA(NDIM2,NDIM2)
      REAL MXA(NDIM2)
      INTEGER IFLGCZ
      CHARACTER MSG*60
      DIMENSION COM1(1),COM2(1)
      COMMON /MAIN4/NDIM2,NDIM3
      COMMON /MAIN2/COM2
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON / INOU/ KIN,KOUT,KPUNCH
      COMMON /MAUMS/ MSG
      COMMON /MAIN6/ ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1    IRHT,IRQA,IO,LOG,IRHM,NUMDTS
      URITE(KOUT,*)' ENTER A 5 IF YOU WANT ALL THE AUGMENTED MATRICES PR
     1INTED OUT, A 6 FOR NO MATRICES TO BE PRINTED>'
      READ(KIN,*)IO
      IRFA=IRFT+IRFM
      NSAV1=NDIM
      NSAV2=NDIM1
      NSAV3=NDIM2
      NSAV4=NDIM3
C
C***FORM AUGMENTED MATRICES THAT ARE REQUIRED WHEN FORMING XA
C     UA=(UT  VT)T   IMPLIES THAT QA=   Q  0
C
C
C    FORM QA  IRQA  X  IRQA,  IRQA=IRHT+ICGM
C FOR EQUIVALENT DISCRETE TIME SYSTEMS    IRQA= IRFT+IRHT
      IF(IFLGSD.EQ.0)THEN
      IRQ=ICGT
      ELSE
      IRQ=IRFT
       END IF
      DO 2703 I=1,IRQ
      DO 2703 J=1,IRHT
2703  UM1(I,J)=0
     -DO 2704 I=1,IRHT
     -DO 2704 J=1,IRQ
2704  UM2(I,J)=0
      IFORM=1
      NDIM3=NSAV3
      NDIM2=NSAV1
      CALL AUGMAT(UM2,R,UMD,IFORM,IRHT,IRQ,IRHT,IRHT)
      CALL AUGMAT(Q,UM1,UMC,IFORM,IRQ,IRQ,IRQ,IRHT)
      ICQA=IRQ+IRHT
      IRQA=ICQA
       IFORM=2
      NDIM2=NSAV3
      CALL AUGMAT(UMC,UMD,QA,IFORM,IRQ,ICQA,IRHT,IRQA)
      MSG=' THE AUGMENTED Q MATRIX IS ,QA'
      CALL MMATIO(QA,IRQA,IRQA,IO,KIN,KOUT,NSAV3,NSAV3)
      IMA=NSAV3-IRFT
C     INITIALIZE PXA MXA AND STORAGE VARIABLES
      DO 5105 IMXA=1,NSAV3
5105  MXA(IMXA)=0
      DO 5100 IMXA=1,IRFT
5100  MXA(IMXA)=XO(IMXA)
      MSG='  THE INITIAL XA VECTOR IS'
      CALL MVECIO(MXA,IRFA,IO,KIN,KOUT,NSAV3)
      DO 5101 IPXA=1,IRFT
      DO 5102 JPXA=1,IRFT
5102  PXA(IPXA,JPXA)=PO(IPXA,JPXA)
      DO 5101 JPX=1,IMA
      JPXA=JPX+IRFT
5101  PXA(IPXA,JPXA)=0
      DO 5103 IPX=1,IMA
      IPXA=IPX+IRFT
      DO 5103 JPXA=1,IRFA
5103  PXA(IPXA,JPXA)=0
      MSG=' THE INTITIAL COVARIANCE MATRIX, PXA IS'
      CALL MMATIO(PXA,IRFA,IRFA,IO,KIN,KOUT,NSAV3,NSAV3)
```

```
C***PXUA CALCULATION--- REQUIRED ONLY FIRST TIME THROUGH LOOP
C                    PXUA=    BT(GCZ)
C                                1/2 BCZ  R
C
C     TO USE THE KLIENMAN MULTIPLY ROUTINES , THE DECLARED DIMENSION OF
C               ARRAY ARGUMENTS MUST BE THE SAME.  THEREFORE IT IS NECESSARY TO
C               FORM GCZA SUCH THAT GCZA(I,J)=GCZ(I,J) FOR I=1,IRHT,  AND J=1,
C               ICBM,  AND ZERO ELSEWHERE
C     THE SAME REASON REQUIRES CALCULATION OF RA
      IT=NSAU3-IRHT
      JT=NSAU3-ICBM
      DO 6013 IG=1,IRHT
      DO 6014 JG=1,ICBM
6014  GCZA(IG,JG)=GCZ(IG,JG)
      DO 6013 JG=1,JT
      JGA=ICBM+JG
6013  GCZA(IG,JGA)=0
      DO 6015 IG=1,IT
      IGI=IG+IRHT
      DO 6015 JG=1,NSAU3
6015  GCZA(IGI,JG)=0
      IR=NSAU3-IRHT
      DO 6017 IRI=1,IRHT
      DO 6017 JRI=1,IRHT
6017  RA(IRI,JRI)=R(IRI,JRI)
      DO 6016 JRI=1,IR
      JRJ=JRI+IRHT
6016  RA(IRI,JRJ)=0
      DO 6018 IRI=1,IR
      IRII=IRI+IRHT
      DO 6018 JRI=1,NSAU3
6018  RA(IRII,JRI)=0
C    RA = R IN UPPER LEFT PARTITION,ZERO ELSEWHERE
      IF((IFLGCZ.EQ.0).AND.(IFLGSD.EQ.0)) THEN
C CALCULATE PXUA ONLY FOR GCZ NOT  EQUAL TO ZERO MATRIX AND NOT FOR S-D
C  RECALL THAT (BT(GCZ)  BCZ)T  IS THE RIGHR PARTITION OF GA
      DO 6000 IPXA=1,IRHT
      IPA=IPXA+ICGT
      DO 6000 JPXA=1,IRFA
6000  PXUA(JPXA,IPXA)=GA(JPXA,IPA)
      C1=.5
      NDIM=NSAU3
      NDIM1=NSAU3+1
      CALL SCALE(UMF,PXUA,IRFA,IRHT,C1)
      CALL MAT1(UMF,RA,IRFA,IRHT,IRHT,PXUA)
C PUXA=PXAUT   IRFA  X  IRHT
      MSG=' CROSS COVARIANCE, PXUA IS'
      CALL MMATIO(PXUA,IRFA,IRHT,IO,KIN,KOUT,NSAU3,NSAU3)
      END IF
C
C
C                   FA= FA11  FA12  *FT+BT(GCZ)(HT)                BT(GCX)
C                     *FA21   FA22  *BCZ(HT)                                FC
C
C  WHERE GCZ IS THE GAIN MATRIX THAT ACTS DIRECTLY ON THE MEASUREMENT
C               VECTOR,  AND GCX IS THE GAIN MATRIX THATS ACTS ON THE CONTROLLER
C               STATE ESTIMATES.*****THESE MUST BE SUPPLIED BY THE GAIN MATRIX
C               ROUTINE----------------C
C
C***FORM FA
      NDIM1=NSAU1+1
      NDIM=NSAU1
C
      CALL MAT1(BT,GCZ,IRFT,ICBT,IRHT,UM1)
C UM1=BT(GCZ) IRFT X IRHT



      CALL MAT1(UM1,HT,IRFT,IRHT,IRFT,UM2)
C UM2=  BT(GCZ)HT   IRFT X IRFT
      C1=1.0
      CALL MADD1(IRFT,IRFT,FT,UM2,UM1,C1)
C UM1=  FA11   IRFT X IRFT
      IF (ICBM.NE.ICBT)THEN
      WRITE(KOUT,*)'ICBM=',ICBM,' ICBT=',ICBT
      WRITE(KOUT,*)'BT AND BM ARE NOT THE SAME SIZE- WILL CAUSE ERRORS'
      END IF
      CALL MAT1(BT,GCX,IRFT,ICBT,IRFM,UM2)
C UM2=FA12     IRFT XIRFM
```

```
      IFORM=1
      NDIM2=NSAV1
      CALL AUGMAT(UM1,UM2,UMA,IFORM,IRFT,IRFT,IRFT,IRFM)
C UMA= (FA11  FA12)   IRFT X IRFT+IRFM
      CALL MAT1(BCZ,HT,IRFM,IRHT,IRFM,UM1)
C UM1= FA21   IRFM  X  IRFT
      CALL AUGMAT(UM1,FC,UMB,IFORM,IRFM,IRFT,IRFM,IRFM)
C UMB= (FA21  FA22)   IRFM  X IRFM+IRFT
      NDIM2=NSAV3
      IFORM=2
      IRFA=IRFT+IRFM
      ICFA=IRFA
      CALL AUGMAT(UMA,UMB,FA,IFORM,IRFT,IRFA,IRFM,ICFA)
      MSG='THE AUGMENTED F MATRIX FA IS'
      CALL MMATIO(FA,IRFA,IRFA,IO,KIN,KOUT,NSAV3,NSAV3)
C FA   IRFA  X  IRFA
C
CXXXFORM BA - - - FOR REGULATOR CASE , NOT REQUIRED Y=0
       CALL MAT1(BT,GCY,IRFT,ICBT,IRY,UM1)
C  UM1= BT(GCY)  IRFT  X  IRY
      IFORM=2
      NDIM2=NSAV1
      CALL AUGMAT(UM1,BCY,BA,IFORM,IRFT,IRY,IRFM,IRY)
      MSG=' THE AUGMENTED B MATRIX BA,IS'
      CALL MMATIO(BA,IRFA,IRY,IO,KIN,KOUT,NSAV3,NSAV3)
C   BA IRFA  X  IRY
      CALL AUGMAT(UM1,BCY,BA,IFORM,IRFT,IRY,IRFM,IRY)
      MSG=' THE AUGMENTED B MATRIX BA,IS'
      CALL MMATIO(BA,IRFA,IRY,IO,KIN,KOUT,NSAV3,NSAV3)
C   BA IRFA  X  IRY

C
CXXXFORM GA
C
C   GA=  GT                    BT(GCZ)
C      =  0                     BCZ
C
      CALL MAT1(BT,GCZ,IRFT,ICBM,IRHT,UM1)
C UM1= BT(GCZ)  IRFT  X  IRHT
      IFORM=1
      CALL AUGMAT(GT,UM1,UMC,IFORM,IRFT,IRQ,IRFT,IRHT)
C  RECALL IRQ=ICGT FOR CONTINUOUS SYS,=IRFT FOR S-D SYS
C UMC= (GT   BT(GCZ))   IRFT  X  IRHT+IRQ
      DO 3001 IR=1,IRFM
      DO 3001 IC=1,IRQ
3001  UM1(IR,IC)=0
      CALL AUGMAT(UM1,BCZ,UMD,IFORM,IRFM,IRQ,IRFM,IRHT)
C UMD=(0   BCZ)    IRFM  X IRQ+IRHT
      ICGA=IRQ+IRHT
      IFORM=2
      NDIM2=NSAV3
      CALL AUGMAT(UMC,UMD,GA,IFORM,IRFT,ICGA,IRFM,ICGA)
      MSG=' THE AUGMENTED  G MATRIX GA , IS'
      CALL MMATIO(GA,IRFA,ICGA,IO,KIN,KOUT,NSAV3,NSAV3)
C GA   IRFA  X  ICGA
C
C   GUA=GCZ(HT)   GCX)
      NDIM=NSAV1
      NDIM1=NSAV1+1
      CALL MAT1(GCZ,HT,ICBT,IRHT,IRFT,UM2)
C UM2= GCZ(HT)    ICBT X IRFT
      IFORM=1
      NDIM2=NSAV1
      NDIM3=NSAV3
      CALL AUGMAT(UM2,GCX,GUA,IFORM,ICBT,IRFT,ICBM,IRFM)
      MSG='THE AUGMENTED MATRIX GUA IS'
      CALL MMATIO(GUA,ICBT,IRFA,IO,KIN,KOUT,NSAV3,NSAV3)
C  GUA  ICBT  X  IRFA
CXXXXXXAUGMENTED SYSTEM MATRICES NOW AVAILBLE FOR COMPUTATION
C
      NDIM=NSAV1
      NDIM1=NSAV2
      NDIM2=NSAV3
      NDIM3=NSAV4
      END
*DECK PERFAL
      SUBROUTINE PERFAL(IRY,IFLGCZ,MXA,GCY,GUA,PXA,PXUA,IFLGSD,
     1  RA,GCZA,YD,EAT,INTGA,UME,UMF,PUU,UM1,PUOUT,MXTOUT,PUOUT,PXTOUT,
     1  MXAMIN,MXAMAX,PXTMIN,PXTMAX,MUMIN,MUMAX,PUMIN,PUMAX,MU,INTBA,
```

```
     1 UU3,UU4)
      CHARACTER MSG*60
      REAL UM1(NDIM,NDIM),EAT
     1   (NDIM2,NDIM2),INTGA(NDIM2,NDIM2),UME(NDIM2,NDIM2),UMF(NDIM2,
     1NDIM2),UU3(NDIM2),UU4(NDIM2),
     1    MXA(NDIM2),PXA(NDIM2,NDIM2),PXUA(NDIM2,
     1    NDIM2),MUOUT(NDIM),MXTOUT(NDIM),PXTOUT(NDIM),PUOUT(NDIM),
     1    YD(NDIM3),MXAMIN(NDIM),MXAMAX(NDIM),MUMIN(NDIM),MUMAX(NDIM),
     1    PXTMIN(NDIM),PXTMAX(NDIM),PUMIN(NDIM),PUMAX(NDIM),GCZA
     1    (NDIM2,NDIM2),RA(NDIM2,NDIM2),GUA(NDIM2,NDIM2),GCV(NDIM,
     1    NDIM),INTBA(NDIM2,NDIM2)
      INTEGER IFLGCZ
      DIMENSION COM1(1),COM2(1)
      REAL MU(NDIM),PUU(NDIM2,NDIM2)
      COMMON /RNTIM/ RNTIME,DELTIM
      COMMON /MAIN4/NDIM2,NDIM3
      COMMON /MAIN2/COM2
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON / INOU/ KIN,KOUT,KPUNCH
      COMMON /MAUN5/ MSG
      COMMON /MAIN6/ ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1    IRHT,IRQA,IO,LQG,IRHM,NUMDTS
      NSAV1=NDIM
      NSAV2=NDIM1
      NSAV3=NDIM2
      NSAV4=NDIM3
C
CXXXXPERFORMANCE ANALYSIS ROUTINE
C     THIS IS A CONTINUOUS TIME MEASUREMENT PERFORMANCE ANALYSIS
C     ROUTINE FOR EVALUATING CONTINUOUS TIME CONTROL SYSTEMS DRIVEN BY
C      WHITE GAUSSIAN NOISE.   IT COMPUTES THE MEAN AND COVARIANCE OF THE
C      OF THE TRUTH MODEL STATES ,THE CONTROLLER STATES,AND THE CONTROLS
C      GENERATED.    A SET OF AUGMENTED MATRICES IS USED TO DO THE
C                  CALCULATIONS---Y=(XT  USTAR)T,    XA=(XT  XM)T .. THE PERFORMANCE
C                  ANALYSIS ROUTINE IS DEVELOPED IN A MASTERS THESES FOR AIR FORCE
C                  INSTITUTE OF TECHNOLOGY BY ERIC LLOYD,  TITLE 'ROBUST CONTROL
C                  SYSTEM DESIGN'
C
C
CXXXMXA,PXA CALCULATION--- THE MEAN  AND COVARIANCE OF THE XA VECTOR
C                    FOUND USING SOLUTION FORMS OF THE PROPAGATION EQUATONS
C                  KLIENMAN ROUTINES ARE USED TO PROVIDE THE SOLUTIONS
C   IN THE FOLLOWING TWO EQS, THE FIRST OCCURRANCE OF PXA OR MXA
C   IS THE VALUE AT TIME T+DELTIM, THE SECOND ----AT TIME T
C                       PXA=EAT(PXA)EATT+INTGA
C                                       MXA=EAT(MXA)+INTBA
C                  SEE DEFINITIONS BELOW FOR EAT ,INTGA,  INTBA
C
C                    NOTE SINCE THIS PROGRAM CONSIDERS.ONLY THE REGULATOR CASE,
C                     Y- THE DESIRED INPUT-  IS ASSUMED = ZERO
C
C                    MXA0=(E(X0)  0)T  =(X0  0)T    E=THE EXPECTED VALUE OPERATOR
C                    PXA0=   P0  0
C                                    0   0
C
C    EAT=   EXP(FA*TIME)
C    INTBA=  INTEG(EAT)BA     FOR CONTINUOUS TIME SYSTEMS
C                 =  BAD FOR DISCRETE SYSTEMS
C    INTGA=    INT(EAT(GA)QA(GAT)EATT)  FOR CONT TIME SYSTEMS
C                   =  GDA(QDA)GDAT     FOR DISCRETE TIME SYSTEMS
C
       LCOUNT=0
       IRN=NINT(RNTIME/DELTIM)
       WRITE(KOUT,*)'ENTER A  0  IF YOU WANT NO PRINTS OF PXA, PUU, MXA,
      1 AND MU MATRICES DURING THE PERFORMANCE ANALYSIS,  ELSE ENTER THE
      1 NUMBER OF TIME INCREMENTS BETWEEN PRINTS( THERE ARE ',IRN,' TOTAL
      1 TIME INCREMENTS IN THIS RUN))'
       READ(KIN,*)IPCNTL
       IF (IPCNTL.EQ.0) THEN
       IO=0
       ELSE
       IO=5
       END IF
       DO  5003 IN=1,NSAV1
       PUU(IN,IN)=0
       MU(IN)=0
```

```
      MXAMIN(IN)=0
      MXAMAX(IN)=0
      PXTMIN(IN)=0
      PXTMAX(IN)=0
      MUMIN(IN)=0
      MUMAX(IN)=0
      PUMAX(IN)=0
      PUMIN(IN)=0
5003  CONTINUE
      WRITE(KOUT,*)'ENTER THE NUMBER OF SAMPLE PERIODS DESIRED BETWEEN
     1 PLOT POINTS(MAX 1000 PLOT POINTS) THERE ARE ',IRN,' SAMPLE PERIOD
     1S REQUESTED FOR THIS RUN)'
      READ(KIN,*)IPLTPS
      DELPLT=DELTIM*IPLTPS
      DO 5000 ITLMP=1,IRN
C
      IF(IO.EQ.5) THEN
      JJ=ITLMP-1
      IPNT=MOD(JJ,IPCNTL)
      IF((IPNT.EQ.0).OR.(ITLMP.EQ.IRN)) THEN
      TIME=JJ*DELTIM
      WRITE(KOUT,*)'TIME=  ',TIME
      MSG='   PXA'
      CALL MMATIO(PXA,IRFA,IRFA,IO,KIN,KOUT,NSAV3,NSAV3)
      MSG='   PUU'
      CALL MMATIO(PUU,ICBM,ICBM,IO,KIN,KOUT,NSAV3,NSAV3)
      MSG='MXA'
      CALL MVECIO(MXA,IRFA,IO,KIN,KOUT,NSAV3)
      MSG='MU'
      CALL MVECIO(MU,ICBM,IO,KIN,KOUT,NSAV1)
      END IF
      END IF
C     NOW WANT TO STORE FOR PLOTTING, MXT,PXX,MU,PUU
      DO 123 IWR=1,IRFT
      MXTOUT(IWR)=MXA(IWR)
      PXTOUT(IWR)=PXA(IWR,IWR)
      MXAMIN(IWR)=MIN(MXAMIN(IWR),MXA(IWR))
      MXAMAX(IWR)=MAX(MXAMAX(IWR),MXA(IWR))
      PXTMAX(IWR)=MAX(PXTMAX(IWR),PXA(IWR,IWR))
123   PXTMIN(IWR)=MIN(PXTMIN(IWR),PXA(IWR,IWR))
      DO 124 IWR=1,ICBM
      MUOUT(IWR)=MU(IWR)
      PUOUT(IWR)=PUU(IWR,IWR)
      MUMIN(IWR)=MIN(MUMIN(IWR),MU(IWR))
      MUMAX(IWR)=MAX(MUMAX(IWR),MU(IWR))
      PUMAX(IWR)=MAX(PUMAX(IWR),PUU(IWR,IWR))
124   PUMIN(IWR)=MIN(PUMIN(IWR),PUU(IWR,IWR))
C*******
C***** NO CROSS CORRELATION TERMS ARE PLOTTED
C
      NDIM6=NSAV1
      IPTCTL=MOD(JJ,IPLTPS)
      IF (IPTCTL.EQ.0) THEN
      CALL STORED(LOG,RNTIME,DELPLT,LCOUNT,IRFT,IRFT,ICBM,ICBM,
     1  MXTOUT,PXTOUT,MUOUT,PUOUT,NDIM6)
      LCOUNT=LCOUNT+1
      END IF
C  NOTE THAT YD IS RESTRICTED BY VALUE OF LCOUNT TO BE CONSTANT
C  BETWEEN PLOT POINTS
      IF (LCOUNT.GT.1000)THEN
C RESET LCOUNT
      LCOUNT=1000
      END IF
C***UPDATE PXA,MXA
C
      NDIM=NSAV3
      NDIM1=NSAV3+1
      CALL MAT3(IRFA,IRFA,EAT,PXA,WME)
      C1=1.0
      CALL MADD1(IRFA,IRFA,WME,INTGA,PXA,C1)
C***PXA AT NEW TIME NOW AVAILABLE
C  MXA= EAT(MXA0)+INTEG(EAT(BA))(YD)
      DO 1814 IK=1,IRFA
1814  WW3(IK)=0
      DO 1815 IK=1,IRFA
      DO 1815 IJ=1,IRFA
```

```
1815  UV3(IK)=UV3(IK)+EAT(IK,IJ)*MXA(IJ)
      DO 1812 IMR=1,IRFA
1812  UV4(IMR)=INTBA(IMR,1)*YD(LCOUNT)
      DO 1813 IJ=1,IRFA
1813  MXA(IJ)=UV3(IJ)+UV4(IJ)
C***MXA AT NEW TIME NOW AVAILABLE
C***MU,PUU CALCULATION FOR ZERO MEAN MEASUREMENT NOISE
C   MU=GUA(MXA)+GCZ(MUT)+GCY(YD)....MUT ,THE MEAN OF NOISE U ASSUMED =0
      X1=YD(LCOUNT)
      NDIM=NSAV1
      NDIM1=NSAV1+1
      CALL SCALE(UM1,GCY,ICBM,IRY,X1)
C UM1= GCY(YD                    ICBM  X  1
      NDIM=NSAV3
      NDIM1=NSAV3+1
      DO 1817 IJ=1,ICBM
1817  UV3(IJ)=0
      DO 1816 IJ=1,ICBM
      DO 1816 IK=1,IRFA
1816  UV3(IJ)=UV3(IJ)+GUA(IJ,IK)*MXA(IK)
C UV3=GUA(MXA)   ICBM  X  1
C   ADDED TO UM1 ABOVE TO GET MU
      DO 329 I=1,ICBM
329   MU(I)=UV3(I)+UM1(I,1)
C MU   IRFA  X 1  ----NOW AVAILABLE------
C***PUU CALCULATION,PUU=GUA(PXA)GUAT+GUA(PXUA)GCZT+GCZ(PXUA)GUAT+
C                    GCZ(R)GCZT
      CALL MAT3(ICBM,IRFA,GUA,PXA,PUU)
C PUU=GUA(PXA)GUAT  ICBM  X  ICBM
C******IFLGCZ*****
      IF (IFLGCZ.EQ.0) THEN
C                 SINCE GCZ NOT EQUAL TO ZERO CALCULATE OTHER TERMS OF PUU
C******IFLGCZ******



      IF (IFLGSD.EQ.0) THEN
C DONT DO THIS FOR S-D CASE
      CALL MAT1(GUA,PXUA,ICBM,IRFA,IRHT,UMF)
      CALL MAT4(UMF,GCZA,ICBM,IRHT,ICBM,UME)
      C1=1.0
      CALL MADD1(ICBM,ICBM,PUU,UME,UMF,C1)
C UMF=GUA(PXA)GUAT+GUA(PXUA)GCZT
      CALL MAT4(GCZA,PXUA,ICBM,IRHT,IRFA,PUU)
      CALL MAT4(PUU,GUA,ICBM,IRFA,ICBM,UME)
      CALL MADD1(ICBM,ICBM,UME,UMF,PUU,C1)
C PUU=GUA(PXA)GUAT+GUA(PXUA)GCZT+GCZ(PUXA)GUAT
      END IF
      CALL MAT3(ICBM,IRHT,GCZA,RA,UME)
      CALL MADD1(ICBM,ICBM,UME,PUU,UMF,C1)
      CALL EQUATE(PUU,UMF,ICBM,ICBM)
      END IF
C*** PUU NOW AVAILABLE  ICBM  X  ICBM
C
C
C
5000  CONTINUE
      CALL STORED(LOG,RNTIME,DELTIM,LCOUNT,IRFT,IRFT,ICBM,ICBM,
     1 MXTOUT,PXTOUT,MUOUT,PUOUT,NDIM6)
C                STORE MIN VALUE AND SCALE FACTOR IN  2 LOCATIONS FOLLOWING
C                   DATA VALUES --FOR USE IN PLOT ROUTINE
C                  NOTE THAT 3.5 INCHES IS CHOSEN HERE AS THE AXIS LENGTH
      CALL STORED(LOG,RNTIME,DELTIM,LCOUNT,IRFT,IRFT,ICBM,ICBM,
     1   MXAMIN,PXTMIN,MUMIN,PUMIN,NDIM6)
      DO 125 IUR=1,IRFT
      MXTOUT(IUR)=(MXAMAX(IUR)-MXAMIN(IUR))/3.5
125   PXTOUT(IUR)=(PXTMAX(IUR)-PXTMIN(IUR))/3.5
      DO 126 IUR=1,ICBM
      MUOUT(IUR)=(MUMAX(IUR)-MUMIN(IUR))/3.5
126   PUOUT(IUR)=(PUMAX(IUR)-PUMIN(IUR))/3.5
      CALL STORED(LOG,RNTIME,DELTIM,LCOUNT,IRFT,IRFT,ICBM,ICBM,
     1   MXTOUT,PXTOUT,MUOUT,PUOUT,NDIM6)
C                CALL TO STORED  WITH LCOUNT < 0 INDICATE THIS DATA RUN  COMPLETE
      LCOUNT=-12
      CALL STORED(LOG,RNTIME,DELTIM,LCOUNT,IRFT,IRFT,ICBM,ICBM,
     1   MXTOUT,PXTOUT,MUOUT,PUOUT,NDIM6)
      NDIM=NSAV1
      NDIM1=NSAV2
      NDIM2=NSAV3
      NDIM3=NSAV4
```

```
C   RESET IO TO SOME NONZERO  VALUE TO AVOID TERMINATING THE PROGRAM
C   WHEN RETURNING  TO MAIN ROUTINE,  LOGRP
      IO=25
      END
*DECK MYPLOT
      SUBROUTINE MYPLOT
      END
*DECK AUGMAT
      SUBROUTINE AUGMAT(A1,A2,A3,IFORM,IRA1,ICA1,IRA2,ICA2)
C
C***NDIM2,NDIM3 MUST BE SET IN THEM CALLING PROGRAM BEFORE USING
C***NDIM2,NDIM3 MUST BE SET IN THE CALLING PROGRAM BEFORE USING
C*****   THIS SUBROUTINE.   THEY MUST BE DECLARED IN A COMMON BLOCK
C              LABELED --MAIN4---
C     THIS SUBROUTINE FORMS AUGMENTED MATRICES OF THE FORM
C                  IFORM=1                    A3=(A1  A2)
C                  IFORM=2                    A3=(A1  A2)T
C
C
C     IRA1,IRA2,ARE ROW DIMENSIONS,ICA1,ICA2,ARE COLUMN DIMENSIONS
      DIMENSION A1(NDIM2,NDIM2),A2(NDIM2,NDIM2),A3(NDIM3,NDIM3)
      COMMON /MAIN4/NDIM2,NDIM3
      IF (IFORM.EQ.1) THEN
C  FORM THE AUGMENTED MATRIX A3=(A1  A2)
      DO 12 II=1,IRA1
      DO 11 III=1,ICA1
11    A3(II,III)=A1(II,III)
      DO 12 IV=1,ICA2
      IVI=IV+ICA1
12    A3(II,IVI)=A2(II,IV)
      RETURN
      END IF
C FORM AUGMENTED MATRIX A3=(A1  A2)T
      IRA3=IRA1+IRA2
      ICA3=ICA1
      DO 14 II=1,ICA3
      DO 13 III=1,IRA1
13    A3(III,II)=A1(III,II)
      DO 14 IV=1,IRA2
      IVI=IV+IRA1
14    A3(IVI,II)=A2(IV,II)
      RETURN
      END
*DECK MVECIO
      SUBROUTINE MVECIO(A,NUMEL,IO,KIN,KOUT,NDIM)
C   THIS SUBROUTINE READS PRINTS ENTIRE (PORTIONS OF) THE VECTOR
C   A, DEPENDING ON THE VALUE OF ---IO---.   IO=1---READ ONLY
C   IO=2---READ AND PRINT,   IO=3 READ SELECTED VALUE,  IO=4
C   READ AND PRINT SELECTED VALUES
C    IO=5---PRINT ONLY
C  TO USE IO=3 OR 4  THE CASLLING PROGRAM MUST INITIALZE THE VEC.
C   xxxxxxTHIS ROUTINE SETS IO=0---- WHEN NO DATA IN INPUT FILE
C
CREAD IS FROM UNIT SPECIFIED BY CALLING PROGRAM IN KIN,WRITE IS TO
C  KOUT.NDIM IS THE DECLARED DIMENSION OF A IN THE CALLING
C    PROGRAM
      CHARACTER MSG*60
      DIMENSION A(NDIM)
      COMMON /MAUNS/ MSG
      IF ((IO.EQ.1).OR.(IO.EQ.2)) THEN
C  READ ENTIRE VECTOR
      WRITE(KOUT,*)'ENTER ',NUMEL,'ELEMENTS>'
      READ(KIN,*,END=29)(A(I),I=1,NUMEL)
      END IF
      IF (IO.EQ.1) THEN
      RETURN
      END IF
      IF ((IO.EQ.3).OR.(IO.EQ.4)) THEN
C   READ ONLY SELECTED ELEMENTS.  THE FIRST NUMBER ON EACH CARD
C IS THE SUBSCRIPT, THE SECOND IS THE DATA ENTRY
C*******NOTE ONLY ONE DATA ENTRY PER CARD
C******** FIRST CARD MUST CONTAIN THE TOTAL NUMBERR OF ENTRIES TO BE
C    READ
      WRITE(KOUT,*)'ENTER THE NUMBER OF ENTRIES TO BE MADE>'
      READ(KIN,*,END=29)NUMENT
      DO 20 IT=1,NUMENT
      WRITE(KOUT,*)'ENTER THE ELEMENT NUMBER ,THEN ITS VALUE>'
```

```
      READ(KIN,*,END=29)I,ENTRY
      A(I)=ENTRY
      IF (IO.EQ.4)THEN
      WRITE(KOUT,33)' '
      WRITE(KOUT,*)MSG
      WRITE(KOUT,*)'ELEMENT NUMBER , ENTRY)'
      WRITE(KOUT,11)I,A(I)
      END IF
   20 CONTINUE
      RETURN
      END IF
      IF ((IO.EQ.2).OR.(IO.EQ.5)) THEN
C    TO GET HERE IO=2 OR 5 SO PRINT OUT ENTIRE VECTOR
      WRITE(KOUT,33)' '
      WRITE(KOUT,*)MSG
      WRITE(KOUT,*)'  THE VECTOR  HAS ',NUMEL, '  ELEMENTS'
      WRITE(KOUT,22)(A(I),I=1,NUMEL)
      RETURN
      END IF
      RETURN
   29 PRINT*,'END OF DAT REACHED DURING INPUT IN MVECIO'
      IO=0
C    *******THIS ROUTINE SETS IO=0---- WHEN NO DATA IN INPUT FILE
33    FORMAT(A10,/)
   11 FORMAT(I4,15X,E12.6)
   22 FORMAT(10(10(1X,E12.6),1,/))
      RETURN
      END


*DECK MMATIO
      SUBROUTINE MMATIO(A,IR,IC,IO,KIN,KOUT,NDIM,NDIM1)
      CHARACTER MSG*60
      DIMENSION A(NDIM,NDIM1)
      COMMON /MAUNS/ MSG
C THIS SUBROUTINE READS AND/OR PRINTS THE MATRIX A DEPENDING ON THE
C VALUE OF IO.  IT READS FROM UNIT SPECIFIED BY KIN AND WRITES TO UNIT
C  KOUT.  IO=1--READ ENIRE ARRAY     IO=2--READ AND PRINT ENTIRE
C
C   ARRAY.    IO=3---READ SELECTED ELEMENTS OF A  IO=4---READ AND
C  PRINT SELECTED ELEMENTS OF A      IO=5  ---PRINT ENTIRE ARRAY
C   NDIM,NDIM1 ARE THE DIMENSIONS OF A IN THE CALLING PROGRAM
C
C  *******NOTE IF IO=3 OR 4  THE CALLING PROGRAM MUST INITIALIZE
C                  THE ENTIRE ARRAY BEFORE CALL
C  ********THIS ROUTINE SETS IO =0 ---- WHEN THE INPUT FILE IS EMPTY
C
      IF ((IO.EQ.1).OR.(IO.EQ.2)) THEN
C    READ ENTIRE ARRAY IN FREE FORMAT,ROW MAJOR ORDER
      WRITE(KOUT,*)'ENTER ',(IR*IC),' ARRAY ELEMENTS IN ROW MAJ ORDER)'
      READ(KIN,*,END=29)((A(I,J),J=1,IC),I=1,IR)
      END IF
      IF (IO.EQ.1) THEN
      RETURN
      END IF
      IF ((IO.EQ.3).OR.(IO.EQ.4)) THEN
C    READ IN SELECTED ELEMENTS OF A
C    THE FIRST CARD IN THE INPUT STREAM MUST CONTAIN THE TOTAL
C     NUMBER OF ELEMENTS TO BE READ IN.  ONLY ONE ENTRY PER CARD,
C     THE FIRST ITEM ON EACH CARD IS THE ROW,  THESECOND IS THE COL THE
C     LAST ON EACH CARD IS THE DATA FOR THAT LOCATION
C              FREE FORMAT IS USED
      WRITE(KOUT,*)'ENTER THE NUMBER OF ENTRIES TO BE MADE)'
      READ(KIN,*,END=29)NUMEL
      DO 20 I=1,NUMEL
      WRITE(KOUT,*)'ENTER THE ROW,AND COLUMN FOLLOWED BY ITS VALUE)'
      READ(KIN,*,END=29)II,J,ENTRY
      A(II,J)=ENTRY
      IF (IO.EQ.4) THEN
      WRITE(KOUT,33)' '
      WRITE(KOUT,*)MSG
      WRITE(KOUT,*)'(',II,',',J,')=',A(II,J)
      END IF
   20 CONTINUE
      RETURN
      END IF
      IF ((IO.EQ.2).OR.(IO.EQ.5)) THEN
C    IO = 2 OR 5 IF HERE SO PRINT ENTIRE ARRAY
```

```
      WRITE(KOUT,33)' '
      WRITE(KOUT,*)MSG
      WRITE(KOUT,*)'  MATRIX  SIZE IS ',IR,'  X ',IC
      DO 49 I=1,IR
   49 WRITE(KOUT,48)(A(I,J),J=1,IC)
33    FORMAT(A10,/)
   48 FORMAT(5(10(1X,E12.6),:,/))
      END IF
      RETURN
   29 PRINT *,'END OF DATA REACHED DURING INPUT'
C  *******THIS ROUTINE SETS IO =0 ---- WHEN THE INPUT FILE IS EMPTY
      IO=0
      RETURN
      END
*DECK CLQGRS
      SUBROUTINE CLQGRS(GCSTR,FM,BM,RKFSS,HM,GCX,GCY,GCZ,
     1BCY,BCZ,FC,YD,RM,QM,FT,BT,GT,QT,RT,HT,IRY,IFLGCZ,WM1
     1 ,WM2,WM3,PO,GM,WM4,WM5,WM6,
     1  WV1,WV2,WUU,WXX,XO,WXU,WM7,WM8)
C  THIS ROUTINE PERFORMS SET UP FOR USING THE CONTINUOUS TIME
C PERFORMANCE ANALYSIS  FOR ANLOG REGULATOR
      DIMENSIONGCSTR(NDIM,NDIM),FM(NDIM,NDIM),BM(NDIM,NDIM),PO(NDIM,
     1NDIM),HM(NDIM,NDIM),GCX(NDIM,NDIM),GCY(NDIM,NDIM),GCZ(NDIM,NDIM),
     1BCY(NDIM,NDIM),BCZ(NDIM,NDIM),FC(NDIM,NDIM),YD(NDIM3),WM1(NDIM,
     1NDIM),WM2(NDIM,NDIM),WM3(NDIM,NDIM),FT(NDIM,NDIM),
     1 BT(NDIM,NDIM),GT(NDIM,NDIM),HT(NDIM,NDIM),RM(NDIM,NDIM),
     1QT(NDIM,NDIM),RT(NDIM,NDIM),QM(NDIM,NDIM),WV1(NDIM),WV2(NDIM
     11),XO(NDIM),GM(NDIM,NDIM),WUU(NDIM,NDIM),
     1 WXX(NDIM,NDIM),WM4(NDIM,NDIM),WM5(NDIM,NDIM),
     1 WM6(NDIM,NDIM),WXU(NDIM,NDIM),WM7(NDIM,NDIM),WM8(NDIM,NDIM)
      DIMENSION COM1(1),COM2(1)
      CHARACTER MSG*60
      INTEGER IFLGCZ
      REAL RKFSS(NDIM,NDIM)
       COMMON /MAIN4/NDIM2,NDIM3
      COMMON /MAUNS/ MSG
      COMMON /MAIN1/ NDIM,NDIM1,COM1
      COMMON /MAIN2/  COM2
      COMMON /RNTIM/RNTIME,DELTIM
      COMMON /INOU/ KIN,KOUT,KPUNCH
      COMMON /MAIN6/ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1  IRHT,IR04,IO,LQG,IRHM,NUMDTS
      WRITE(KOUT,*)'DO YOU WANT TO CALCULATE EIGENVALUES OF THE TRUTH MO
     1DEL AND CONTROLLER MODEL  F  MATRICES,  Y  OR  N>'
      READ(KIN,23,END=2933) MSG
23    FORMAT(A1)
      IF (MSG.EQ.'Y') THEN
      WRITE(KOUT,*)' '
      WRITE(KOUT,*)'THE EIGENVALUES OF THE TRUTH MODEL F MATRIX'
       CALL MEIGN(FT,WV1,WV2,IRFT,WM1)
      WRITE(KOUT,*)'THE EIGENVALUES OF THE CONT. MODEL F MATRIX'
      CALL MEIGN(FM,WV1,WV2,IRFM,WM1)
      END IF
      CALL CKFTR(FM,GM,RM,HM,NUMDTS,RKFSS,QM,WM1,WM2,WM3,WM4,WM5,WM6,
     1  IRFM,IRHM,ICGM,GCY,GCZ,BM,ICBM)
C  GCZ,GCY,  IN CALL TO CKFTR ARE USED AS DUMMY ARRAYS FOR M2 ,F2
      CALL CDTCON(FM,BM,WXX,WUU,GCSTR,IRFM,ICBM,WM1,WM2,WM3,WM4,WM5,WM6,
     1 WXU,WM7,WM8)
      WRITE(KOUT,*)'ENTER THE TOTAL RUN TIME AND THE TIME INCREMENT)'
      READ(KIN,*,END=2933)RNTIME,DELTIM
      C1=-1
      CALL IDNT(IRFM,WM1,C1)
C WM1=-I  IRFM X IRFM
      CALL MAT1(GCSTR,WM1,ICBM,IRFM,IRFM,GCX)
      IO=5
      MSG='GCX FOLLOWS,  GCY,GCZ SET=0'
      CALL MMATIO(GCX,ICBM,IRFM,IO,KIN,KOUT,NDIM,NDIM)
C  NOTE******* OTHER GAIN MATRICES,GCZ,AND GCY SHOULD BE CALCULATED IN
C                 THIS MODULE FOR USE IN THE PERFORMANCE ANALYSIS ROUTINE.
C
      DO 2902 III=1,ICBM
      DO 2902 IIII=1,IRHM
2902  GCZ(III,IIII)=0
      IFLGCZ=1
C   IFLGCZ=1 INDICATES GCZ IS SET TO ZERO--PREF ANALYSIS ROUTINE USES IF
C
      DO 2903 I=1,1000
2903  YD(I)=0
```

```
      IRY=1
      DO 1723 I=1,ICBM
      DO 1723 J=1,IRY
1723  GCY(I,J)=0
C  FORM BCY
      DO 1702 I=1,IRFM
      DO 1702 J=1,IRY
1702  BCY(I,J)=0
C  YD IS ALLOWED TO ONLY BE A SCALAR AT THIS TIME
C   FORMBCZ
      C1=1.0
      CALL EQUATE(BCZ,RKFSS,IRFM,IRHM)
      MSG='BCZ FOLLOWS,BCY=0'
      CALL MMATIO(BCZ,IRFM,IRHM,IO,KIN,KOUT,NDIM,NDIM)
C   FORM FC
      CALL MAT1(BM,GCX,IRFM,ICBM,IRFM,WM1)
      CALL MADD1(IRFM,IRFM,FM,WM1,WM2,C1)
      WRITE(KOUT,1)' '
      WRITE(KOUT,1)'DO YOU WANT TO CALCULATE THE EIGENVALUES OF THE CONT
     1INUOUS-TIME LQ CONTROLLER?    Y   OR N)'
      READ(KIN,23)MSG
      IF (MSG.EQ.'Y') THEN
        WRITE (KOUT,1)'THE EIGENVALUES THAT CORRESPOND TO THE POLES  OF
     1 THE CONTINUOUS-TIME LQ CONTROLLER  ARE...'
        CALL  MEIGN(WM2,WV1,WV2,IRFM,WM1)
        END  IF
      C1=-1
      CALL MAT1(RKFSS,HM,IRFM,IRHM,IRFM,WM1)
        WRITE(KOUT,1)'
        WRITE(KOUT,1)'DO YOU WANT TO  CALCULATE THE POLES OF THE CONTINU
     1OUS-TIME KALMAN FILTER?   Y  OR  N)'
        READ(KIN,23)MSG
       IF(MSG.EQ.'Y') THEN
        CALL  MADD1(IRFM,IRFM,FM,WM1,WM7,C1)
        WRITE(KOUT,1)'THE EIGENVALUES  THAT ARE THE  POLES OF THE CNOTINU
     1OUS-TIME KALMAN FILTER  ARE....'
        CALL   MEIGN(WM7,WV1,WV2,IRFM,WM8)
          END IF
      CALL MADD1(IRFM,IRFM,WM2,WM1,FC,C1)
      MSG=' FC FOR THE LQG CONTORLLER IS'
      CALL MMATIO(FC,IRFM,IRFM,IO,KIN,KOUT,NDIM,NDIM)
      WRITE(KOUT,1)' '
      WRITE(KOUT,1)'DO YOU WISH TO CALCULATE THE EIGENVALUES OF THE LQG -
     1CONTROLLER   F  MATRIX?   Y  OR N)'
      READ(KIN,23)MSG
      IF (MSG.EQ.'Y') THEN
      WRITE(KOUT,1)' '
      WRITE(KOUT,1)'THE EIGENVALUES OF THE LQG CONTROLLER F MATRIX ARE'
      CALL MEIGN(FC,WV1,WV2,IRFM,WM1)
      END IF
      RETURN
2933  IO=0
      END
$DECK DAS1
      SUBROUTINE DAS1(GQGT,BM,V,ICBM,WM3,IRFM)
      DIMENSION GQGT(NDIM,NDIM),BM(NDIM,NDIM),V(NDIM,NDIM),
     1  WM3(NDIM,NDIM),COM1(1),COM2(1)
C    THIS SUBROUTINE MODIFIES  GQGT AND RETURNS THE MODIFIED
C     VALUE IN GQGT, WHERE GQGT IS USED IN THE KALMAN FILTER
C     GAIN CALCULATIONS.   THE MODS ARE IN ACCORDANCE WITH THE
C     THE TECHNIQUE DEVELOPED BY DOYLE AND STEIN IN 'ROBUSTNESS
C      WITH OBSERVERS',IEEE TRANS. ON AUTO. CONTROL,VOIL AC24,
C     NO. 4,AUG, 79,PGS 607-611.
C
C                 THE VALUE RETURNED IN GQGT IS QQ , WHERE  QQ IS
C    QQ=GQGT+SQ(SQ)BM(V(BMT)
C
C    SQ IS A SCALAR DESIGN PARAMETER, THAT AS IT APPROACHES
C    INFINITY,  CAUSES THE LQG CONTROLLER TO RECOVER THE ROBUSTNESS
C     PROPERTIES OF A FULL STATE FEEDBACK CONTROLLER.
CTHE MATROX--V-- IS ALSO A DESIGN, PARAMETER WITH THE REQUIREMENT
C   THAT IT BE POSITIVE DEFINITE.    BM---- IS THE CONTROLLER MODEL INPUT
C    MATRIX.    GQGT --- IS THE CONTROLLER MODEL INPUT NOISE STRENGTH
C   MATRIX QM, PREMULTIPIED BY GM AND POST MUTIPLIED BY GMT WHERE GM IS
C    THE INPUT NOIESE MATRIX.
      CHARACTER MSG*60
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON /MAIN2/ COM2
      COMMON /INOU/ KIN,KOUT,KPUNCH
      COMMON /MAUN5/ MSG
      WRITE(KOUT,11)' '
      WRITE(KOUT,1)'THIS ROUTINE MODIFIES THE VALUE OF GM(QM)GMT
     1 USED IN CALCULATING THE KALMAN FILTER GAIN, RKFSS.'
      WRITE(KOUT,1)'THE MODIFIED Q IS = GM(QM)GMT+SQ*SQ(BM)V(BMT)WHERE
```

```
     1 SQ IS A SCALAR DESIGN PARAMETER AND U IS A POSITIVE DEFINITE
     1MATRIX DESIGN PARAMETER. THE LARGER SQ, THE MORE ROBUST THE CONTRO
     1L SYSTEM WILL BE.'
      DO 5 INP=1,1000
      WRITE(KOUT,11)' '
11    FORMAT(A1Q,/)
      WRITE(KOUT,*)'ENTER 1-TO INPUT SQ, 2-TO INPUT U 3- TOCALCULATE MOD
     1IFIED QQ,  4- TO EXIT THIS ROUTINE)'
      READ(KIN,*)ISEL
      GO TO (1,2,3,4)ISEL
1     WRITE(KOUT,11)' '
      WRITE(KOUT,*)'ENTER SQ)E-->'
      READ(KIN,*)SQ
      GO TO 5
2     WRITE(KOUT,11)' '
       WRITE (KOUT,*)'U IS INIALIZED TO .ZERO UPON ENTRY INTO THIS OPTIO
     1N'
      DO 7 I=1,NDIM
      DO 7 J=1,NDIM
 7      U(I,J)=0
      WRITE(KOUT,*)'ENTER I/O OPTION FOR POSITIVE DEF U(SEE INPUT ROUTIN
     1E))'
      READ(KIN,*)IO
      MSG='DESIGN PARAMETER   U MATRIX ENTRIES'
      CALL MMATIO(U,ICBM,ICBM,IO,KIN,KOUT,NDIM,NDIM)
3     CALL MAT3(IRFM,IRFM,BM,U,WM3)
C  WM3=BM(U)BMT   IRFM  X  IRFM
      SQ1=SQ*SQ
      CALL MADD1(IRFM,IRFM,GQGT,WM3,U,SQ1)
C  U IRFM  X  IRFM NOW CONTAINS THE MOD. VALUE  ---QQ
      MSG='THE DOYLE AND STEIN MODIFED QQ MATRIX IS'
      IO=5
      CALL MMATIO(U,IRFM,IRFM,IO,KIN,KOUT,NDIM,NDIM)
5     CONTINUE
4     CONTINUE
C   QQ IS ACCEPTABLE SO PUT INTO GQGT
      DO 20 I=1,IRFM
      DO 20 J=1,IRFM
20    GQGT(I,J)=U(I,J)
      RETURN
      END
*DECK MYINTG
      SUBROUTINE MYINTG(PHI,INTGA,INTBA,WME,GA,QA,FA,BA,IRFA,ICGA,IRY,
     1IRGA)
C     THIS SUBROUTINE SETS UP THE NECESSARY INTEGRALS FOR USE BY
C                THE PERFORMANCE ANAL. ROUTINE.  THE STATE TRANSITION MATRIX,
C     EXP(FA*TIME)=EAT,  INTEG(EAT(GA, QA (GAT) EATT),  AND
C     INTEG(EAT (BA)).   WME IS A DUMMY WORK SPACE
      REAL PHI(NDIM2,NDIM2),INTGA(NDIM2,NDIM2),INTBA(NDIM2,NDIM2),
     1 WME(NDIM2,NDIM2),QA(NDIM2,NDIM2),BA(NDIM2,NDIM2),
     1GA(NDIM2,NDIM2),FA(NDIM2,NDIM2)
      DIMENSION COM1(1),COM2(1)
      COMMON/MAIN1/NDIM,NDIM1,COM1
      COMMON/MAIN2/ COM2
      COMMON /INOU/ KIN,KOUT,KPUNCH
      COMMON /RNTIM/RNTIME,DELTIM
      COMMON /MAIN4/NDIM2,NDIM3
C***FORM QA(QA)GAT --NEED FOR KLIENMAN ROUTINE
C
      NSAV1=NDIM
      NSAV2=NDIM1
      NDIM=NDIM2
      NDIM1=NDIM2+1
      CALL DSCRT(IRFA,FA,DELTIM,INTGA,WME,10)
C WME=INT(EAT)
      CALL MAT1(WME,BA,IRFA,IRFA,IRY,INTBA)
C INTBA= INT(EAT)BA   IRFA  XIRY   NEEDED IN MXA UPDATE
      CALL MAT3(IRFA,IRGA,GA,QA,PHI)
C PHI=GA(QA)(GAT)   IRFA  X  IRFA
      CALL INTEG(IRFA,FA,PHI,INTGA,DELTIM)
C PHI=EXP(FA)   IRFA  X  IRFA
C INTGA=INTEGRAL  ( EXP(FA)(GA)(QA)(GATT)(EXP(FA)T))   IRFA  X  IRFA
C
      NDIM=NSAV1
      NDIM1=NSAV2
      RETURN
      END
```

```
*DECK DSCRTZ
      SUBROUTINE DSCRTZ(UM1,PHIM,BCYD,BCZD,IRFM,DELTIM,FC,BCY,
     1 IRY,BCZ,IRHM,PHIT,QTD,BTD,GT,QT,FT,BT,IRFT,ICGT,ICBT,IRHT,
     IRTD,RT,GCX,ICBM)
C    THIS. ROUTINE DISCRTIZES A CONTINUOUS TIME LQG CONTROLLER USING
C  FIRST ORDER APRROXIMATIONS TO THE REQUIRED INTEGRALS
C  AND PROVIDES AN EQUIVALENT DISCRETE TIME REPRESENTATION OF THE TRUTH
C MODEL FOR USE IN THE PERFAL ROUTINE
      REALUM1(NDIM,NDIM),PHIM (NDIM,NDIM),BCYD (NDIM,NDIM),
     1 BCZD(NDIM,NDIM),FC(NDIM,NDIM), BCY(NDIM,NDIM),
     1 BCZ(NDIM,NDIM),PHIT(NDIM,NDIM),QTD(NDIM,NDIM),
     1 BTD(NDIM,NDIM),GT(NDIM,NDIM),QT(NDIM,NDIM),
     1 FT(NDIM,NDIM), BT(NDIM,NDIM),RTD (NDIM,NDIM),
     1 GCX(NDIM,NDIM),RT(NDIM,NDIM)
      REAL COM1(1),COM2(1)
      CHARACTER MSG*1
      COMMON/MAIN1/NDIM,NDIM1,COM1
      COMMON/MAIN2/ COM2
      COMMON/MAIN4/NDIM2,NDIM3
      COMMON /INOU/KIN,KOUT,KPUNCH
      C1=1.0
      CALL IDNT(IRFM,UM1,C1)
      CALL IDNT(IRFM,UM1,C1)
      CALL MADD1(IRFM,IRFM,UM1,FC,PHIM,DELTIM)
C PHIM=I+FC(DELTIM) IST ORDER APPROX TO STATE TRANS MATRIX OF CONT
C CALCULATE GCX =-GCSTR
      C1=-1.0
C  RECALL THAT GCSTR WAS PASSED INTO THIS ROUTINE IN BTD
      CALL SCALE(GCX,BTD,ICBM,IRFM,C1)
      C1=1.0
      CALL SCALE(BCYD,BCY,IRFM,IRY,DELTIM)
C  BCYD DISCRETE TIME APPROX OF BCY
      CALL SCALE(BCZD,BCZ,IRFM,IRHM,DELTIM)
C  BCZD DISCRETE TIME APPROX OF BCZ
      NSAV3=NDIM2
      NDIM2=NDIM
      CALL MYINTG(PHIT,QTD,BTD,UM1,GT,QT,FT,BT,IRFT,ICGT,ICBT,IRHT)
      NDIM2=NSAV3
C PHIT,QTD,BTD ARE EQUIV. DISCRETE TIME REPRESTATIONS OF TRUTH MODL
C MATRICE
      WRITE(KOUT,1)' '
      WRITE(KOUT,1)' WAS THE VALUE ENTERED IN RT DURING INPUT A CONTINUO
     1US TIME  OR A DISCRETE TIME VALUE?  ENTER A C FOR  CONTINUOUS , A
     1 D FOR DISCRETE VALUE)'
      READ(KIN,12)MSG
12    FORMAT(A1)
      IF (MSG.EQ.'C') THEN
      C1=1/DELTIM
      CALL SCALE(RTD,RT,IRHT,IRHT,C1)
      ELSE
      CALL EQUATE(RTD,RT,IRHT,IRHT)
      END IF
C RTD IS THE DISCRETE TIME APPROX OF RT
      C1=1.0
      CALL IDNT(IRFT,UM1,C1)
C  UM1=GTD  = I
      END
*DECK DLOGRS
      SUBROUTINE DLOGRS(GCX,GCY,GCZ,BCY,BCZ,PHIT,PHIC,RTD,GTD,QTD,
     1 BTD,FM,BM,QM,GM,RM,HM,GT,QT,FT,BT,RT,HT,UXX,UUU,GCSTR,RKFSS,YD,
     1 IRY,IFLGCZ,UXU,UM1,UM2,UM3,UM4,UM5,UU1,UU2)
CTHIS  SUBROUTINE FORMATS THE SAMPLED DATA CONTROLLER INTO THE FORMAT
C  REQUIRED BY THE PERFORMANCE ANALYSIS ROUTINE
C THE FORMAT IS SPECIFIED IN THE COMMENT STATEMENTS IN THE CODE, AND
C IN MORE DETAIL IN E. LLOYD S MASTERS THESIS,DB1,AFIT.
      REAL COM1(1),COM2(1)
      REAL GCX(NDIM,NDIM),GCY(NDIM,NDIM),GCZ(NDIM,NDIM),
     1BCY(NDIM,NDIM), BCZ(NDIM,NDIM),PHIT(NDIM,NDIM),
     1 PHIC(NDIM,NDIM),RTD(NDIM,NDIM),GTD(NDIM,NDIM),
     1QTD(NDIM,NDIM),BTD(NDIM,NDIM),FM(NDIM,NDIM),
     1 BM(NDIM,NDIM), QM(NDIM,NDIM),GM(NDIM,NDIM),
     1 UU1(NDIM),UU2(NDIM),
     1 RM(NDIM,NDIM),HM(NDIM,NDIM),GT(NDIM,NDIM),
     1 QT(NDIM,NDIM), FT(NDIM,NDIM), BT(NDIM,NDIM),
     1 RT(NDIM,NDIM),HT(NDIM,NDIM),UXX(NDIM,NDIM),
     1 UUU(NDIM,NDIM),GCSTR(NDIM,NDIM), RKFSS(NDIM,NDIM),
     1 UXU(NDIM,NDIM),UM1(NDIM,NDIM),UM2(NDIM,NDIM),
     1 UM3(NDIM,NDIM),UM4(NDIM,NDIM),UM5(NDIM,NDIM)
      REAL YD(NDIM3)
      INTEGER IFLGCZ
```

```
      CHARACTERMSG*60,MSG1*1,MSG2*1
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON /MAIN2/COM2
      COMMON/INOU/KIN,KOUT,KPUNCH
      COMMON /MAIN4/NDIM2,NDIM3
      COMMON/MAUNS/MSG
      COMMON /RNTIM/RNTIME,DELTIM
      COMMON/MAING/ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1 IRHT,IRQA,IO,LOG,IRHM,NUMDTS
      WRITE(KOUT,*)'ENTER THE TOTAL RUN TIME AND THE SAMPLE TIME>'
      READ(KIN,*)RNTIME,DELTIM
11    FORMAT(A1)
C CALCULATE EQUIVALENT DISCRETE TIME VERSIONS OF, BM--BMD,GM--GMD=I,
C  QM--QMD,   AND PHIM THE STATE TRANSITION MATRIX FOR FM
C**********************************************************************
C  SINCE WORKSPACE IS AT A PREMIUM,  THE TRUTH MAODEL MATRICES
CPHIT,BTD,GTD,QTD,RTD WILL BE USED FOR THEIR  CONTROLLER MODEL
C COUNTER PARTS DURING THIS ROUTINE BEFORE THE EQUIVALENT DISCRETE
C TIME TRUTH MODEL IS COMPUTED.  AT THAT TIME THERE IS NO LONGEG ANY
C NEED FOR THOSE CONTROLLER MODEL MATRICE SINCE THE CONTROLLER IS PUT
C INTO THE PERFOMANNCE ANALYSIS FORMAT,PHIC BCY,.....GCZ
C**********************************************************************
      NSAV3=NDIM2
      NDIM2=NDIM
      CALL MYINTG(PHIT,GTD,BTD,RTD,GM,QM,FM,BM,IRFM,ICGM,ICBM,ICGM)
C RTD IS USED AS DUMMY WORK SPACE IN CALL TO MYINTG
      NDIM2=NSAV3
      CALL IDNT(IRFM,GTD,1.000)
      WRITE(KOUT,*)'ENTER A C IF THE  VALUED ENTERED INTO RM IS A CONTIN
     1UOUS TIME VALUE TO FORM THE BASIS OF AN APPROXIMATE DISCRETE TIME
     1 RM,ENTER A  D  OTHERWISE)'
      READ(KIN,11)MSG1
      IF  (MSG1.EQ.'C') THEN
C   APPROXIMATE RMD=RM/SAMPLE TIME
      C1=1/DELTIM
      CALL SCALE(RTD,RM,IRHM,IRHM,C1)
      ELSE
C  THE VALUE IN RM IS DISCRETE TIME ALREADY
      CALL EQUATE(RTD,RM,IRHM,IRHM)
      END IF
C SET UP X , S , AND U FOR DDTCON
      CALL XSU(GCZ,GCY,PHIC,GCX,BCY,BCZ,RKFSS,UM1,UM2,UM3,UM4,UM5,
     1 FM,BM,IRFM,ICBM,UXX,UUU,UXU,PHIT)
C GCZ NOW CONTAINS X, PHIC CONTAINS U,GCY CONTAINS  S
C GCX,BCZ,BCY,RKFSS WERE DUMMY WORK AREAS IN XSU
      CALL DDTCON(PHIT,UXX,GCX,GCZ,BTD,PHIC,RKFSS,BCY,BCZ,GCY,UUU,
     1 GCSTR,UXU,UM1,UM2,UM3,UM4,UM5)
CBCY,BCZ,GCY,PHIC,GCZ,GCX,RKFSS  ARE USED AS DUMMY WORK SPACE IN DDTCON
      WRITE(KOUT,*)'DO YOU WISH TO COMPUTE THE KALMAN FILTER GAIN OR PIC
     1K IT DIRECTLY (AS IN MAYBECK SECTION, 14.5)ENTER A C TO COMPUTE  E
     1NTER A P TO PICK IT DIRECTLY>'
      READ (KIN,11)MSG2
      IF (MSG2.EQ.'C')THEN
      CALL DKFTR(PHIT,BTD,GTD,QTD,GCX,RTD,HM,GCY,GCZ,PHIC,RKFSS,BCY,BCZ,
     1FM,GM,QM,BM,UM1,UM2)
CGCX,GCY,GCZ,BCZ,BCY ARE USED AS DUMMY WORK SPACE IN CALL TO DKFTR
      ELSE
      CALL PKDIRC(GCX,PHIT,GCY,GCZ,HM,BTD,RKFSS,ICBM,IRFM,IRHM)
C  GCX,GCY,GCZ ARE, DUMMY WORKSPACES IN CALL TO PKDIRC
      END IF
C**********INTHE FOLLOWING CALCULATIONS,  BCY,GCY,BCZ ARE DUMMY WORKSP
C**********ACES UNTIL THEIR LAST USE WHEN THEY ARE SET EQUAL TO THEIR F
C**********INAL VALUES FOR PERFAL SUBROUTINE
      C1=1.0
      CALL IDNT(IRFM,BCY,C1)
      CALL MAT1(RKFSS,HM,IRFM,IRHM,IRFM,GCY)
      C1=-1.0
      CALL MADD1(IRFM,IRFM,BCY,GCY,BCZ,C1)
C BCZ = I-RKFSS(HM)
      WRITE(KOUT,*)' '
      WRITE(KOUT,*)'DO YOU WISH TO CALCULATE  THE SAMPLED-DATA FILTER P
     1OLES?    Y    OR    N>'
      READ(KIN,12)MSG
      IF (MSG.EQ.'Y') THEN
      CALL MAT1(PHIT,BCZ,IRFM,IRFM,IRFM,PHIC)
      WRITE(KOUT,*)'THE EIGENVALUES THAT CORRESPOND TO THE SAMPLED-DATA
     1 FILTER POLES ARE.....'
```

```
        CALL MEIGN(PHIC,WV1,WV2,IRFM,WM1)
          END IF
        CALL SCALE(BCY,GCSTR,ICBM,IRFM,C1)
C BCY=-GCSTR   ICBM   X   IRFM
        IF (MSG2.EQ.'C') THEN
C  FORMULATE THE OPTIMAL CONTROL LAW FOR PERFAL
        CALL MAT1(BCY,BCZ,ICBM,IRFM,IRFM,GCX)
C  GCX= -GCSTR(I-RKFSS(HM))   ICBM   X   IFRM
        CALL MAT1(BCY,RKFSS,ICBM,IRFM,IRHM,GCZ)
C  GCZ=  -GCSTR(RKFSS)  ICBM  X  IRHM
        IFLGCZ=0
        CALL  MAT1(BTD,BCY,IRFM,ICBM,IRFM,GCSTR)
        C1=1.0
        CALL MADD1(IRFM,IRFM,PHIT,GCSTR,BCY,C1)
C  BCY=  PHIT-BTD(GCTR)   IRFM  X  IRFM
         WRITE(KOUT,1)' '
         WRITE(KOUT,1)'DO YOU WISH TO CALCULATE  THE POLES OF  THE OPTIMAL
       1 LQ SAMPLED-DATA CONTROLLER?     Y  OR  N)'
         READ(KIN,12)MSG
         IF (MSG.EQ.'Y')  THEN
         WRITE(KOUT,1)'THE EIGENVALUES THAT CORRESPOND TO THE  POLES OF TH
      1E OPTIMAL  LQ CONTROLLER ARE.....'
         CALL MEIGN(BCY,WV1,WV2,IRFM,PHIC)
         END IF
        CALL MAT1(BCY,BCZ,IRFM,IRFM,IRFM,PHIC)
C  PHIC =(PHIT-BTD(GCSTR))(I-RKFSS(HM))  IRFM  X  IRFM
        CALL MAT1(BCY,RKFSS,IRFM,IRFM,IRHM,BCZ)
C  BCZ=(PHIT-BTD(GCSTR))RKFSS  IRFM  X  IRHM
        ELSE
C  FORMULATE THE SUBOPTIMAL CONTROL LAW USTR=-GCSTR(X AT  TSUB I MINUS)
        CALL EQUATE(GCX,BCY,ICBM,IRFM)
C  GCX=-GCSTR  ICBM  X  IRFM
        CALL MAT1(PHIT,BCZ,IRFM,IRFM,IRFM,GCSTR)
        C1=1.0
        CALL MAT1(BTD,GCX,IRFM,ICBM,IRFM,BCY)
         WRITE(KOUT,1)' '
         WRITE(KOUT,1)'DO YOU  WISH TO  CALCULATE THE  POLES OF THE OPTIMA
       1L  LQ  SAMPLED DATA CONTROLLER?   Y   OR  N)'
        READ(KIN,12)MSG
        IF (MSG.EQ.'Y') THEN
         CALL MADD1(IRFM,IRFM,PHIT,BCY,PHIC,C1)
        WRITE(KOUT,1)'THE EIGENVALUES THAT CORRESPOND TO THE POLES OF THE
       1OPTIMAL  LQ CONTROLLER ARE.....'
         CALL MEIGN(PHIC,WV1,WV2,IRFM,WM1)
        END IF
        CALL MADD1(IRFM,IRFM,GCSTR,BCY,PHIC,C1)
C  PHIC= PHIT(I-RKFSS(HM))-BTD,GCSTR  IRFM  X  IRFM
        CALL MAT1(PHIT,RKFSS,IRFM,IRFM,IRHM,BCZ)
C  BCZ=PHIT(RKFSS)  IRFM   X  IRHM
        DO 101 I=1,IRFM
        DO 101J=1,IRHM
101     GCZ(I,J)=0
        IFLGCZ=1
        END IF
        IRY=1
        DO 102 I=1,NDIM
        DO 102 J=1,IRY
        GCY(I,J)=0
102        BCY(I,J)=0
        DO 107 I=1,1000
107        YD(I)=0
        IO=5
        MSG='PHIC FOR THE SAMPLED DATA CONTROLLER IS'
        CALL MMATIO(PHIC,IRFM,IRFM,IO,KIN,KOUT,NDIM,NDIM)
        WRITE(KOUT,1)' '
        WRITE(KOUT,1)'DO YOU WANT TO CALCULATE THE EIGENVALUES OF THE LQG
       1 CONTROLLER STATE TRANSITION MATRIX, PHIC?  Y  OR  N)'
        READ(KIN,12)MSG
        IF (MSG.EQ.'Y') THEN
        WRITE(KOUT,1)'THE EIGENVALUES OF THE LQG CONTROLLER STATE TRANSITI
       1ON MATRIX ARE.....'
        CALL MEIGN(PHIC,WV1,WV2,IRFM,WM1)
```

```
      END IF
      MSG='BCZ FOLLOWS, BCY=0'
      CALL MMATIO(BCZ,IRFM,IRHM,IO,KIN,KOUT,NDIM,NDIM)
      MSG='GCX FOR THE SAMPLED DATA CONTROLLER IS'
      CALL MMATIO(GCX,ICBM,IRFM,IO,KIN,KOUT,NDIM,NDIM)
      IF (MSG2.EQ.'C') THEN
      MSG='GCZ FOR A COMPUTED KFSS'
      ELSE
      MSG='GCZ FOR KFSS PICKED DIRECTLY'
      END IF
      CALL MMATIO(GCZ,ICBM,IRHM,IO,KIN,KOUT,NDIM,NDIM)
      WRITE(KOUT,*)' GCY IS SET = 0'
C  THIS IS A REGULATOR SO YD IS ALWAYS ZERO
      MSAV3=NDIM2
      NDIM2=NDIM
      CALL MYINTG(PHIT,QTD,BTD,RTD,GT,QT,FT,BT,IRFT,ICGT,ICBT,IRHT)
C  RTD IS USED AS DUMMY WORK SPACE IN CAL TO MYINTG
      NDIM2=MSAV3
C PHIT,QTD,BTD ARE EQUIV. DISCRETE TIME REPRESTATIONS OF TRUTH MODL
C MATRICE
      WRITE(KOUT,*)' '
      WRITE(KOUT,*)' WAS THE VALUE ENTERED IN RT DURING INPUT A CONTINUO
     1US TIME  OR A DISCRETE TIME VALUE?  ENTER A C FOR  CONTINUOUS , A
     1 D  FOR DISCRETE VALUE)'
      READ(KIN,12)MSG
12    FORMAT(A1)
      IF (MSG.EQ.'C') THEN
      C1=1/DELTIM
      CALL SCALE(RTD,RT,IRHT,IRHT,C1)
      ELSE
      CALL EQUATE(RTD,RT,IRHT,IRHT)
      END IF
C RTD IS THE DISCRETE TIME APPROX OF RT
      C1=1.0
      CALL IDNT(IRFT,GTD,C1)
C  GTD=GTD  + I
      RETURN
      END
*DECK DKFTR
      SUBROUTINE DKFTR(PHIM,BMD,GMD,QMD,UM1,RMD,HM,UM3,UM5,UM6,RKFSS,
     1 F2,H2,FM,GM,QM,BM,UM2,UM4)

C               THIS ROUTINE CALCULATE THE STEADY STATE KALMAN FILTER GAIN MATRIX
C               FOR A SAMPLED  ATA CONTROLLER
      REAL PHIM(NDIM,NDIM),BMD(NDIM,NDIM),QMD(NDIM,NDIM),
     1 GMD(NDIM,NDIM),UM1(NDIM,NDIM),RMD(NDIM,NDIM),
     1 HM(NDIM,NDIM), UM3(NDIM,NDIM),   UM5(NDIM,NDIM),
     1 UM6(NDIM,NDIM),  RKFSS(NDIM,NDIM),F2(NDIM,NDIM)
     1 ,H2(NDIM,NDIM),FM(NDIM,NDIM),
     1 GM(NDIM,NDIM),QM(NDIM,NDIM),BM(NDIM,NDIM),
     1 UM2(NDIM,NDIM),UM4(NDIM,NDIM)


      REAL COM1(1),COM2(1)
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON /INOU/ KIN,KOUT,KPUNCH
      COMMON /RNTIM/ RNTIME,DELTIM
      COMMON/MAIN4/NDIM2,NDIM3
      COMMON/MAUN5/MSG
      COMMON /MAIN2/ COM2
       COMMON /MAIN6/ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1 IRHT,IRQA,IO,LQG,IRHM,NUMDTS
       CHARACTER MSG1*1,MSG*60
C  DELETE DETERMINISTIC STATES AS IN THE CONTINUOUS TIME CASE
       WRITE(KOUT,*)'IF YOU PLAN TO USE THE DOYLE AND STEIN TECHNIQUE FOR
     1 THIS RUN YOU MAY WISH TO MODIFY THE VALUE OF NUMDTS, THE NUMBER O
     1F DETERMINISTIC STATES.  DO YOU WANT TO CHANGE NUMDTS?  Y  OR  N)'
      READ(KIN,11)MSG1
      NUMSAV=NUMDTS
      IF (MSG1.EQ.'Y') THEN
      WRITE(KOUT,*)'ENTER THE NEW VALUE OF NUMDTS FOR THIS RUN)'
      READ(KIN,*)NUMDTS
      END IF
      WRITE(KOUT,*)'NUMDTS= ',NUMDTS
      IDS=NUMDTS+1
      IRF2=IRFM-NUMDTS
      IF(NUMDTS.EQ.0) THEN
```

```
C  STORE SYSTEM MODEL IN INTERMEDIATE MATRICES COMPATIBBLE
C  WITH THOSE BELOW WHEN THER ARE DETERMINISTIC STATES REMOVED
      CALL EQUATE(F2,PHIM,IRF2,IRF2)
      CALL EQUATE(WM1,GMD,IRF2,IRF2)
      CALL EQUATE(H2,HM,IRHM,IRF2)
      CALL EQUATE(WM5,BMD,IRF2,ICBM)
      CALL EQUATE(WM2,QMD,IRF2,IRF2)
      ELSE
C DELETE THE DETERMINISTIC STATES FORM THE MODEL USED TO FORM
C  THE STEADY STATE KALMAN FILTER GAIN MATRIX
      DO 2112 I=IDS,IRFM
      II=I-NUMDTS
      DO 2112 J=IDS,IRFM
      JJ=J-NUMDTS
2112  WM4(II,JJ)=FM(I,J)
      DO 2113 I=1,IRHM
      DO 2113 J=IDS,IRFM
      JJ=J-NUMDTS
2113  H2(I,JJ)=HM(I,J)
C FORM B2D G2D NOW
      DO 2115 I=IDS,IRFM
      II=I=NUMDTS-
      DO 2115 J=1,ICBM
2115  WM3(II,J)=BM(I,J)
      DO 2114 I=IDS,IRFM
      II=I-NUMDTS
      DO 2114 J=1,ICGM
2114  WM1(II,J)=GM(I,J)
11      FORMAT(A1)
      NSAV=NDIM2
      NDIM2=NDIM
      CALL MYINTG(F2,WM2,WM5,WM6,WM1,QM,WM4,WM3,IRF2,ICGM,ICBM,ICGM)
      NDIM2=NSAV
      C1=1.0
      CALL IDNT(IRF2,WM1,C1)
      END IF
C CALCULATE QMD (QMD) GMDT
      CALL MAT3(IRF2,IRF2,WM1,WM2,RKFSS)
CRKFSS IS DUMMY WORK SPACE AT THIS POINT IN THE PROGRAM
     -WRITE(KOUT,1)' DO YOU WISH TO MODIFY THE QMD MATRIX BY THE DOYLE
     -1 AND STEIN TECHNIQUE FOR CONTINUOUS TIME CONTROLLERS EXTENDED TO
      1 THE DISCRETE TIME SYSTEMS,  Y  OR  N>'
      READ(KIN,11)MSG1
      IF (MSG1.EQ.'Y') THEN
      CALL DAS2(BM,RKFSS,WM1,ICBM,WM3,IRF2,WM6)
C  RETURNS A MODIFIED QMD VALUE TO BE USED IN FINDING RKFSS
      END IF
C   CALCULATE THE KALMAN FILTER GAINS, RKFSS, FOR EITHER THE MODIFIED
C   QMD OR THE UNMODIFIED QMD
C    QMD IS STORED IN RKFSS
      CALL TRANS2(IRHM,IRF2,H2,WM3)
      CALL KFLTR(IRF2,IRHM,F2,WM3,RKFSS,RMD,WM6,WM1,WM5)
C  WM6=PMSS,WM5 CLOSED LOOP MEAS MATRIX
C WM1 = RKFSS WITHOUT THE ZEROS FOR THE DETERMINISTIC STATES
C   NOW ADD THE ZEROS FOR THOSE STATES
      IF (NUMDTS.EQ.0)THEN
      DO 2029 I=1,IRFM
      DO 2029J=1,IRHM
2029  RKFSS(I,J)=WM1(I,J)
      ELSE
      DO 2119 J=1,IRHM
      DO 2118I=1,NUMDTS
2118  RKFSS(I,J)=0
      DO 2119 I=IDS,IRFM
      II=I-NUMDTS
2119  RKFSS(I,J)=WM1(II,J)
      END IF
C  NOW WRITE OUT THE RKFSS MATRIX
27      IO=5
      MSG='STEADY STATE SAMPLED DATA KALMAN FILTER GAIN MATRIX'
      CALL MMATIO(RKFSS,IRFM,IRHM,IO,KIN,KOUT,NDIM,NDIM)
      NUMDTS=NUMSAV
      RETURN
      END
*DECK DAS2
      SUBROUTINE DAS2(BM,QMD,V,ICBM,WM3,IRFM,WM1)
      REAL BM(NDIM,NDIM),QMD(NDIM,NDIM), V(NDIM,NDIM),
```

```
     1 WM3(NDIM,NDIM),  WM1(NDIM,NDIM)
C  THIS ROUTINE ALLOWS QMD TO  BE MODIFIED IN A MANNER SIMILAR TO THE DO
C   YLE AND STEIN TECHNIQUE FOR CONTINUOUS TIME SYSTEMS.
C  QMDMOD=QMD +(SQ*SQ)(BM(V)BMT)*SAMPLE  TIME
C  WHERE SQ IS A SCALAR DESIGN PARMETER AND V  IS A POSITIVE DEFINITE
C SYMMETRIC MATRIX  DESIGN PARAMETER.    AS  SQ--->TO INFINITY IN THE CO
C NTINUOUS TIME CASE,  ROBUSTNESS PROPERTIES OF A FULL STATE  FEEDBACK
C  CONTROLLER ARE RECOVERED.  THIS ROUTINE IS BASED ON E. LLOYDS MASTERS
C  THESIS ,DBL,  AFIT
      CHARACTER MSG*60
      REAL COM1(1),COM2(1)
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON /RNTIM/RNTIME,DELTIM
      COMMON /MAINS/MSG
      COMMON /MAIN2/ COM2
      COMMON /INOU/KIN,KOUT,KPUNCH
      WRITE(KOUT,11)' '
   11 FORMAT(A1)
      WRITE(KOUT,*)'THIS ROUTINE MODIFIES THE VALUE OF QMD USED TO CALCU
     1LATE THE STEADY STATE KALMAN FILTER GAIN.  THE MODIFICATION PERFOR
     1MED IS SIMILAR TO THE DOYLE AND STEIN TECHNIQUE FOR CONTINUOUS-TIM
     1E SYSTEMS.  FOR A COMPLETE DESCRIPTION SEE E. LLOYDS MASTERS THESI
     1S, AFIT, DBL.  BRIEFLY THE MODIFICATION ATTEMPTS TO ENHANCE ROBUST
     1NESS OF THE LOG CONTROLLER AND IS   QMOD=QMD(QMD)QMDT+SQ*SQ*DELTIM
     1 *(BM(V)BMT))...  THE LARGER THE VALUE CHOSEN FOR THE SCALAR SQ TH
     1E MORE ROBUSTNESS RECOVERY (COMPARED TO FULLSTATE FEED-BACK),  V M
     1UST BE A POSITIVE DEF. MATRIX,  CHOOSING V=I ADDS PSEUDONOISE EQUA
     1LLY TO ALL CONTROL INPUTS.'
      WRITE(KOUT,11)' '
      WRITE(KOUT,*)'ENTER 1-- TO INPUT SQ, 2-- TO INPUT V,  3-- TO
     1 COMPUTE MODIFIED Q, 4-- TO EXIT ROUTINE.....NOTE 1 ,2 MUST BE
     1 ACCOMPLISHED BEFORE 3,  AND 3 BEFORE 4,  BUT THAT 1,2,3, CAN BE
     1 DONE ANY NUMBER OF TIMES BEFORE USING 4'
5      WRITE(KOUT,11)' '
      WRITE(KOUT,*)'ENTER OPTION>'
      READ(KIN,*)IOPT
      GO TO ( 1,2,3,4)IOPT
1     WRITE(KOUT,11)' '
      WRITE(KOUT,*)'ENTER SQ>'
      READ(KIN,*)SQ
      GO TO 5
2     WRITE(KOUT,11)' '
      WRITE(KOUT,*)'V IS INITIALIZED TO THE IDENTITY MATRIX UPON ENTRY T
     1O THIS OPTION.  IF YOU DESIRE TO CHANGE V ,..REMEMBER IT MUST BE
     1 POSITIVE DEFINITE..... ENTER THE I/O OPTION (I/O OPTIONS ARE
     1 PRINTED AT THE BEGINNING OF THE PROGRAM) ELSE ENTER A 0  >'
      C1=1.0
      CALL IDNT(ICBM,V,C1)
      READ(KIN,*)IO
      IF (IO.EQ.0) THEN
      GO TO 5
      ELSE
      MSG='THE CHOSEN V MATRIX IS'
      CALL MMATIO(V,ICBM,ICBM,IO,KIN,KOUT,NDIM,NDIM)
      END IF
      GO TO 5
3     WRITE(KOUT,11)' '
      CALL MAT4(V,BM,ICBM,ICBM,IRFM,WM3)
      CALL MAT1(BM,WM3,IRFM,ICBM,IRFM,WM1)
      C1=SQ*SQ*DELTIM
      CALL MADD1(IRFM,IRFM,QMD,WM1,WM3,C1)
      IO=5
      MSG='MODIFIED Q MATRIX, QMOD='
      CALL MMATIO(WM3,IRFM,IRFM,IO,KIN,KOUT,NDIM,NDIM)
      GO TO 5
4     CALL EQUATE(QMD,WM3,IRFM,IRFM)
C  REPLACE THE VALUE IN QMD WITH QMOD
      RETURN
      END
*DECK DDTCON
      SUBROUTINE DDTCON(PHIM,WXX,WM1,X,BMD,U,WM2,PHIPRM,
     1 XPRIM,S,WUU,GCSTR)
      REAL COM1(1),COM2(1), PHIM(NDIM,NDIM),
     1WXX (NDIM,NDIM),WM1(NDIM,NDIM),X(NDIM,NDIM),
     1 BMD(NDIM,NDIM), U(NDIM,NDIM),WM2(NDIM,NDIM),
     1 PHIPRM(NDIM,NDIM),XPRIM(NDIM,NDIM),S(NDIM,NDIM),
     1 WUU(NDIM,NDIM),GCSTR(NDIM,NDIM)
```

```
      CHARACTER MSG*60
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON /MAIN2/ COM2
      COMMON /MAIN4/NDIM2,NDIM3
      COMMON /INOU/KIN,KOUT,KPUNCH
      COMMON /RNTIM/ RNTIME,DELTIM
      COMMON /MAIN6/ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1 IRHT,IRQA,IO,LQG,IRHM,NUMDTS
      COMMON /MAUN5/MSG
C THIS SUBROUTINE COMPUTES THE STESDY STATE  OPTIMAL FEEDBACK GAIN MATRI
CX  GCSTR,  BASED ON A LINEAR QUADRATIC COST CRITERION, FOR A SAMPLED
C  DATA CONTROLLER
C  SEE MAYBECK CHAP 14 FOR A DETAILED DISCUSSION OF ALGORITHM AND
C EQUATIONS
C**********************************************************************
      MT=1
C
C  TRANSFORM SYSTEM SO KLEINMAN RICCATI SOLVER WILL HANDLE S NOT=0
      CALL PRIMIT(UM2,U,ICBM,GCSTR,S,IRFM,BMD,UM1,PHIPRM,X,XPRIM,
     1 PHIM)
C
C UM2=UI*ST
CnOU COMPUTE KPRIM FROM RICCATI EQUATION
      CALL MAT3(IRFM,ICBM,BMD,GCSTR,UM1)
      CALL DRIC(IRFM,PHIPRM,UM1,XPRIM,X,GCSTR)
C GCSTR CONTAINS INFO THAT IS NOT USED
C X CONTAINS **KPRIM**  IRFM  X  IRFM
C  NOW COMPUTE GCSTRPRIM
      C1=1.0
      CALL MAT3(ICBM,IRFM,BMD,X,GCSTR)
      CALL MADD1(ICBM,ICBM,U,GCSTR,UM1,C1)
      CALL  GMINU(ICBM,ICBM,UM1,GCSTR,MR,MT)
      IF (MR.NE.ICBM) THEN
      PRINT *,'INVERSE OF U IN DDTCON NOT OF FULL RANK,  RANK IS ',
     1 MR,' RANK SHOULD BE =ICBM=',ICBM
      END IF
      CALL MAT4(GCSTR,BMD,ICBM,ICBM,IRFM,UM1)
      CALL MAT1(UM1,X,ICBM,IRFM,IRFM,GCSTR)
      CALL MAT1(GCSTR,PHIPRM,ICBM,IRFM,IRFM,UM1)
C UM1=GCSTRPRIM=((U+BMDT(KPRIM)BMD)I)(BMD(KPRIM)PHIPRM)
C   ICBM   X   IRFM
      CALL MADD1(ICBM,IRFM,UM1,UM2,GCSTR,C1)
C  GCSTR ICBM   X   ICBM
      IO =5
      MSG='THE OPTIMAL STEADY STATE FEEDBACK GAIN MATRIX,GCSTR'
      CALL MMATIO(GCSTR,ICBM,IRFM,IO,KIN,KOUT,NDIM,NDIM)
      RETURN
      END
*DECK PKDIRC
      SUBROUTINE PKDIRC(U,PHIM,UM1,UM2,MM,BMD,RKFSS,ICBM,IRFM,IRHM)
      REAL COM1(1),COM2(1),U(NDIM,NDIM),
     1 PHIM (NDIM,NDIM), UM1(NDIM,NDIM), UM2(NDIM,NDIM),
     1 MM(NDIM,NDIM), BMD(NDIM,NDIM), RKFSS(NDIM,NDIM)
      CHARACTER MSG*60,MSG1*1
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON /MAIN2/COM2
      COMMON /MAUN5/ MSG
      COMMON /INOU/ KIN, KOUT,KPUNCH
C  AS IN MAYBECK, SECTION 14.5, RKFSS=PHIMI(BMD)U *SQ.  SQ IS A
C SCALAR DESIGN PARAMETER AND U IS ANY NONSINGULAR  M X M MATRIX.
C  MAYBECK SUGGESTS THAT U=(MM(PHIMI)BMD)I IS A POSSIBLE CHOICE.
C  THE RKFSS PICKED AS A RESULT OF THIS  ALGORITHM FORMS THE BASIS
C OF A SUBOPTIMAL CONTRL LAW,USTAR=GCSTR(X(TI-MINUS)
      WRITE(KOUT,12) ' '
      WRITE(KOUT,*)'THIS ROUTINE CALCULATES THE STEADY STATE KALMAN FILT
     1ER GAIN DIRECTLY( THAT IS WITH OUT USE PMSS FROM THE MATRIX RICCAT
     1I EQUATION AS THE BASIS OF KFSS)I EQUATION AS IN SECTION 14.5 ,MAY
     1BECK.  KFSS=SQ*(PHIMI)BMD(U)  WHERE THE SCALAR SQ AND THE MATRIX
     1 U  ARE DESIGN PARAMETERS...THE LARGER THE SQ THE MORE ROBUSTNESS
     1  THE SUBSEQUENT CONTROLLER WILL HAVE.  NOTE THERE ARE NO STABILIT
     1Y CLAIMS FOR THE RESULTING CONTROLLER,  SO BE SURE TO CHECK THE EI
     1GENVALUES OF THE SUBSEQUENT CONTROLLER.'
      WRITE(KOUT,12)' '
12     FORMAT(A1/)
      WRITE(KOUT,*)'THE OPTIONS FOR THIS ROUTINE ARE 1->CHOOSE SQ, 2-> C
     1HOOSE U, 3-. COMPUTE AND PRINT RKFSS, 4-> EXIT ROUTINE.....'
5      WRITE(KOUT,12)' '
```

```
      WRITE(KOUT,2)'ENTER OPTION)'
      READ(KIN,2)IOPT
      GO TO (1,2,3,4)IOPT
      GO TO 4
1     WRITE(KOUT,12)' '
      WRITE(KOUT,2)'ENTER A VALUE FOR SQ, LARGER SQ GIVE BETTER ROBUSTNE
     1SS)'
      READ(KIN,2)SQ
      GO TO 5
2     WRITE(KOUT,12)' '
      WRITE(KOUT,2)'DO YOU WANT TO PICK U ARBTRARILY OR DO YOU WANT U TO
     1 BE HM(PHIMI)BMD --INVERSE AS IN MAYBECK SECTION 14.5. NOTE THAT
     1 IRHM MUST BE EQUAL TO ICBM SINCE U MUST BE ICBM  X  ICBM.  ENTER
     1AN  A  FOR ARBITRARY,  U  OTHERWISE)'
      READ(KIN,11)MSG1
11    FORMAT(A1)
      IF(MSG1.EQ.'A')THEN
      WRITE(KOUT,2)'ENTER I/O OPTION FOR U(SEE INPUT ROUTINE FOR EXPLANA
     1TION OF INPUT  OPTIONS 1,2,3,4,5,6))'
      READ(KIN,2)IO
      MSG='ARBITRARY U MATRIX'
      CALL MMATIO(U,ICBM,ICBM,IO,KIN,KOUT,NDIM,NDIM)
      ELSE
C COMPUTE U AS DESCRIBED ABOVE
      IF (ICBM.NE.IRHM) THEN
      WRITE(KOUT,2)'NOTE THAT IRHM MUST EQUAL ICBM TO USE THIS METHOD OF
     1 CALCULATING U, ICBM=',ICBM,'  IRHM= ' ,IRHM
      GO TO 5
      END IF
      CALL EQUATE(UM1,PHIM,IRFM,IRFM)
C  GMINV DESTROYS THE CALLING ARRAY
      MT=1
      CALL GMINV(IRFM,IRFM,UM1,UM2,MR,MT)
      CALL MAT1(HM,UM2,IRHM,IRFM,IRFM,UM1)
      CALL MAT1(UM1,BMD,IRHM,IRFM,ICBM,UM2)
      CALL GMINV(IRHM,ICBM,UM2,U,MR,MT)
      END IF
      GO TO 5
3     WRITE(KOUT,12)' '
C  CALCULATE RKFSS
      CALL EQUATE(UM1,PHIM,IRFM,IRFM)
C  GMINV DESTROYS CALLING ARRAY
      MT=1
      CALL GMINV(IRFM,IRFM,UM1,UM2,MR,MT)
      CALL MAT1(UM2,BMD,IRFM,IRFM,ICBM,UM1)
      CALL MAT1(UM1,U,IRFM,ICBM,ICBM,UM2)
      CALL SCALE(RKFSS,UM2,IRFM,IRHM,SQ)
      IO=5
      MSG='RKFSS, PICKED DIRECTLY IS'
      CALL MMATIO(RKFSS,IRFM,IRHM,IO,KIN,KOUT,NDIM,NDIM)
      GO TO 5
4     RETURN
      END
*DECK PRIMIT
      SUBROUTINE PRIMIT(UM2,U,ICBM,GCSTR,S,IRFM,BMD,UM1,PHIPRM,X,XPRIM,
     1 PHIM)
C    THIS SUBROUTINE COMPUTES THE PRIMED QUANTITIES NEEDED WHEN USING
C    KLEINMAN RICCATI SOLVER WITH NON ZERO CROSS COST WEIGHTING
C    MATRIX UXU
      REAL UM2(NDIM,NDIM),GCSTR(NDIM,NDIM),S(NDIM,NDIM),
     1BMD(NDIM,NDIM),UM1(NDIM,NDIM),PHIPRM(NDIM,NDIM),
     1 X(NDIM,NDIM),XPRIM(NDIM,NDIM),PHIM(NDIM,NDIM)
      REAL COM1(1),COM2(1)
      COMMON /MAIN1/ NDIM,NDIM1,COM1
      COMMON /MAIN2/COM2
      COMMON /INOU/KIN,KOUT,KPUNCH
C  NOW COMPUTE XPRIM,  PHIPRM
      CALL EQUATE(UM2,U,ICBM,ICBM)
C GMINV DESTRIYS THE CALLING ARRAY
      MT=1
      CALL GMINV(ICBM,ICBM,UM2,GCSTR,MR,MT)
C  GCSTR= UI ICBM  X  ICBM
      CALL MAT4(GCSTR,S,ICBM,ICBM,IRFM,UM2)
CUM2= UI(ST)   ICBM  X  IRFM
      CALL MAT1(BMD,UM2,IRFM,ICBM,IRFM,UM1)
      C1=-1.0
      CALL MADD1(IRFM,IRFM,PHIM,UM1,PHIPRM,C1)
```

```
C PHIPRM= PHIM-BMD(UI)ST   IRFM   X  IRFM
      CALL MAT1(S,UM2,IRFM,ICBM,IRFM,UM1)
      CALL MADD1(IRFM,IRFM,X,UM1,XPRIM,C1)
C  XPRIM= X-S(UI)ST   IRFM  X  IRFM
C
      RETURN
      END
*DECK RGS
      SUBROUTINE RGS(GCSTR,RKFSS,GCX,GCY,GCZ,BCY,BCZ,FC,YD,
     1 UM1,UM2,UM3,UM4,UM5,UM6,UM7,UM8,UM9,UM10,UU1,UU2,
     1 UMA,UMB,UMC,UMD,UME,UMF,FT,BT,GT,HT,QT,RT,
     1 FM,BM,GM,HM,QM,RM,XO,PO,UXX,UUU,UXU,FA,BA,GA,QA,GUA,
     1MXA,PXA,RA,PXUA,GCZA,IRY,IFLGCZ,IFLGSD,
     1MU,PUMAX,PUMIN,PXTMAX,PXTMIN,MUMAX,MUMIN,MXAMAX,MXAMIN,
     1 PUOUT,PXTOUT,MXTOUT,MUOUT,UU3,UU4)
      REAL COM1(1),COM2(1)
C
C  THIS SUBROUTINE SETS UP A CONTINUOUS, DISCRETIZED CONTINUOUS OR A
C   SAMPLED DATA LQG CONTROLLER BASED ON USERS REQUEST.  EACH
C   CONTROLLER IS PUT INTO THE PROPER FORMAT FOR  THE PERFORMANCE
C   ANALYSIS SUBROUTINE, PERFAL.
      CHARACTER MSG*60
      REAL FT (NDIM,NDIM),BT(NDIM,NDIM),GT(NDIM,NDIM),
     1 HT  (NDIM,NDIM),RM  (NDIM,NDIM),FM(NDIM,NDIM),
     1BM(NDIM,NDIM),GM  (NDIM,NDIM),XO(NDIM),
     1HM(NDIM,NDIM),QM  (NDIM,NDIM),PO(NDIM,NDIM),
     1QT(NDIM,NDIM),RT  (NDIM,NDIM),
     1GCSTR(NDIM,NDIM),RKFSS(NDIM,NDIM),
     1UU1(NDIM),UU2(NDIM),UXU(NDIM,NDIM),
     1UM1  (NDIM,NDIM),UM2 (NDIM,NDIM),UM3(NDIM,NDIM),
     1UM4  (NDIM,NDIM),UM5 (NDIM,NDIM),UM6(NDIM,NDIM),
     1UM7  (NDIM,NDIM),UM8 (NDIM,NDIM),UM9(NDIM,NDIM),
     1UM10 (NDIM,NDIM),UMA(NDIM2,NDIM2),UMB(NDIM2,NDIM2),
     1UMC(NDIM2,NDIM2),UMD(NDIM2,NDIM2),UME(NDIM2,NDIM2),
     1UMF(NDIM2,NDIM2),UUU(NDIM,NDIM),UXX(NDIM,NDIM),
     1FA(NDIM2,NDIM2),BA(NDIM2,NDIM2),GA(NDIM2,NDIM2),
     1MXA(NDIM2),GCX(NDIM,NDIM),GCZ(NDIM,NDIM),
     1QA(NDIM2,NDIM2),PXA(NDIM2,NDIM2),PXUA(NDIM2,NDIM2),
     1MUOUT(NDIM),MXTOUT(NDIM),PXTOUT(NDIM)
        REAL PUOUT(NDIM),YD(NDIM3),GUA(NDIM2,NDIM2),
     1MXAMIN(NDIM),MXAMAX(NDIM),MUMIN(NDIM),
     1MUMAX(NDIM),PXTMIN(NDIM),PXTMAX(NDIM),
     1PUMIN(NDIM),PUMAX(NDIM),GCZA(NDIM2,NDIM2),
     1RA(NDIM2,NDIM2),BCY (NDIM,NDIM),BCZ(NDIM,NDIM),
     1UU3(NDIM2),UU4(NDIM2),
     1GCY  (NDIM,NDIM),FC  (NDIM,NDIM)
      INTEGER IFLGCZ,IRY
      REAL MU(NDIM)
      COMMON /RNTIM/ RNTIME,DELTIM
      COMMON /MAIN2/COM2
      COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON / INOU/ KIN,KOUT,KPUNCH
      COMMON /MAIN4/NDIM2,NDIM3
      COMMON /MAUN5/ MSG
      COMMON /MAIN6/ ICBT,ICBM,ICFA,ICGA,ICGM,ICGT,ICQA,IRFA,IRFM,IRFT,
     1   IRHT,IRQA,IO,LQG,IRHM,NUMDTS
C
      WRITE(KOUT,1)'ENTER A C FOR CONTINUOUS TIME LQG CONTROLLER AND A
     1 D FOR A SAMPLED DATA LQG CONTROLLER)'
      READ(KIN,12,END=2933)MSG
      IF (MSG.EQ.'C') THEN
      IFLGSD=0
      CALL CLQGRS(GCSTR,FM,BM,RKFSS,HM,GCX,GCY,GCZ,BCY,BCZ,FC,YD,
     1  RM,QM,FT,BT,GT,QT,RT,HT,
     1IRY,IFLGCZ,UM1,UM2,UM3
     1 ,PO,GM,UM4,UM5,UM6,UU1,UU2,UUU,UXX,XO,UXU,UM7,UM8)
      IF (IO.EQ.0)THEN
      GO TO 2933
      END IF
      CALL FRMAUG(QT,RT,FT,BT,GCZ,HT,GCX,BCZ,FC,GCY,BCY,GT,XO,PO,FA,BA,
     1 GA,QA,GUA,UM1,UM2,UMA,UMB,UMC,UMD,UME,UMF,
     1 MXA,PXA,RA,PXUA,GCZA,IRY,IFLGCZ,IFLGSD)
      WRITE(KOUT,1)' '
      WRITE(KOUT,1)'DO YOU WISH TO CALCULATE THE EIGENVALUES OF THE CLOS
     1ED-LOOP F MATRIX?  Y   OR   N)'
      READ(KIN,12)MSG
```

```
      IF (MSG.EQ.'Y') THEN
      NSAV1=NDIM
      NDIM=NDIM2
      NDIM1=NDIM2+1
      WRITE(KOUT,*)'THE EIGENVALUES OF THE CLOSED-LOOP F MATRIX  ARE...'
      CALL MEIGN(FA,WV3,WV4,IRFA,WMA)
      NDIM=NSAV1
      NDIM1=NDIM+1
      END IF
      CALL MYINTG(WMA,WMB,WMC,WME,GA,QA,FA,BA,IRFA,ICGA,IRY,IRQA)
      ELSE
C     SOME SAMPLED DATA CONTROLLER IS WANTED
        IFLGSD=1
       WRITE(KOUT,*)'DO YOU WISH TO MERELY DISCRETIZE THE CONTINUOUS TIME
      1 CONTROLLER,   Y  OR   N)'
       READ(KIN,12,END=2933)MSG
       IF (MSG.EQ.'Y')THEN
       WRITE(KOUT,*)'*********NOTE THAT WHEN ENTERING THE TIME INCREMENT
      1 IN THE CONTINUOUS TIME CONTROLLER SET UP,   REMEMBER IT WILL BECO
      1ME THE SAMPLE TIME FOR THE DISCRETIZED CONTROLLER*********'
       CALL CLQGRS(GCSTR,FM,BM,RKFSS,HM,GCX,GCY,GCZ,BCY,BCZ,FC,YD,
      1   RM,QM,FT,BT,GT,QT,RT,HT,IRY,IFLGCZ,WM1,WM2,WM3,PO,GM,
      1   WM4,WM5,WM6,WV1,WV2,WVV,WXX,XO,WXV,WM7,WM8)
       IF (IO.EQ.0)THEN
       GO TO 2933
       END IF
       CALL DSCRTZ(WM1,WM2,WM3,WM4,IRFM,DELTIM,FC,BCY,IRY,BCZ,IRHM,
      1 WM5,WM6,GCSTR,GT,QT,FT,BT,IRFT,ICGT,ICBT,IRHT,RKFSS,RT,GCX,
      1 ICBM)
       WRITE(KOUT,*)' '
       WRITE(KOUT,*)'DO YOU WANT TO CALCULATE THE EIGENVALUES OF THE STAT
      1E TRANSITION MATRIX FOR THE DISCRETIZED CONTROLLER?  Y  OR  N)'
       READ(KIN,12)MSG
       IF (MSG.EQ.'Y') THEN
       WRITE(KOUT,*)'THE EIGENVALUES OF THE STATE TRANSITION MATRIX FOR T
      1HE DISCRETIZED CONTROLLER ARE...'
       CALL MEIGN(WM2,WV1,WV2,IRFM,BCY)
       END IF
C  RKFSS, IN PRECEDING CALL STATEMENT ARE MERELY DUMMY WORK SPACES
C  GCSTR CONTAINS BCZD UPON RETURN FROM DSCRTZ
       CALL FRMAUG(WM6,RKFSS,WM5,GCSTR,GCZ,HT,GCX,WM4,WM2,GCY,WM3,WM1,XO,
      1 PO,FA,BA,GA,QA,GUA,BCY,FC,WMA,WMB,WMC,WMD,WME,WMF,MXA,PXA,RA,
      1 PXUA,GCZA,IRY,IFLGCZ,IFLGSD)
       ELSE
       IFLGCZ=0
       CALL DLQGRS(GCX,GCY,GCZ,BCY,BCZ,WM3,FC,WM2,WM5,WM1,WM4,FM,BM,
      1 QM,GM,RM,HM,GT,QT,FT,BT,RT,HT,WXX,WUU,GCSTR,RKFSS,YD,
      1 IRY,IFLGCZ,WXU,WM6,WM7,WM8,WM9,WM10,WV1,WV2)
       CALL FRMAUG(WM1,WM2,WM3,WM4,GCZ,HT,GCX,BCZ,FC,GCY,BCY,WM5,XO,
      1 PO,FA,BA,GA,QA,GUA,RKFSS,GCSTR,WMA,WMB,WMC,WMD,WME,WMF,MXA,PXA,
      1 RA,PXUA,GCZA,IRY,IFLGCZ,IFLGSD)
C  RKFSS,GCSTR ARE DUMMY WORK SPACE IN CALL TO FRMAUG
       END IF
       NSAV1=NDIM
       NSAV2=NDIM1
       NDIM1=NDIM2+1
       NDIM=NDIM2
       CALL EQUATE(WMA,FA,IRFA,IRFA)
       CALL EQUATE(WMC,BA,IRFA,IRY)
        CALL MAT1(GA,QA,IRFA,IRQA,IRQA,WME)
       CALL MAT4(WME,QA,IRFA,IRQA,IRFA,WMB)
       NDIM=NSAV1
       NDIM1=NSAV2
C  NOTE FC AND BCY IN CALL TO FRMAUG ARE DUMMY WORKSPACES
       END IF
       RETURN
2933   IO=0
12     FORMAT(A1)
       END
*DECK XSU
       SUBROUTINE XSU(X,S,U,PHIJO,BJO,PHII,INTPII,PHIJ,BJ,
      1 TEMP,TEMP1,TEMP2,FM,BM,IRFM,ICBM,WXX,WUU,WXU,PHIT)
C   THIS ROUTINE COMPUTES X S  AND U AT TIME T-SUB-I FOR USE IN THE
C   SAMPLED DATA CONTROLLERDETERMINISTIC GAIN CALCULATION.
C   THIS ROUTINE APPROXIMATES THE INTEGRALSREQUIRED (SEE MAYBECK,
C   EQUATIONS 14.25) BY TREATING  TIME VAYING ENTITIES IN  THE
C   INTEGRNDS AS CONSTANTS OVER SOME SUBINTERVAL OF THE SAMPLE TIME
```

```
C   THAT IS CHOSEN BY THE USER
      REAL X  (NDIM,NDIM), S(NDIM,NDIM), U(NDIM,NDIM),
     1PHIJO(NDIM,NDIM),BJO(NDIM,NDIM),PHII(NDIM,NDIM),
     1INTPII  (NDIM,NDIM),PHIJ(NDIM,NDIM),BJ(NDIM,NDIM),
     1TEMP (NDIM,NDIM),TEMP1(NDIM,NDIM),TEMP2(NDIM,NDIM),
     1FM(NDIM,NDIM),BM  (NDIM,NDIM),UXX(NDIM,NDIM),
     1UUU  (NDIM,NDIM),UXU(NDIM,NDIM),PHIT(NDIM,NDIM)
      REAL COM1(1),COM2(1)
      CHARACTER MSG*60
       COMMON /MAIN1/NDIM,NDIM1,COM1
      COMMON/MAIN2/COM2
      COMMON /INOU/ KIN,KOUT,KPUNCH
       COMMON /RNTIM/ RNTIME,DELTIM
      COMMON /MAUNS/ MSG
      IZIT=0
      DO 782 IJK=1,1000
C   GIVE USER UP TO 1000 CHANCES TO CHOOSE DIFFERENT SUBINTERVAL LENGTH

      IF (IZIT.EQ.1)THEN
      WRITE(KOUT,12)' '
      WRITE(KOUT,*)'DO YOU WISH TO RECOMPUTE X,S,U, BASED ON A DIFFERENT
     1 SUBINTERVAL LENGTH,  Y  OR  N)'
      READ(KIN,11)MSG
11    FORMAT(A1)
12    FORMAT(A1,/)
      IF (MSG.EQ.'N')THEN
      RETURN
      END IF
      END IF
      IZIT=1
      WRITE(KOUT,*)'ENTER THE NUMBER OF SUBINTERVALS TO USE IN THE APPRO
     1XIMATIONS  OF INTEGRALS NEEDED TO CALCULATE X, S, AND U (SUGGEST 5
     1 OR MORE )>'
      READ(KIN,*)INTVAL
C  NOW INITIALIZE VARIABLES REQUIRED IN CALCULATIONS
      C1=1.0
      CALL IDNT(IRFM,PHIJO,C1)
      C1=0.0
       CALL IDNT(IRFM,X,C1)
      CALL IDNT(ICBM,U,C1)
        DO 13 I=1,IRFM
       DO 13 J=1,ICBM
      BJO(I,J)=0
13     S(I,J)=0
C  INITIALIZATION COMPLETE,   NOW COMPUTE PHIJ, INTPHJ FOR SUBINTERVAL
C  PHI,INTPHI ARE APPROXIMATED BY TAKING AVERAGE OF VALUES AT THE
C BEGINNING ,END AND 8 POINTS IN THE MIDDLE OF EACH SUBINTERVAL
C   THIS MEANS 9 SUB SUB INTERVAL POINTS TO BE CALCULATED.  HOWEVER, O
C NLY 1 CALL TO INTEGRATE ROUTINIS REQUIRED SINCE FM IS A CONSTANT
C MATRIX
      DEL=0.0
      SUBINT=DELTIM/(4*INTVAL)
      DO 23 JK=1,INTVAL
C  COMPUTE PHIJ BJ AND THEN UPDATE PHIJO,BJO
C   FOR EACH SUBINTERVAL
      C1=1.0
      DO 371 INTUL=1,4
      DEL=DEL+SUBINT
      CALL DSCRT(IRFM,FM,DEL,PHII,INTPII,5)
      CALL MADD1(IRFM,IRFM,PHIJO,PHII,TEMP,C1)
      CALL EQUATE(PHIJO,TEMP,IRFM,IRFM)
C PHIJO=PHIJO+PHII
      CALL MAT1(INTPII,BM,IRFM,IRFM,ICBM,TEMP1)
      CALL MADD1(IRFM,ICBM,BJO,TEMP1,TEMP,C1)
      CALL EQUATE(BJO,TEMP,IRFM,ICBM)
C  BJO=BJO+INTPII*BM
371   CONTINUE
C NOW CALCULATE BJ PHIJ
      C1=0.2
      CALL SCALE(PHIJ,PHIJO,IRFM,IRFM,C1)
      CALL SCALE(BJ,BJO,IRFM,ICBM,C1)
C  BJ,  PHIJ   NOW AVAILABLE FOR THIS SUBINTERVAL
C  RESET PHIJO,BJO
      CALL EQUATE(PHIJO,PHII,IRFM,IRFM)
      CALL EQUATE(BJO,TEMP1,IRFM,ICBM)
C  NOW UPDATE X,S,U FOR THIS SUBINTERVAL
C  X=SUM OF (PHIJT*UXX*PHIJ*(DELTIM/INTVAL)) FOR ALL JK
```

```
C  S=SUM OF ((PHIJT*UXX*BJ+PHIJT*UXU)*(DELTIM/INTVAL))
      CALL MAT4A(PHIJ,UXX,IRFM,IRFM,IRFM,TEMP)
C TEMP= PHIJT * UXX
      CALL MAT1(TEMP,PHIJ,IRFM,IRFM,IRFM,TEMP1)
      CALL MAT1(TEMP,BJ,IRFM,IRFM,ICBM,TEMP2)
      C1A=DELTIM/INTVAL
      CALL MADD1(IRFM,IRFM,X,TEMP1,TEMP,C1A)
      CALL EQUATE(X,TEMP,IRFM,IRFM)
C  X IS NOW UPDATED FOR THIS SUB-SUB INTERVAL
      CALL MAT4A(PHIJ,UXU,IRFM,IRFM,ICBM,TEMP1)
      C1=1.0
      CALL MADD1(IRFM,ICBM,TEMP2,TEMP1,TEMP,C1)
      CALL MADD1(IRFM,ICBM,S,TEMP,TEMP1,C1A)
      CALL EQUATE(S,TEMP1,IRFM,ICBM)
C  S UPDATE FOR THIS SUB-SUB INTERVAL NOW COMPLETE
C
C  U=SUM OF((BJT*UXX*BJ+UUU+BJT*UXU+UXUT*BJ)*(DELTIM/INTVAL)) FOR ALL JK
      C1=1.0
      CALL MAT3A(ICBM,IRFM,BJ,UXX,TEMP)
      CALL MADD1(ICBM,ICBM,TEMP,UUU,TEMP1,C1)
      CALL MAT4A(BJ,UXU,ICBM,IRFM,ICBM,TEMP)
      CALL MADD1(ICBM,ICBM,TEMP1,TEMP,TEMP2,C1)
      CALL MAT4A(UXU,BJ,ICBM,IRFM,ICBM,TEMP)
      CALL MADD1(ICBM,ICBM,TEMP2,TEMP,TEMP1,C1)
      CALL MADD1(ICBM,ICBM,U,TEMP1,TEMP,C1A)
      CALL EQUATE(U,TEMP,ICBM,ICBM)
C  U  UPDATE FOR THIS SUB-SUB INTERVAL NOW COMPLETE
 23    CONTINUE
       IO=5
       MSG='X(TI) IS'
       CALL MMATIO(X,IRFM,IRFM,IO,KIN,KOUT,NDIM,NDIM)
       MSG='U(TI) IS'
       CALL MMATIO(U,ICBM,ICBM,IO,KIN,KOUT,NDIM,NDIM)
       MSG='S(TI) IS'
       CALL MMATIO(S,IRFM,ICBM,IO,KIN,KOUT,NDIM,NDIM)
  782   CONTINUE
        RETURN
        END
```

Appendix C

Software Considerations

The program, LQGRP, is designed to run interactively at a remote terminal for the CYBER computer system at the AFIT. The program memory requirements far exceed the 65000 octal word list for using a remote terminal so a special loading technique, segmentation, is used. The segmentation loader actively manages the subroutines so that only those required for a particular program phase are loaded into the CYBER central memory. The remainder of the subroutines reside on a disk file.

One reason this program requires a large amount of memory space is merely that state-space system models require many matrices and vectors to describe them. Another reason is that the library of subroutines for basic matrix manipulation such as multiplication, addition and inversion, as well as those used for more complex operations such as integration and solving the matrix Riccati equation require all matrix arguments to be square and to have the same declared size (Refs 4 and 5). Because of this, a large amount of memory space is required even though it might otherwise be unnecessary considering that the number of inputs and outputs in many physical systems is generally less than the number of states. For example, in the model used in this thesis there is one input and one output and five states. However, in order to use Kleinman's routines all matrices ($\underline{F}_t$, $\underline{B}_t$, $\underline{G}_t$,...) must be declared square and at least of dimension 5. Thus, just for $\underline{B}_t$ and $\underline{H}_t$ 20 memory locations are never used.

In addition to the fact that Kleinman's routines require unnecessary storage space, some of the specialized software routines for solving the matrix Riccati equations require that some of the equations in the body of this thesis be rearranged in order to use the software. Eqs (2.13) and (2.92) through (2.98) encompass the most significant modifications.

The Riccati equation solver, subroutine MRIC, is designed for controller calculations and thus solves Eq (2.14) directly. To solve Eq (2.13) using MRIC, it is important to note that MRIC transposes the $\underline{F}$ matrix in the call statement. Therefore, before using MRIC to solve Eq (2.13), the $\underline{F}$ matrix must be transposed before the call to MRIC.

The modifications to use subroutine DRIC to solve Eqs (2.92) through (2.98) are more extensive. First, it must be recognized that DRIC solves for the steady-state solution of $\underline{K}_c$ defined in Eq (2.106). It is also necessary to note that DRIC will not solve for $\underline{K}_c$ with a non-zero $\underline{S}$ matrix (defined in Eq (2.97)) unless the proper transformation is used; this transformation will be explained later in this appendix.

For the case of constant controller model matrices, stationary noise inputs and constant cost-weighting matrices, Eqs (2.92) through (2.98) can be integrated directly or they can be solved using the equivalent forms of the equations (Ref 10),

$$\underline{\Phi}(t_{i-1}, t_i) = \exp \underline{F}_c(t_{i+1} - t_i) \qquad (C.1)$$

$$\underline{B}_d(t_i) = \int_{t_i}^{t_{i+1}} \underline{\Phi}(t_{i+1}, \tau)\, d\tau\, \underline{B}_c \tag{C.2}$$

$$\underline{Q}_d(t_i) = \int_{t_i}^{t_{i+1}} \underline{\Phi}(t_{i+1}, \tau)\, \underline{G}\, \underline{Q}\, \underline{G}^T\, \underline{\Phi}^T(t_{i+1}, \tau)\, d\tau \tag{C.3}$$

$$\underline{X}(t_i) = \int_{t_i}^{t_{i+1}} \underline{\Phi}^T(t, t_i)\, \underline{W}_{xx}\, \underline{\Phi}(t, t_i)\, dt \tag{C.4}$$

$$\underline{U}(t_i) = \int_{t_i}^{t_{i+1}} \Big(\overline{\underline{B}}^T(t, t_i)\, \underline{W}_{xx}\, \overline{\underline{B}}(t, t_i) + \underline{W}_{uu} + \overline{\underline{B}}^T(t, t_i)\, \underline{W}_{xu} + \underline{W}_{xu}^T\, \overline{\underline{B}}(t, t_i) \Big)\, dt \tag{C.5}$$

$$\underline{S}(t_i) = \int_{t_i}^{t_{i+1}} \Big(\underline{\Phi}^T(t, t_i)\, \underline{W}_{xx}\, \overline{\underline{B}}(t, t_i) + \underline{\Phi}^T(t, t_i)\, \underline{W}_{xu} \Big)\, dt \tag{C.6}$$

where

$$\overline{\underline{B}}(t, t_i) = \int_{t_i}^{t} \underline{\Phi}(t, \tau)\, d\tau\, \underline{B}_c \tag{C.7}$$

The forms in Eqs (C.1) through (C.7) are used because they can be solved using the Kleinman subroutines whereas forms in Eqs (2.92) through (2.98) cannot. It is necessary to make approximations to $\underline{\Phi}(t, t_i)$ and $\overline{\underline{B}}(t, t_i)$ in order to solve Eqs (C.4) through (C.6) using the Kleinman routines.

The approximation that is used breaks up the interval from t_i to t_{i+1} into N, equal subintervals. During each subinterval, $\underline{\Phi}(t, t_i)$ and $\overline{\underline{B}}(t, t_i)$ are treated as constants. Smaller subintervals provide better approximation but require more execution time, therefore a tradeoff is required. The

subroutine that accomplishes the calculations of $\underline{X}(t_i)$, $\underline{S}(t_i)$ and $\underline{U}(t_i)$ allows the user to input the desired number of subintervals, N. The routine then displays $\underline{X}(t_i)$, $\underline{U}(t_i)$ and $\underline{S}(t_i)$ for that N, and allows the user to change the value of N and recompute.

As stated above, constant values will be used over each subinterval for $\underline{\Phi}(t, t_i)$ and $\underline{\bar{B}}(t, t_i)$. The constant values used for the j^{th} subinterval, $\underline{\Phi}_j$ and $\underline{B}_j$ respectively, are the average-values of $\underline{\Phi}(t, t_i)$ and $\underline{\bar{B}}(t, t_i)$ for the j^{th} subinterval. $\underline{\Phi}(t, t_i)$ and $\underline{\bar{B}}(t, t_i)$ are calculated at the beginning and end of the j^{th} subinterval as well as at three equally spaced points between the beginning and the end. These are the values that are averaged to form $\underline{\Phi}_j$ and $\underline{B}_j$. Thus, for $\Delta t = (t_{i+1} - t_i)$

$$\underline{\Phi}_j = \left[\text{average of } \underline{\Phi}(t, t_i) \text{ for all } t \in \left[t_i + (j-1)\frac{\Delta t}{N},\ t_i + j\frac{\Delta t}{N} \right] \right] \tag{C.8}$$

$$\underline{B}_j = \left[\text{average of } \underline{\bar{B}}(t, t_i) \text{ for all } t \in \left[t_i + (j-1)\frac{\Delta t}{N},\ t_i + j\frac{\Delta t}{N} \right] \right] \tag{C.9}$$

The Kleinman subroutine DSCRT simultaneously returns $\underline{\Phi}(t, 0)$ and $\int_0^t \underline{\Phi}(t, \tau)\, d\tau$ and is the only major matrix subroutine needed in order to compute $\underline{\Phi}_j$ and $\underline{B}_j$.

When the approximations just described for $\underline{\Phi}_j$ and $\underline{B}_j$ are used, Eqs (C.4) through (C.6) become

$$\underline{X}(t_i) = \sum_{j=1}^{N} \left(\underline{\Phi}_j^T \, \underline{W}_{xx} \, \underline{\Phi}_j \right) \frac{\Delta t}{N} \qquad (C.10)$$

$$\underline{U}(t_i) = \sum_{j=1}^{N} \left(\underline{B}_j^T \, \underline{W}_{xx} \, \underline{B}_j + \underline{W}_{uu} + \underline{B}_j^T \, \underline{W}_{xu} + \underline{W}_{xu}^T \, \underline{B}_j \right) \frac{\Delta t}{N} \qquad (C.11)$$

$$\underline{S}(t_i) = \sum_{j=1}^{N} \left(\underline{\Phi}_j \, \underline{W}_{xx} \, \underline{B}_j + \underline{\Phi}_j \, \underline{W}_{xu} \right) \frac{\Delta t}{N} \qquad (C.12)$$

Once values have been obtained for $\underline{\Phi}(t_{i+1}, t_i)$, $\underline{B}_d(t_i)$, $\underline{Q}_d(t_i)$, $\underline{X}(t_i)$, $\underline{U}(t_i)$ and $\underline{S}(t_i)$ it is still necessary to transform Eqs (C.1) through (C.6) in order to use DRIC, the matrix Riccati equation solver. The transformation of the LQ sampled-data controller is (Refs 7 and 10)

$$\underline{x}(t_{i+1}) = \underline{\Phi}'(t_{i+1}, t_i) \, \underline{x}(t_i) + \underline{B}_d(t_i) \, \underline{u}'(t_i) \qquad (C.13)$$

where

$$\underline{\Phi}'(t_{i+1}, t_i) = \underline{\Phi}(t_{i+1}, t_i) - \underline{B}_d(t_i) \, \underline{U}^{-1}(t_i) \, \underline{S}^T(t_i) \qquad (C.14)$$

$$\underline{u}'(t_i) = \underline{u}(t_i) + \underline{U}^{-1}(t_i) \, \underline{S}^T(t_i) \, \underline{x}(t_i) \qquad (C.15)$$

Now the quadratic cost equation that is minimized by $\underline{u}(t_i)$ is

$$J' = \tfrac{1}{2} \, \underline{x}^T(t_{N+1}) \, \underline{X}_f \, \underline{x}(t_{N+1}) + \sum_{i=0}^{N} \tfrac{1}{2} \left[\underline{x}^T(t_i) \, \underline{X}' \, \underline{x}(t_i) + \underline{u}'^T(t_i) \, \underline{U}(t_i) \, \underline{u}'(t_i) \right] \qquad (C.16)$$

where

$$\underline{X}'(t_i) = \underline{X}(t_i) - \underline{S}(t_i) \, \underline{U}^{-1}(t_i) \, \underline{S}^T(t_i) \qquad (C.17)$$

Now Eqs (2.105) and (2.106) can be put into the format of DRIC to solve for steady-state values of $\underline{K}_c'$ and $\underline{G}_c^{*'}$. In the

notation used in this thesis DRIC solves

$$\underline{K}_c' = \underline{\Phi}^T \underline{K}_c' \left(\underline{I} + \underline{B}\, \underline{U}^{-1}\, \underline{B}^T\, \underline{K}_c \right)^{-1} \underline{\Phi} + \underline{K} \qquad \text{(C.18)}$$

which can be obtained from Eqs (2.105) and (2.106) by setting $\underline{S}(t_i) = \underline{0}$ and using the matrix inversion (Ref 8)

$$\left(\underline{I} - \underline{Y}^T \left(\underline{Z}\, \underline{Y}^T + \underline{W}\right)^{-1} \underline{Z}\right) = \left(\underline{I} + \underline{Y}^T\, \underline{W}^{-1}\, \underline{Z}\right)^{-1} \qquad \text{(C.19)}$$

and defining $\underline{Y}^T = \underline{B}$, $\underline{W} = \underline{U}$, $\underline{Z} = \underline{B}^T\, \underline{K}_c$ and $\underline{I}$ to be the identity matrix. Note that (C.19) holds only when $\underline{W}$ ($\underline{U}$ in this case) is positive definite.

Once $\underline{G}_c^{*'}$ is computed using the results from DRIC, $\underline{G}_c^*$ can be computed from

$$\underline{G}_c^* = \underline{G}_c^{*'} + \underline{U}^{-1}\, \underline{S}^T \qquad \text{(C.20)}$$

Note, for $\underline{W}_{xu} \neq 0$ in the continuous-time case, the transformation described in Eqs (C.13) through (C.17) and (C.20) can be used if, in those equations, the $\underline{\Phi}$s are replaced by $\underline{F}$s, $\underline{B}_d$s by $\underline{B}$s and $\underline{S}$s by $\underline{W}_{xu}$s. Now Eqs (2.11) and (2.12) can be used for the $\underline{G}_c^*$ of the transformed system.

As can be seen from the foregoing discussion, while Kleinman's matrix software routines are very powerful, care must be taken in order to arrange the equations properly so that they match the arrangements used in these routines (Ref 5).

Appendix D

Software Performance Verification

To verify the performance of the software developed to support this thesis, several test cases with known results were used.

For the continuous-time software, 3 test cases were used. The first is example 14.25 in Maybeck (Ref 10). The second is example 5.3 in Kwakernaak and Sivan (Ref 13). The third is the example given by Doyle and Stein (Ref 2). The details of these examples appear in Table D.1. In all cases, the truth model and the controller design model are equivalent.

For the sampled-data software, the same three test cases are used. The intent is to show that as the sample period decreases, the steady-state value of $\underline{P}_{x_t x_t}(t_i)$ approaches the steady-state value of its continuous-time counterpart, $\underline{P}_{x_t x_t}(t)$.

Table D.2 presents the results of the software verification run for both the continuous-time LQG regulator and the discretized continuous-time LQG regulator. The results of the continuous-time case are in agreement with the sources of the examples (Refs 2 and 7). By examining Table D.2, it is evident that as the sample period decreased, the performance of the discretized controllers approached that of their continuous-time counterparts. Also note in the table that for a sufficiently large sample period, the discretized controllers do not approach the steady-state values of their continuous-time counterparts and may even diverge. This is ex-

pected since the approximations used in discretization are not valid for large sample periods.

Table D.3 presents the results of the software verification runs for the sampled-data LQG regulator software. As in the discretized continuous-time case presented in Table D.2, the performance of the sampled-data controllers approaches that of their counterpart continuous-time controller for sufficiently small sample periods.

TABLE D.1

Test Cases for Software Performance Verification

Test Case 1 (Ex 14.25 in Maybeck (Ref10) with T=2)	
Program Inputs	**Expected Program Outputs**
$F = [-0.5]$ $\quad G = [1]$ $B = [0.5]$ $\quad H = [1]$ $Q = [2]$ $\quad R = [3]$ $P_o = [1]$ $\quad x_{t_o} = [0]$ $W_{uu} = [4]$ $W_{xu} = [0]$ $\quad W_{xx} = [5]$	$\bar{K}_f = [0.457]$ $\bar{G}_c^* = [0.5]$ $m_{x_t} = m_u = [0]$ $P_{x_t} = [1.79]$ $P_u = [0.1045]$

Test Case 2 (Example 5.3 in Kwakernaak (Ref 7))	
Program Inputs	**Expected Program Outputs**
$\underline{F} = \begin{bmatrix} 0 & 1 \\ 0 & -4.6 \end{bmatrix}$ $\quad \underline{G} = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$ $\underline{B} = \begin{bmatrix} 0 \\ 0.787 \end{bmatrix}$ $\quad \underline{H} = [1 \quad 0]$ $\underline{R} = [10^{-7}]$ $\underline{Q} = [10]$ $\underline{x}_{t_o} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\underline{P}_o = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\underline{W}_{xx} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ $\underline{W}_{uu} = [0.00002]$ $\underline{W}_{xu} = [0]$	$\bar{K}_f = \begin{bmatrix} 40.36 \\ 814.3 \end{bmatrix}$ $\bar{G}_c^* = [223.6 \quad 18.69]$ $\underline{m}_{xt} = [0 \quad 0]^T$ $\underline{m}_u = [0]$ $\underline{P}_{x_t} = \begin{bmatrix} 0.00004562 & 0 \\ 0 & 0.00619 \end{bmatrix}$ $\underline{P}_u = [2.26]$

TABLE D.1 (con't)

Test Case 3 (Example in Doyle and Stein, Without Robustness Recovery, (Ref 2))	
Program Inputs	**Expected Program Outputs**
$\underline{F}=\begin{bmatrix}0 & 1\\ -3 & -4\end{bmatrix}$ $\underline{B}=\begin{bmatrix}0\\ 1\end{bmatrix}$ $\underline{G}=\begin{bmatrix}35\\ -61\end{bmatrix}$ $\underline{H}=[2 \quad 1]$ $\underline{R}=[1]$ $\underline{Q}=[1]$ $\underline{x}_{t_o}=[0 \quad 0]$ $\underline{P}_{t_o}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ $\underline{W}_{xx}=\begin{bmatrix}2800 & 473.3\\ 473.3 & 80\end{bmatrix}$ $\underline{W}_{uu}=[1]$ $\underline{W}_{xu}=[0]$	$\underline{K}_f=[30 \quad -50]$ $\underline{G}_c^*=[50 \quad 10]$ $\underline{m}_{x_t}=[0 \quad 0]$ $\underline{m}_u=[0]$ $\underline{P}_{x_t}=\begin{bmatrix}221 & -613\\ -613 & 2070\end{bmatrix}$ $\underline{P}_u=[4(10)^4]$
Test Case 4 (Same as Test Case 3 Except that Doyle and Stein Technique is Applied)	
Program Inputs	**Expected Program Outputs**
all program input matrices same as for test case 3. Then when $\underline{V}=\underline{I}$ and $q^2=100$	$\underline{K}_f=[26.8 \; - \; 40.2]^T$ $\underline{P}_{x_t}=\begin{bmatrix}236 & -613\\ -613 & 1810\end{bmatrix}$
$q^2=500$	$\underline{K}_f=[20.4 \; - \; 17.7]^T$ $\underline{P}_{x_t}=\begin{bmatrix}268 & -613\\ -613 & 1500\end{bmatrix}$
$q^2=10000$	$\underline{K}_f=[16.7 \; - \; 1.9]^T$ $\underline{P}_{x_t}=\begin{bmatrix}285 & -613\\ -613 & 1360\end{bmatrix}$

TABLE D.2

Software Verification Data for Continuous-Time and Discretized Continuous-Time LQG Controllers

Test Case	Controller [a]	Δt [b]	T [c]	q^2 [d]	Steady-state $\underline{P}_{x_t x_t}$ [e]	Steady-state $\underline{P}_{uu}$
1	C^f	-	6	0	$[1.79]$	$[.1045]$
2	C^f	-	1.5	0	$\begin{bmatrix} 0.0000456 & (10)^{-10} \\ (10)^{-10} & 0.00612 \end{bmatrix}$	$[2.25]$
3	C^f	-	1.5	0	$\begin{bmatrix} 221 & -613 \\ -613 & 2068 \end{bmatrix}$	$[4(10)^4]$
3	C^f	-	1.5	100	$\begin{bmatrix} 235 & -613 \\ -613 & 1812 \end{bmatrix}$	$[2.9(10)^4]$
3	C^f	-	1.5	$(10)^4$	$\begin{bmatrix} 316 & -611 \\ -611 & 1192 \end{bmatrix}$	$[7.4(10)^3]$
1	D	3	9	0	$[5.7]$ [e]	$[44.5]$ [e]
1	D	0.1	6	0	$[1.80]$	$[0.11]$
1	D	0.01	6	0	$[1.79]$	$[0.1051]$
1	D	0.001	6	0	$[1.79]$	$[0.1046]$
2	D	0.1	1.5	0	$\begin{bmatrix} 2.2(10)^{16} & -2.4(10)^{17} \\ -2.4(10)^{17} & 2.8(10)^{18} \end{bmatrix}$ [e]	$[4.6(10)^{23}]$ [e]

TABLE D.2 (con't)

Test Case	Controller [a]	Δt [b]	T [c]	q^2 [d]	Steady-state $\underline{P}_{x_t x_t}$ [e]	Steady-state $\underline{P}_{uu}$
2	D	0.01	1.5	0	$\begin{bmatrix} 0.0000475 & -1.6(10)^{-6} \\ -1.6(10)^{-6} & 0.00666 \end{bmatrix}$	[2.95]
2	D	0.001	1.5	0	$\begin{bmatrix} 0.0000458 & (10)^{-10} \\ (10)^{-10} & 0.00616 \end{bmatrix}$	[2.31]
2	D	0.0001	1.5	0	$\begin{bmatrix} 0.00004563 & (10)^{-10} \\ (10)^{-10} & 0.00612 \end{bmatrix}$	[2.26]
3	D	0.25	1.5	0	$\begin{bmatrix} 2.3(10)^{7} & 2(10)^{7} \\ 2(10)^{7} & 1.9(10)^{8} \end{bmatrix}$	$[2.5(10)^{12}]$ [e]
3	D	0.05	1.5	0	$\begin{bmatrix} 223 & -611 \\ -611 & 2605 \end{bmatrix}$	$[1.0(10)^{5}]$
3	D	0.01	1.5	0	$\begin{bmatrix} 220 & -612 \\ -612 & 2126 \end{bmatrix}$	$[4.7(10)^{5}]$
3	D	0.001	1.5	0	$\begin{bmatrix} 221 & -613 \\ -613 & 2073 \end{bmatrix}$	$[4.1(10)^{5}]$
3	D	0.005	1.5	100	$\begin{bmatrix} 235 & -612 \\ -612 & 1826 \end{bmatrix}$	$[3.1(10)^{5}]$
3	D	0.001	1.5	100	$\begin{bmatrix} 235 & -612 \\ -612 & 1814 \end{bmatrix}$	$[2.9(10)^{4}]$

TABLE D.2 (con't)

Test Case	Con- troller [a]	Δt [b]	T [c]	q^2 [d]	Steady-state $\underline{P}_{x_t x_t}$ [e]	Steady-state $\underline{P}_{uu}$
3	D	0.05	1.5	$(10)^4$	$\begin{bmatrix} 2.45(10)^{31} & 1.5(10)^{33} \\ 1.5(10)^{33} & 9.6(10)^{34} \end{bmatrix}$	$[7.4(10)^3]$
3	D	0.0001	1.5	$(10)^4$	$\begin{bmatrix} 316 & -611 \\ -611 & 1192 \end{bmatrix}$ [f]	$[7.4(10)^3]$ [f]

Notes:

a. C= continuous-time LQG controller, D= discretized version of continuous-time LQG controller, S= sampled-data controller

b. Δt= sample time for discrete-time controllers

c. T= total time of simulation $\geq$ twice the time needed to reach steady-state

d. q= scalar design parameter of Doyle and Stein Robustness Enhancement Technique

e. no steady state

f. steady state values agree with those obtained by authors of examples used as test case.

TABLE D.3

Software Verification Data for
Sampled-Data LQG Controllers

Test Case	Δt [a]	T [b]	q^2 [c]	q [d]	Steady-state $\underline{P}_{x_t x_t}$	Steady-state $\underline{P}_{uu}$
1	1	12	0	0	$[1.82]$	$[.094]$
1	.5	12	0	0	$[1.80]$	$[.1018]$
1	.1	6	0	0	$[1.79]$	$[.1045]$
1	.001	6	0	0	$[1.79]$	$[.1046]$
2	.1	1.5	0	0	$\begin{bmatrix} 5.07(10)^{-3} & 1.65 \\ 1.65 & 5.44(10)^{3} \end{bmatrix}$	$[3.5(10)^{6}]$
2	.05	1.5	0	0	$\begin{bmatrix} 2.96(10)^{-5} & -2.32(10)^{-6} \\ -2.32(10)^{-6} & 8.44(10)^{-3} \end{bmatrix}$	$[6.45]$
2	.0001	1.5	0	0	$\begin{bmatrix} 4.56(10)^{-5} & -(10)^{-10} \\ -(10)^{-10} & 6.12(10)^{-3} \end{bmatrix}$	$[2.26]$
3	.25	3	0	0	$\begin{bmatrix} 267 & -594 \\ -594 & 1791 \end{bmatrix}$	$[1.82(10)^{4}]$
3	.1	1.5	0	0	$\begin{bmatrix} 221 & -608 \\ -608 & 2184 \end{bmatrix}$	$[5.92(10)^{4}]$
3	.001	1.5	0	0	$\begin{bmatrix} 220 & -613 \\ -613 & 2071 \end{bmatrix}$	$[4.07(10)^{4}]$

TABLE D.3 (con't)

Test Case	Δt [a]	T [b]	q^2 [c]	q [d]	Steady-state $\underline{P}_{x_t x_t}$	Steady-state $\underline{P}_{uu}$
3	.001	1.5	100	0	$\begin{bmatrix} 235 & -612 \\ -612 & 1813 \end{bmatrix}$	$[2.90(10)^4]$
3	.001	1.5	10000	0	$\begin{bmatrix} 316 & -611 \\ -611 & 1193 \end{bmatrix}$	$[7.74(10)^3]$
1	.01	6	0	.01	[1.65]	[.1002]
1[e]	.01	6	0	1.0	[1.34]	[.1060]
3	.001	6	0	.001	$\begin{bmatrix} 386 & -612 \\ -612 & 974 \end{bmatrix}$	[60.2]
3	.001	6	0	.1	$\begin{bmatrix} 323 & -612 \\ -612 & 1165 \end{bmatrix}$	$[5.8(10)^3]$
3[e]	.001	6	0	1.0	$\begin{bmatrix} 321 & -612 \\ -612 & 1174 \end{bmatrix}$	$[(10)^5]$
2[e]	---	---	---	---	---- ----	----

TABLE D.3 (con't)

Notes:

a. sample time

b. total length of each run

c. q^2 is the Doyle and Stein scalar design parameter. Note that $\underline{V}$, the Doyle and Stein matrix design parameter, is set equal to I for this verification. (see the Doyle and Stein in Discrete-Time Systems - 2 section of this thesis.

d. q is scalar design parameter used in picking the Kalman filter gain directly. Note $\underline{W}$, companion matrix design parameter for q, is always chosen to be $(\underline{H}\ \underline{\Phi}^{-1}\ \underline{B}_d)^{-1}$ in this verification. (see the Enhancing Robustness of Discrete-Time Systems by Directly Choosing K section of this thesis).

e. when picking $\underline{K}$ directly, at some value of q between 1 and 10, the steady-state covariance of the states and controls no longer exists (the exact value of q where this phenomena begins to occur was not determined) for test cases 1 and 2; no value of q allows the covariance of the states and controls to reach steady-state for test case 2.

Appendix E

Users Manual for Linear Quadratic Gaussian Regulator Performance (LQGRP)

The purpose of this manual is to describe how to use LQGRP to design and test Linear Quadratic Gaussian Regulators. The software provides the capability to enhance the robustness of both continuous-time and sampled-data LQG regulators. The techniques used are described in a paper by J.C. Doyle and Stein (Ref 2), in Maybeck section 14.5 (Ref 10), and in the body of this thesis.

Input

The program always begins in the input mode. For the first run through the program, a list of input/output options (1-6) and a list of options to designate which vector/array is to be input or output (1-21), is provided. These options are presented in Table E.1. Note that when entering $\underline{R}_t$ or $\underline{R}_c$ using options 11 and 13 respectively, the value entered may be in either continuous-time or sampled-data format. User response to prompts at other places in LQGRP is used to distinguish between the formats.

During the input mode of the program, the user selects the desired vector/array to be input, then follows prompts about entering I/O options, dimensions and vector/array elements. Note that option 18 allows all controller model matrices to be equated to their truth model counterparts. The number of deterministic states in the system models must also be input at this time (see discussion under LQG Regulator

TABLE F.1

Input Routine Options

Matrix Vector Selection Options	
Option	Function
1	Truth model $\underline{F}_t$ matrix
2	Truth model $\underline{B}_t$ matrix
3	Truth model $\underline{G}_t$ matrix
4	Truth model $\underline{H}_t$ matrix
5	Controller design model $\underline{F}_c$ matrix
6	Controller design model $\underline{B}_c$ matrix
7	Controller design model $\underline{G}_c$ matrix
8	Controller design model $\underline{H}_c$ matrix
9	Truth model initial covariance matrix, $\underline{P}_o$
10	Truth model input noise strength matrix, $\underline{Q}_t$
11	Truth model measurement noise strength matrix, $\underline{R}_t$
12	Controller design model input noise strength matrix, $\underline{Q}_c$
13	Controller design model measurement noise strength matrix, $\underline{R}_c$
14	Truth model initial state vector, $\underline{x}_o$
15	Control cost-weighting matrix, $\underline{W}_{uu}$
16	State cost-weighting matrix, $\underline{W}_{xx}$
17	Cross (state-control) cost-weighting matrix, $\underline{W}_{xu}$
18	Equate $\underline{F}_c$, $\underline{B}_c$, $\underline{G}_c$, $\underline{H}_c$, $\underline{Q}_c$, $\underline{R}_c$ to their truth model counterparts
19	Terminate input mode, start regulator development mode
20	Store all matrices/vectors from options 1-17 to local file Tape7
21	Read all matrices/vectors for options 1-17 from local file Tape8

TABLE E.1 (con't)

Input/Output Options	
Option	Action Taken
0	Terminate main program
1	Read entire array (row by row) or vector
2	Read entire array (row by row) or vector and then print it
3	Read selected array or vector elements
4	Read selected array or vector elements and then print them
5	Print the entire array or vector
6 or greater	Return to calling program without taking any action

Set-Up).

Once a truth model and controller model have been completely entered and the cost-weighting matrices have been entered, option 20 will store all this data to local file Tape7 for future use. Once a file Tape7 has been created (and possibly stored as a permanent file) option 21 can be used to read data from it if it is first copied into a local file Tape8. Note any number of models can be entered during the input mode and stored to Tape7 for future reference.

Note any options 1-18, 20 and 21 can be executed any number of times during the input mode. This allows for changes to be made to specific vector/arrays in case an error is made. Option 19 terminates the input mode.

When the input mode is re-entered at the termination of the performance analysis routine, any, all or none of the options 1-18, 20, 21 may be executed.

Stopping LQGRP

LQGRP may be aborted gracefully while in the input mode. Simply choosing any option 1-17 and entering a zero for the I/C option and (any number for the dimension required, if required) will accomplish the abort.

LQG Regulator Set-Up

Upon leaving the input mode, the program enters the LQG regulator set-up mode. At this point the user must choose either a continuous-time controller or a sampled-data controller. Note, there is provision in LQGRP to allow deterministic

(ie, states which are not controlled by input noise) states to be part of the system description. Note that such states should be the first states in the state vector when entering system model matrices in the input mode. The deterministic states are removed from the system controller design model before Kalman filter gain calculation. The Kalman filter gain corresponding to those states deleted from the Kalman filter gain calculations is set to zero.

Continuous-Time Controller

If the continuous-time controller is chosen, the user has the option of calculating and displaying the eigenvalues of both the truth model and controller model $\underline{F}$ matrices, of modifying the $\underline{G}\ \underline{Q}\ \underline{G}^T$ (used in the matrix Riccati equation for calculating the steady-state $\underline{P}$ matrix), and choosing the total run time and the time increment between integration steps. The modification to $\underline{G}\ \underline{Q}\ \underline{G}^T$ referred to above was derived by Doyle and Stein (Ref 1). Simply, it replaces $\underline{G}\ \underline{Q}\ \underline{G}^T$ by $\underline{Q}(q)$ where

$$\underline{Q}(q) = \underline{G}\ \underline{Q}\ \underline{G}^T + q^2\ \underline{B}\ \underline{V}\ \underline{B}^T$$

q^2 is a scalar design parameter such that $q \to \infty$ causes the LQG regulator to regain asymptotically the robustness characteristics of a full-state optimal deterministic feedback LQ controller. $\underline{V}$ is also a design parameter that must be any symmetric positive definite m x m matrix where m is the number of inputs to the system input matrix $\underline{B}$. Note, $\underline{V} = \underline{I}$ is a good first choice when there is no reason to weight the addition

of pseudo-noise to selected states. The filter poles, LQ controller poles, and the LQG controller poles are computed if the user wants to examine them.

Sampled-Data Controller

For sampled-data controllers, two basic options are available. The first merely discretizes a continuous-time controller. The second developes a sampled-data controller based on the matrices stored during the input mode.

In the first option, a continuous-time LQG regulator is set-up (see previous section) and is then discretized using first order approximations. Note that when entering the time increment for the continuous-time controller to be discretized, the time increment will become the sample-period of the discrete-time controller.

In the second option, an appropriate sampled-data controller is set up using the values entered during the input mode. In this option an approximation to the values of the continuous-time cost-weighting matrices, $\underline{X}(t_i)$, $\underline{U}(t_i)$ and $\underline{S}(t_i)$ for sampled-data controllers is made. A discussion of this approximation is in Appendix C. Briefly, it entails using constant values for certain matrices for each of a number of subintervals of the sample-period. The more subintervals, the better the approximations, but the more computer time required.

Also in the second option, there are two design options for increasing robustness properties of the controller. The first is a discretized version of the Doyle and Stein technique

described previously in the continuous-time section. The second option for increasing robustness involves picking $\underline{K}$, the Kalman filter gain directly such that

$$\underline{K} = q\ \underline{\Phi}^{-1}\ \underline{B}_d\ \underline{W}$$

where $\underline{\Phi}$ is the controller design model state transition matrix, $\underline{B}_d$ is the discrete-time input matrix and $\underline{W}$ is any nonsingular m x m matrix where m is the number of inputs to the system. The user has the option of picking any $\underline{W}$ or of calculating $\underline{W} = (\underline{H}\ \underline{\Phi}^{-1}\ \underline{B}_d)^{-1}$ as in Maybeck (Ref 10). As in the continuous-time case, q is a scalar design parameter that, as $q \to \infty$, asymptotically causes the LQG regulator to recover the robustness characteristics of the full-state feedback optimal deterministic LQ regulator. As in the continuous time case the filter poles, LQ controller poles and the LQG regulator poles are computed if the user wants to examine them.

Performance Analysis

After the LQG regulators are properly formatted by the aforementioned routines, a covariance analysis is performed. This analysis is described in the body of the thesis. First a set of augmented system matrices are formed and then the performance analysis begins. Eqs (E.1) through (E.8) describe the augmented system used in the performance analysis.

$$\dot{\underset{\sim}{x}}_a = \underline{F}_a\ \underset{\sim}{x}_a + \underline{B}_a\ \underline{y}_d + \underline{G}_a\ \underset{\sim}{w}_a \quad ; \quad \underset{\sim}{x}_a^T = [\underset{\sim}{x}_t^T \quad \underset{\sim}{x}_c^T] \tag{E.1}$$

where

$$\underline{F}_a = \begin{matrix} \underline{F}_t + \underline{B}_t\ \underline{G}_{cz}\ \underline{H}_t & \underline{B}_t\ \underline{G}_{cx} \\ \underline{B}_{cz}\ \underline{H}_t & \underline{F}_c \end{matrix} \tag{E.2}$$

$$\underline{B}_a = \begin{bmatrix} \underline{B}_t & \underline{G}_{cy} \\ & \underline{B}_{cy} \end{bmatrix} \tag{E.3}$$

$$\underline{G}_a = \begin{bmatrix} \underline{G}_t & \underline{B}_t \underline{G}_{cz} \\ \underline{0} & \underline{B}_{cz} \end{bmatrix} \tag{E.4}$$

$$\underset{\sim}{w}_a = \begin{bmatrix} \underset{\sim}{w}_t \\ \underset{\sim}{v}_t \end{bmatrix} \tag{E.5}$$

$$\underline{Q}_a = \begin{bmatrix} \underline{Q}_t & \underline{0} \\ \underline{0} & \underline{R}_t \end{bmatrix} \tag{E.6}$$

and then

$$\underline{m}_{x_a}(t) = \underline{\Phi}_a(t, t_o)\, \underline{m}_{x_a}(t_o) + \int_{t_o}^{t} \underline{\Phi}_a(t, \tau)\, \underline{B}_a(\tau)\, \underline{y}_d(\tau)\, d\tau \tag{E.7}$$

$$\underline{P}_{x_a x_a}(t) = \underline{\Phi}_a(t, t_o)\, \underline{P}_{x_a x_a}(t_o)\, \underline{\Phi}_a^T(t, t_o) + \int_{t_o}^{t} \underline{\Phi}_a(t, \tau)\, \underline{G}_a\, \underline{Q}_a\, \underline{G}_a^T\, \underline{\Phi}_a^T(t, \tau)\, d\tau \tag{E.8}$$

Note that in Eqs (E.1) through (E.8) for sampled-data systems $\underline{F}$s are replaced by $\underline{\Phi}$s and other continuous-time matrices are replaced by their equivalent discrete-time counterparts.

The user has the option of printing out the augmented system matrices. The user also has the option of specifying how many time increments there should be between plot points and points printed at the terminal for $\underline{P}_{x_a}$ and $\underline{P}_{uu}$, the covariance of the augmented system and the controller, respectively.

The plot points are formatted and stored to a series of local files as follows:

$\underline{m}_{x_t}$ → Tape 12

$\underline{P}_{x_t x_t}$ → Tape 13

$\underline{m}_u$ → Tape 14

$\underline{P}_{uu}$ → Tape 15

where $\underline{m}_{x_t}$ is the mean of the truth model states, $\underline{P}_{x_t x_t}$ is the covariance of the truth model state (only diagonal entries are written to Tape 13), $\underline{m}_u$ is the mean of the controls generated and $\underline{P}_{uu}$ is the covariance of the controls generated (only diagonal entries are written to Tape 15).

Each performance analysis run is tagged with a run number, total time, sample-time and the dimension of the state vector or control vector as appropriate. This is the first record of information written to the local files, Tape 12 Tape 15 for each run. The last two records written to these files for each run contain the minimum value of the data array and a scale factor; both are needed for plotting.

Upon exit from the performance analysis routine, the program re-enters the input mode and changes the run number.

Plotting

In order to plot the data, it is necessary to abort the LQGRP program and execute the plotting program, MYPLOT on a terminal connected to an HP plotter. If the HP plotter

is unavailable or unacceptable for some reason, MYCLPT will generate a Calcomp plot file at a terminal that can be routed to the Calcomp plotter.

Either MYPLOT or MYCLPT reads data formatted by the performance analysis routine off any local file, Tape xx (where xx is 12, 13, 14, or 15). Note that if the user wishes to do any manipulation of the data tapes before using this program, the result can be stored on any tape numbered 1 through 99. This number can be used in place of 12-15 as indicated earlier. Before plotting it gives the user the option to preview the data. Note, when previewing, the user should check to be sure the last two data entries are the minimum value and the scale factor, respectively.

After each plot, the user has the option to abort, plot a different variable (ie, x_{t_2} instead of x_{t_1}), read and plot the next run of data from the same tape, or read and plot data from a new tape. All options executed in either program are results of user inputs to self-explanatory program prompts printed at the terminal.

A Typical LQG Regulator Performance Analysis Session with LQGRP and MYPLOT or MYCLPT

1. Preliminaries

   A. At the beginning of the program it is necessary to have available for input the matrices listed in Table E.1, options 1 through 17 (Could all be on local file Tape8 saved from last run using option 20). The number of deterministic states in the model is also needed. (if any, they must be the states listed first in the system description).

   B. Attach LQGRP.

   C. Be sure the INPUT and OUTPUT files are connected to the terminal.

   D. In response to COMMAND type LQGRP.

2. Input Mode- Always begins with a printout of the "run number" (got here from 1.D above or 4 below)

   A. Respond to prompts about entering matrices (note the information in Table E.1 is printed at the beginning of the first run of LQGRP)

   B. If desired, use option 20 to save all matrices on local file Tape7.

   C1. Terminate input mode, using option 19

   C2. Terminate program, using any option 1-17 and I/O option 0 (and any dimension if required by prompt, see section 5 below)

3. Regulator Setup Mode (got here from 2.C1 above)

   A. In response to prompt type
      1. C- for continuous-time LQG controller
      2. D- for discretized continuous-time or sampled-data LQG controller.

   B1. (got here from 3.A.1)
      1. Respond to self-explanatory prompts (note time increment required by prompt controls maximum number of points from time 0 to run time and does not affect the accuracy of any results.

   B2. (got here from 3.A.2)
      1. In response to prompt, choose to
         a. Discretize a continuous-time controller or
         b. Choose a sampled-data controller

2a. (got here from 3.B2.1.a)
1. Note that when entering time increment for continuous-time controller it will become the sample-time of the discretized controller and affect stability.
2. Follow the remaining prompts.

2b. (got here from 3.B2.1.b)
1. Answer prompts
2. In response to prompt about subinterval for calculating X, S and U, remember a better approximation is obtained for more subintervals but that this requires more
b computer time.
3. Follow remaining self-explanatory prompts.

4. Performance Analysis Mode (got here from 3.B1 or 3.B2)

A. Be careful in selecting number of prints at terminal (in response to prompts) of $\underline{m}_{x_a}$, $\underline{m}_u$, $\underline{P}_{x_a}$ and $\underline{P}_u$ since the printouts require a significant amount of time and paper.

B. Be careful in choosing the number of points to go on the plot files (in response to prompt) since the number of points significantly affects plotting times. (For example, 100 points plotted on the HP plotter takes about 5 minutes, 200 points approximately 10 minutes etc..... With 200 points to be plotted, there will be more than 20 points plotted per inch since the time axis of the plot is scaled to $7\frac{1}{2}$ inches).

5. Data File Plotting/Storing (got here from 2.C2)

A. If data was saved on Tape7 using option 20, now is the time to create a permanent file of this data (see terminal operations manuals)

B1. If you don't have time to plot the data on local files Tape12, Tape13, Tape14 and Tape 15, now is the time to create permanent files for this data.

B2. Want to plot data from todays run or from previous runs stored on permanent file.
1. Make sure files are rewound before use
2. Attach copy of MYPLOT or MYCLPT depending on where you want the plots made.
a. Type MYPLOT or MYCLPT in response to "COMMAND ".
b. Follow prompts (note Calcomp plotter at AFIT can only handle 5 plots per plot file. It is necessary to terminate MYCLPT (following self-explanatory prompts) after each set of 5 plots is created. Then the plot file should be routed to the plotter,

and then MYCLPT can be re-entered for the next 5 plots).

c. If you are using MYCLPT, the plot file must be routed to the plotter at the termination of MYCLPT.

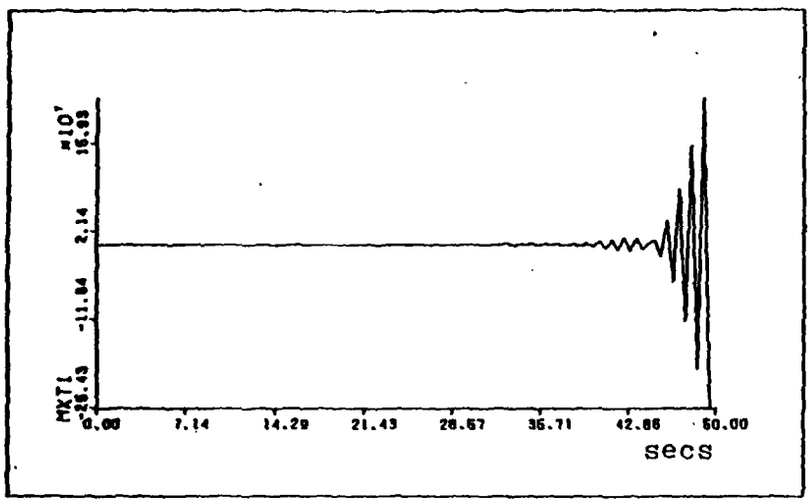
Appendix F

Apollo Model Performance Data

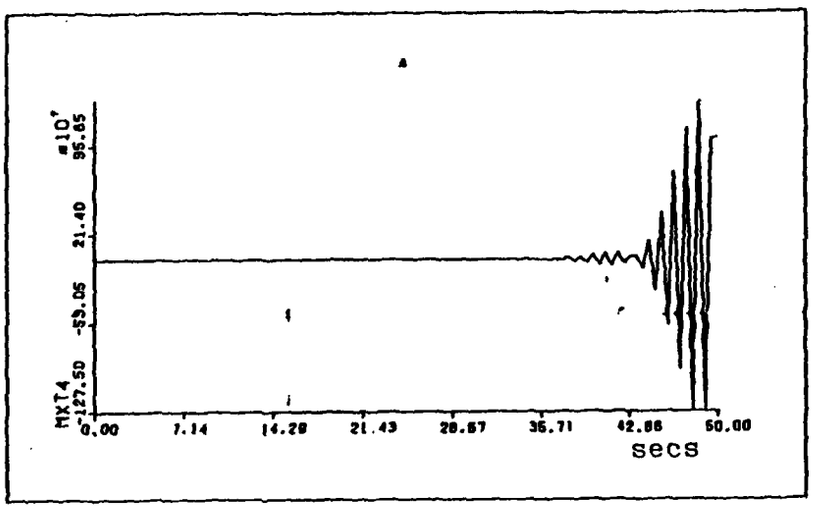
This appendix contains additional mean and covariance plots used to support the results and conclusion of Chapter III.

AD-A115 478 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 1/3
ROBUST CONTROL SYSTEMS.(U)
DEC 81 E D LLOYD
UNCLASSIFIED AFIT/GE/EE/81D-36 NL

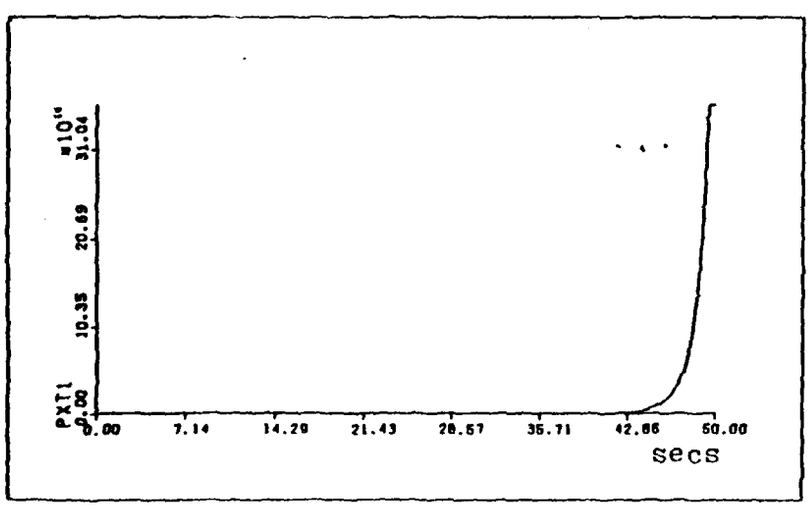
END
DATE FILMED
7 82
DTIC



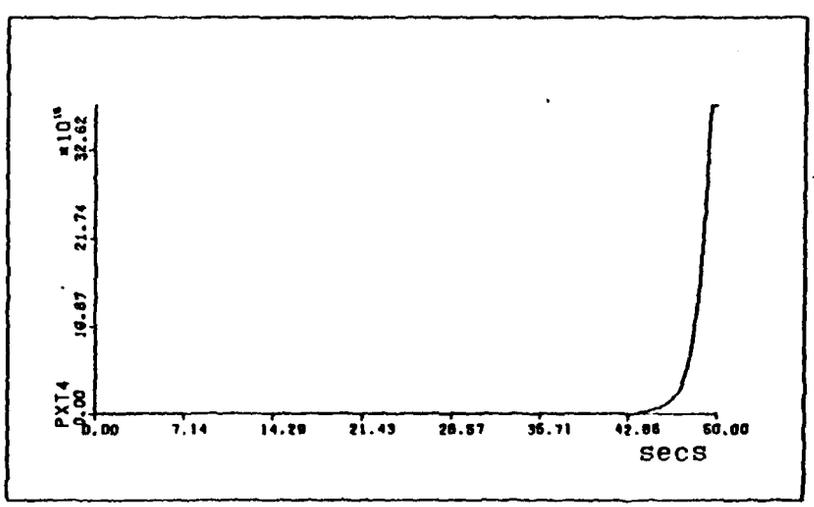
a) mean of state 1



c) mean of state 4

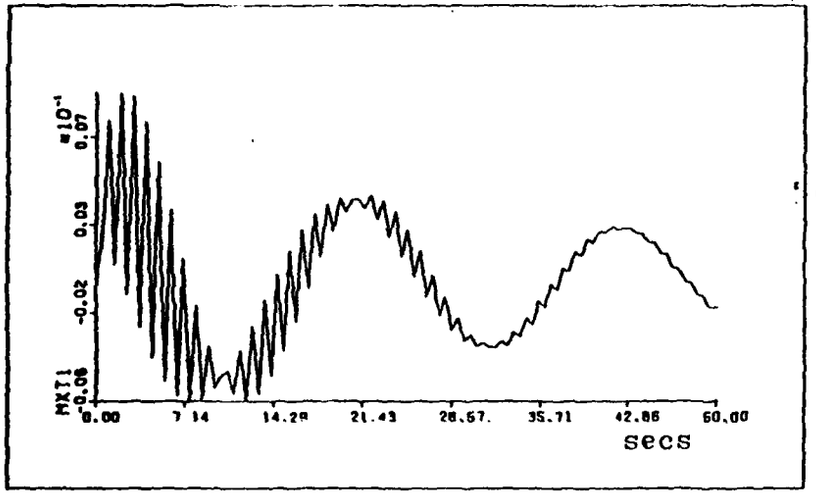


b) variance of state 1

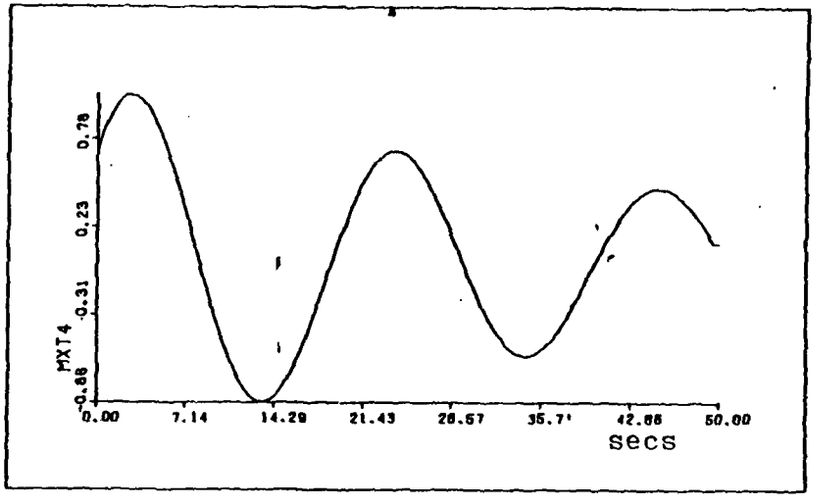


d) variance of state 4

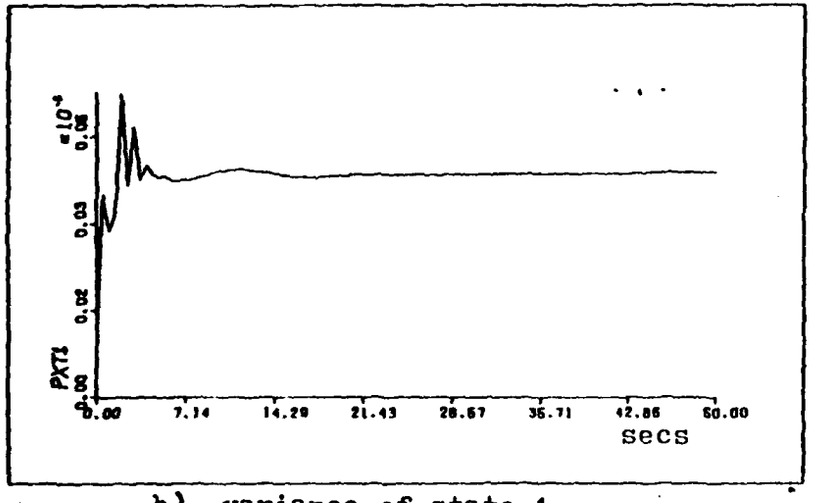
Fig F.1 Continuous-time performance with $\omega_b^2=50$ *in the Apollo model*



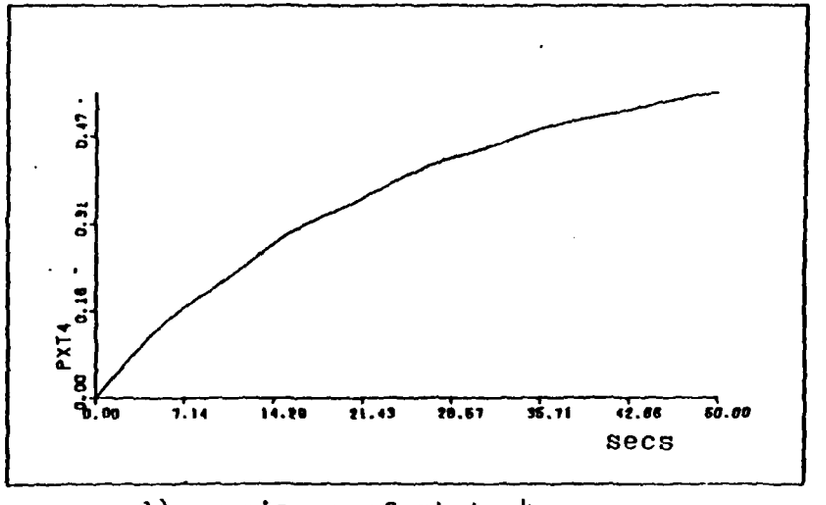
a) mean of state 1



c) mean of state 4

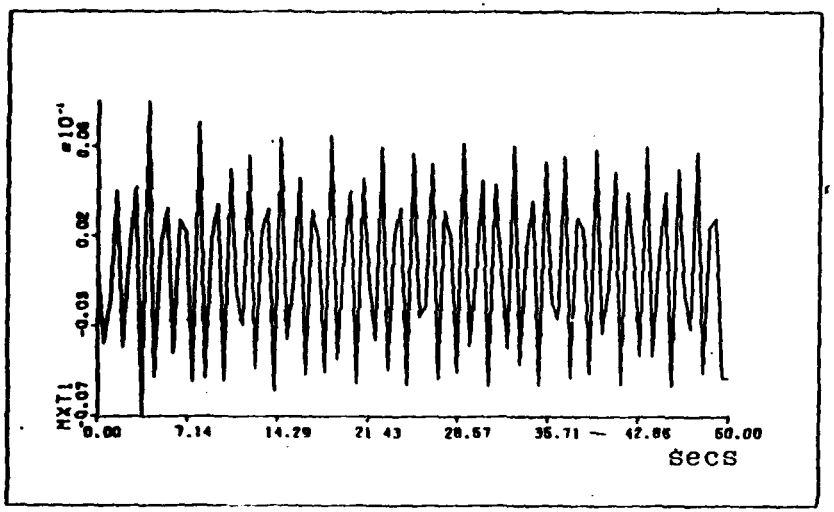


b) variance of state 1

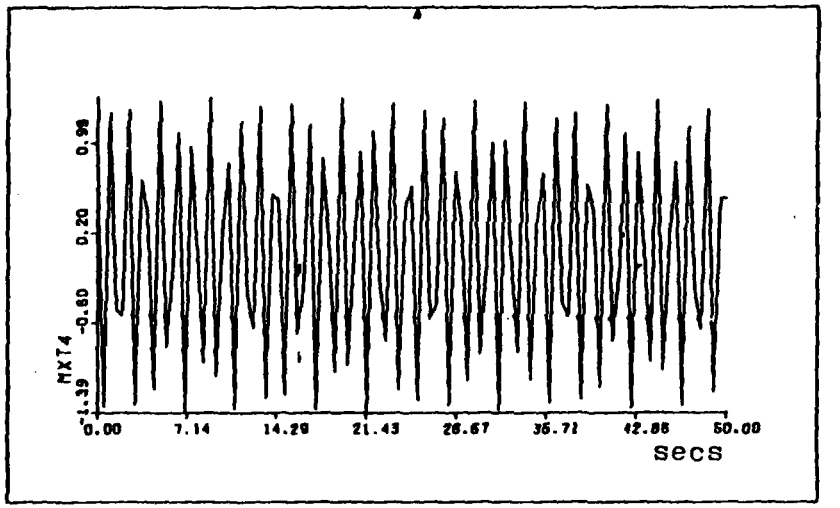


d) variance of state 4

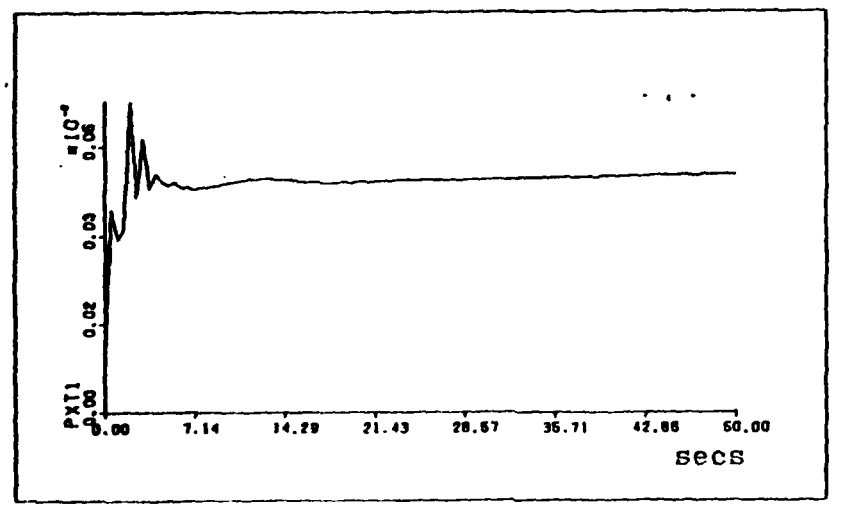
Fig F.2 Continuous-time performance with ω_b^2=150 in the Apollo model



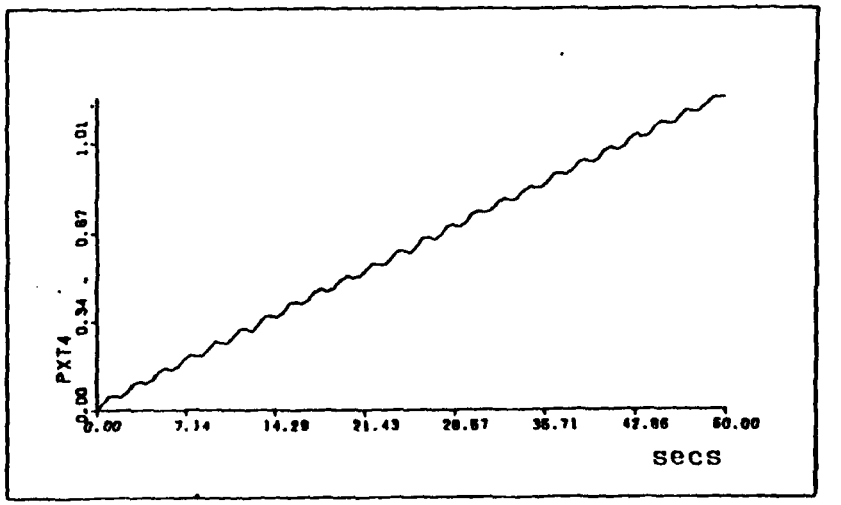
a) mean of state 1



c) mean of state 4

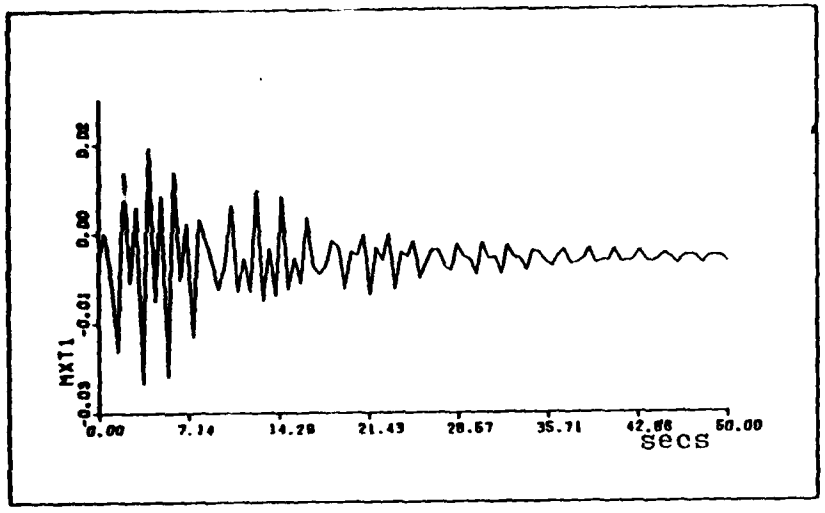


b) variance of state 1

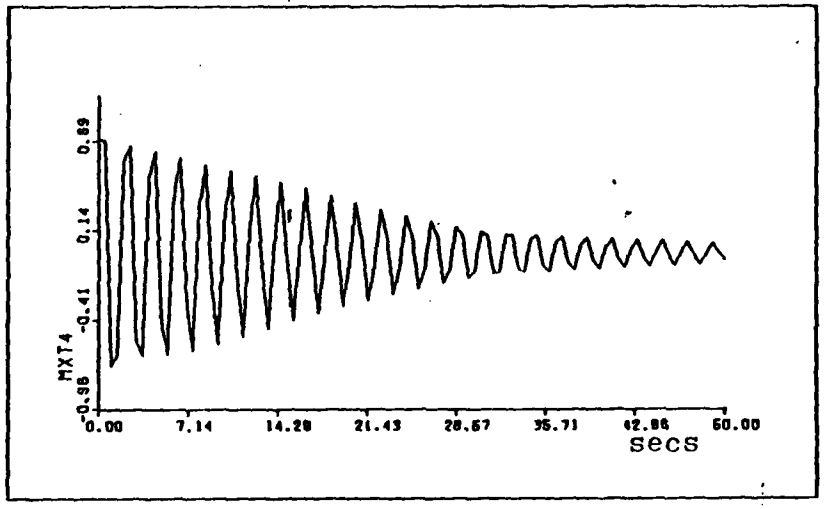


d) variance of state 4

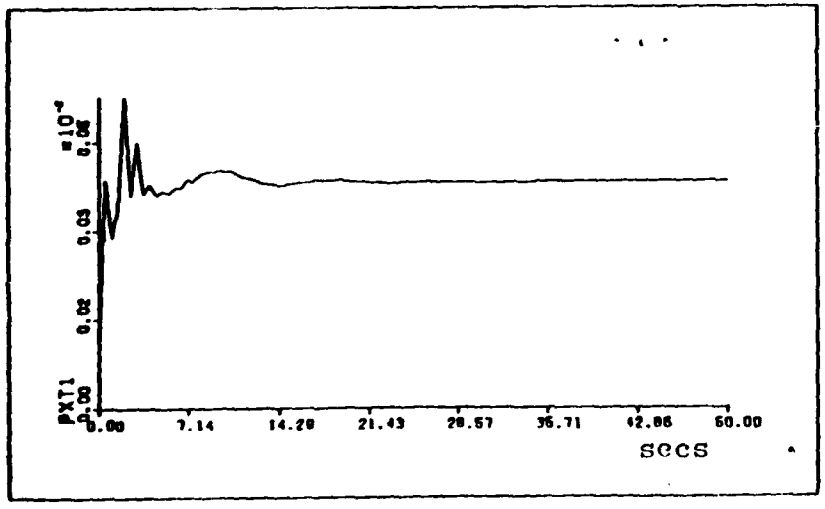
Fig 5.3 Continuous-time performance with ω_h^2=300 in the Apollo model



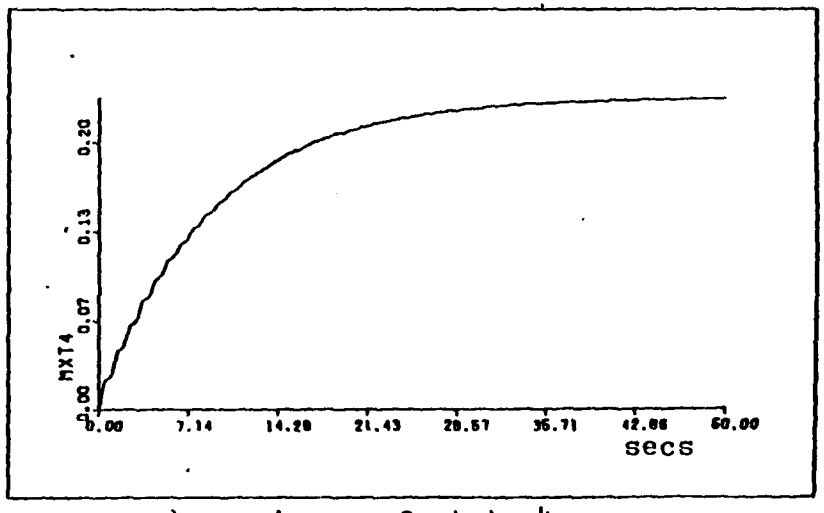
a) mean of state 1



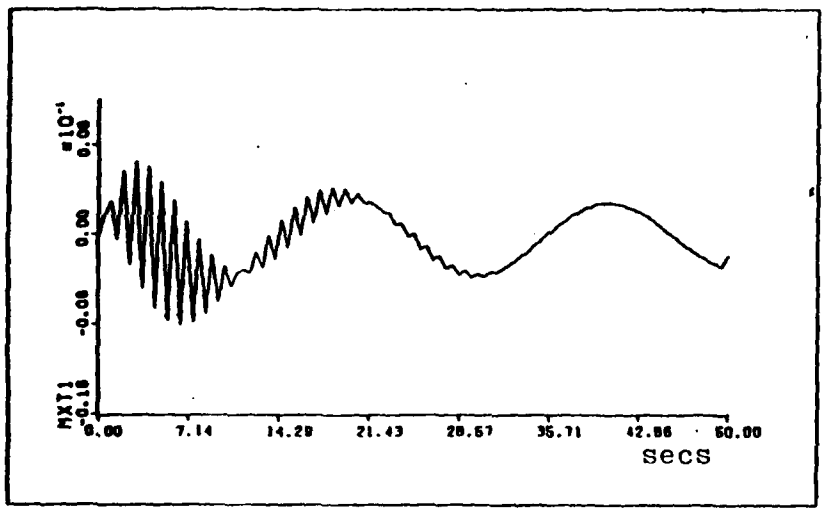
c) mean of state 4



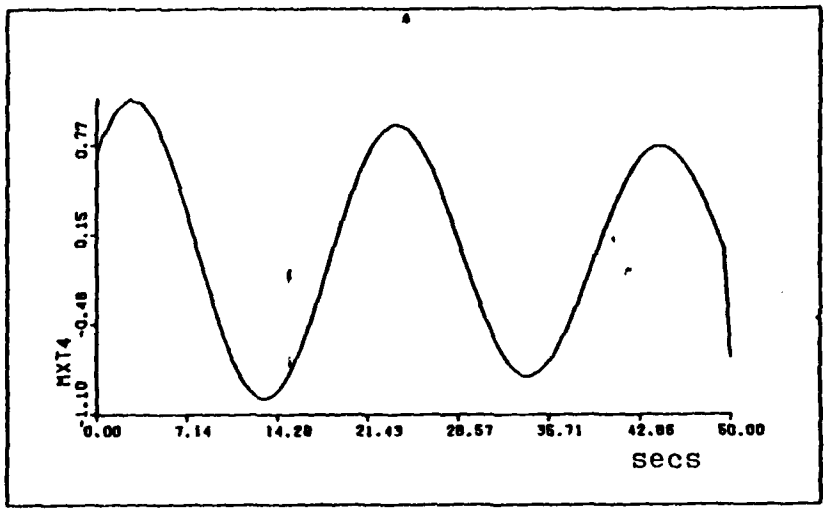
variance of state 1



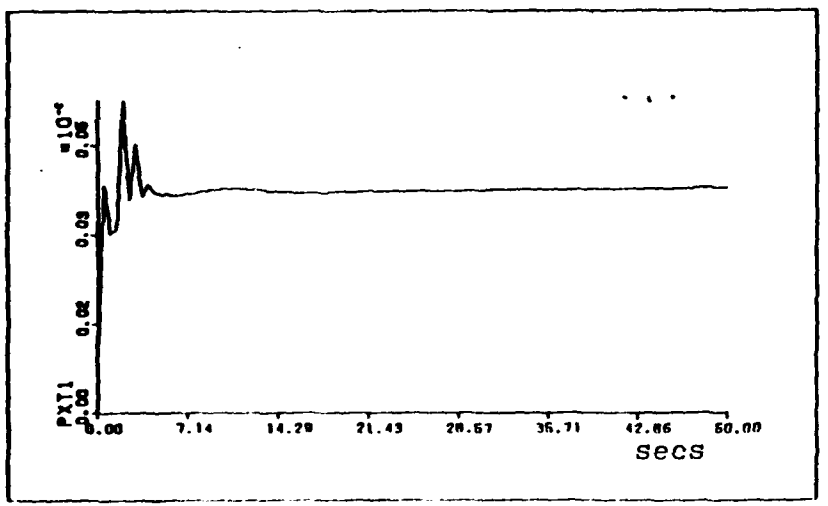
d) variance of state 4



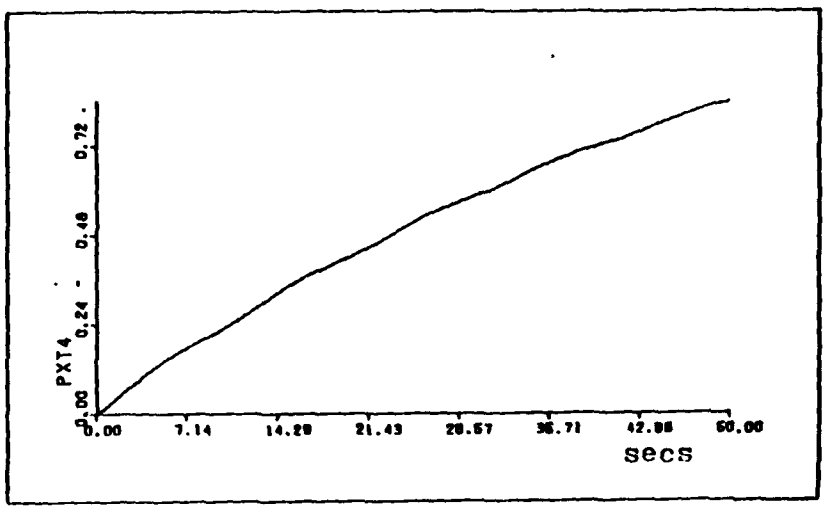
a) mean of state 1



c) mean of state 4

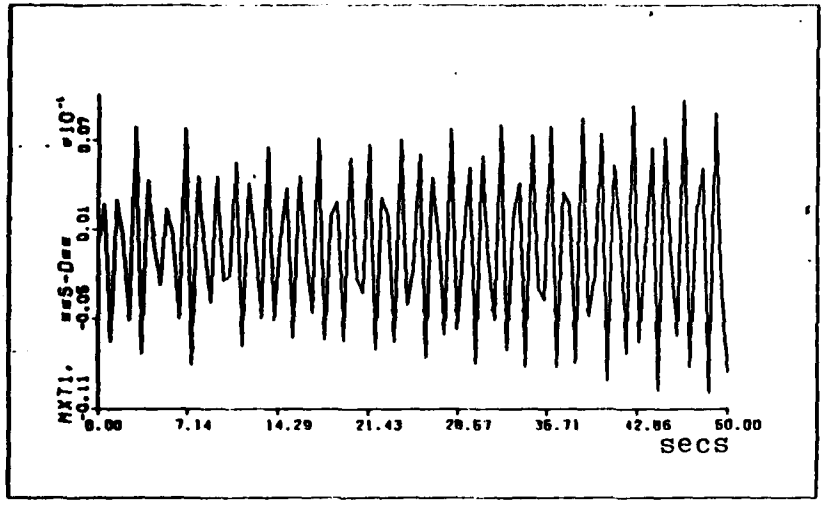


b) variance of state 1

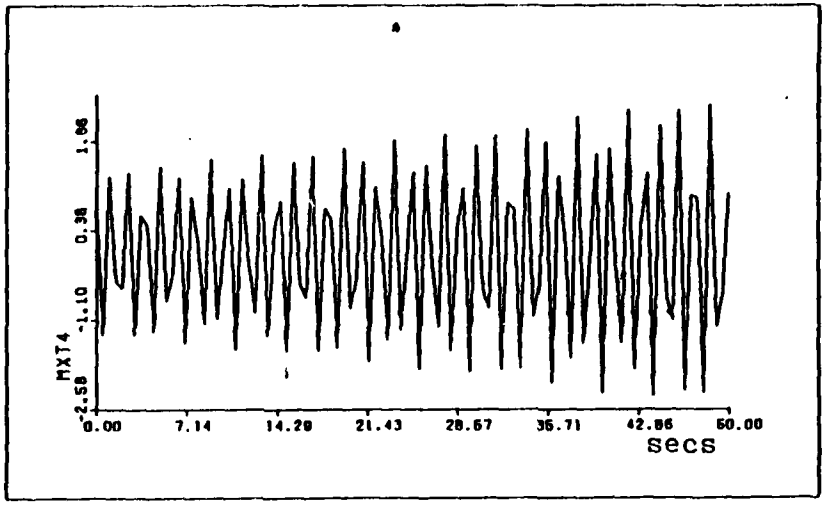


d) variance of state 4

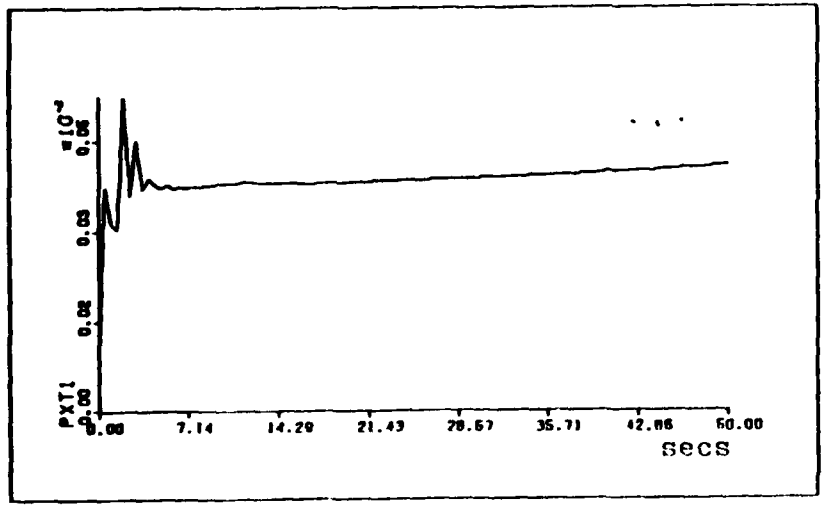
Fig F.5 Sampled-data performance with ω_b^2=150 in the Apollo model



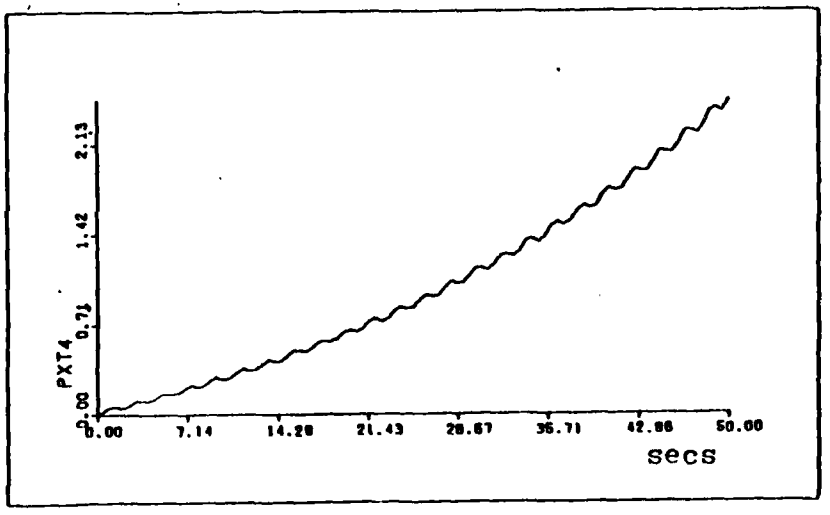
a) mean of state 1



c) mean of state 4

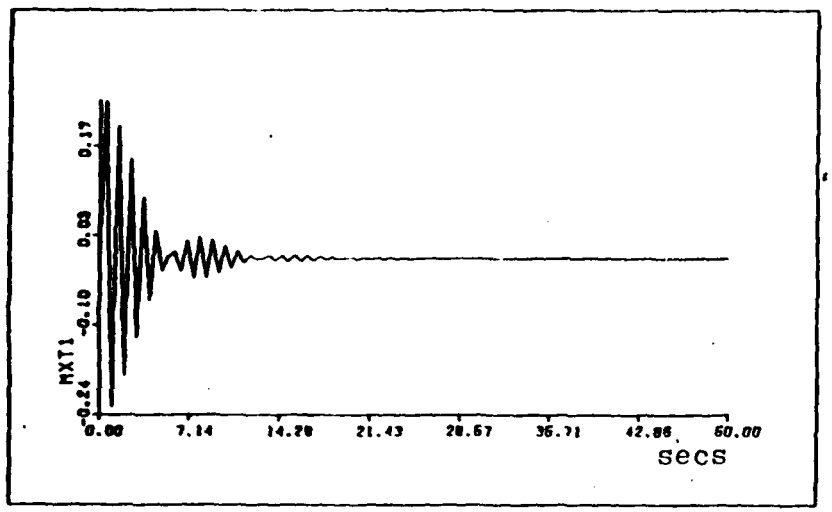


b) variance of state 1

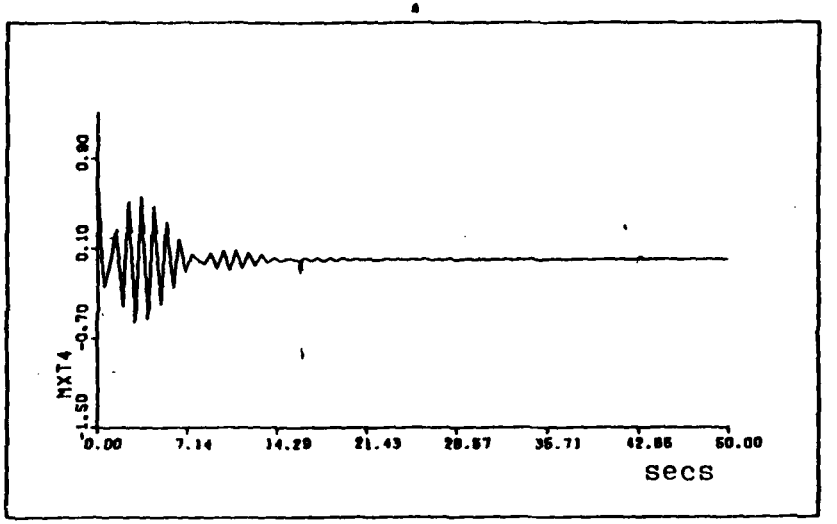


d) variance of state 4

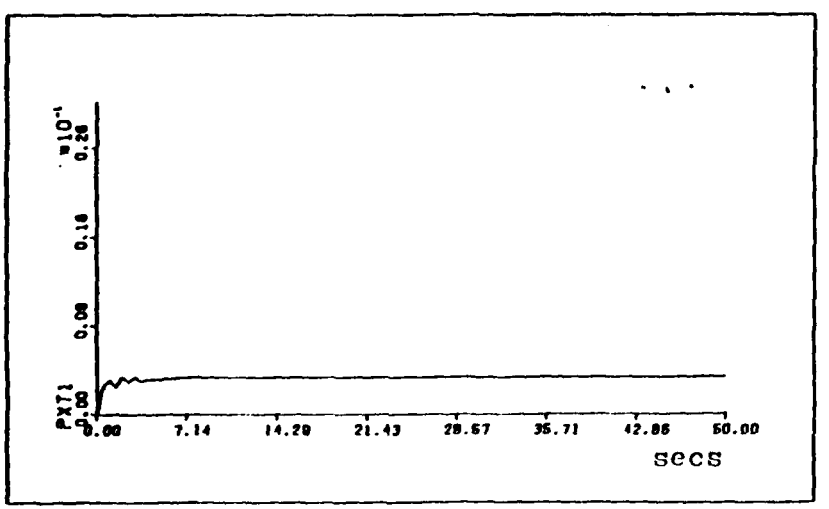
Fig F.6 Sampled-data performance with ω_b^2=300 in the Apollo model



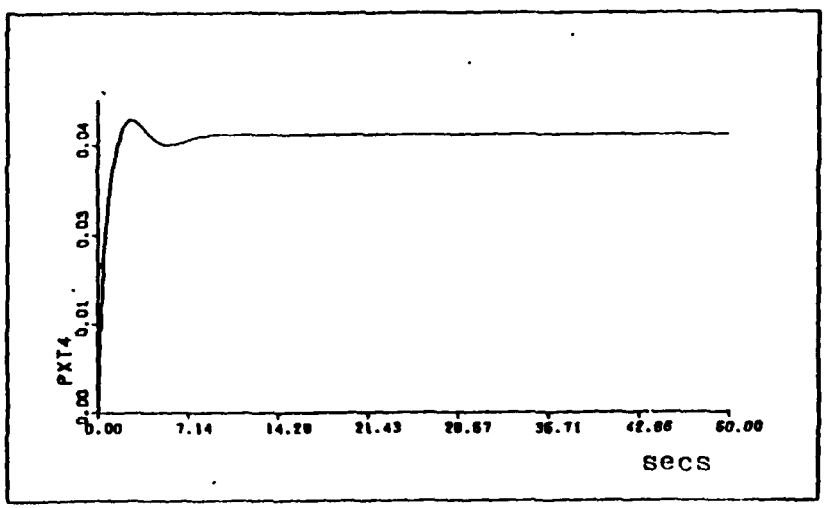
a) mean of state 1



c) mean of state 4

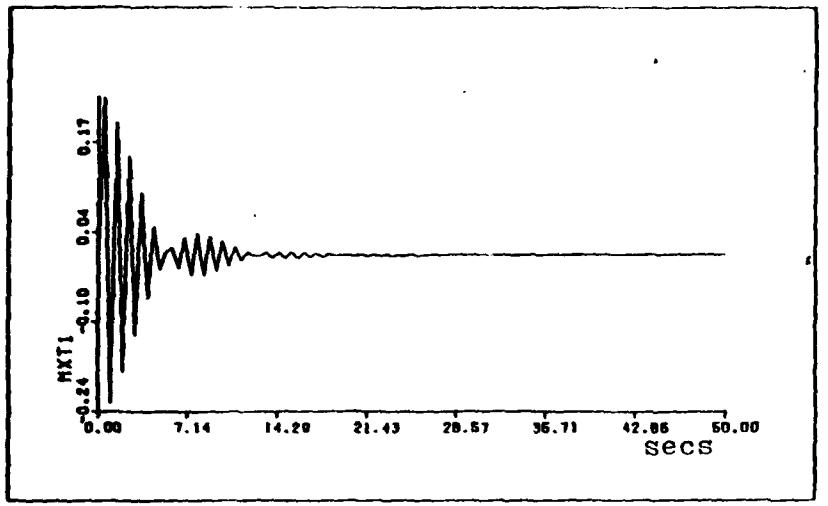


b) variance of state 1

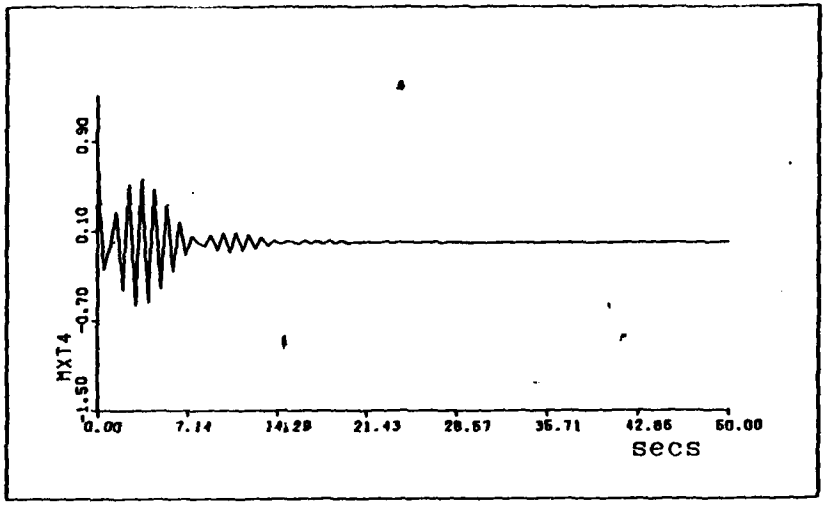


d) variance of state 4

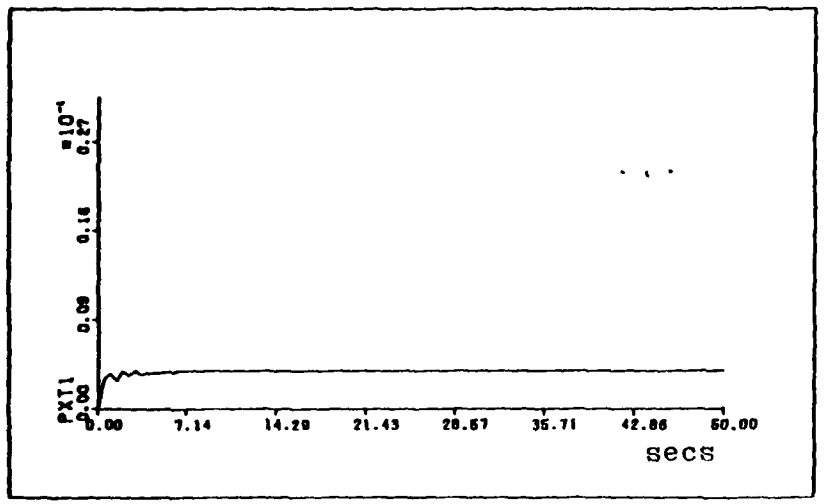
Fig F.7 Sampled-data performance with ω^2=400 and q^2=1



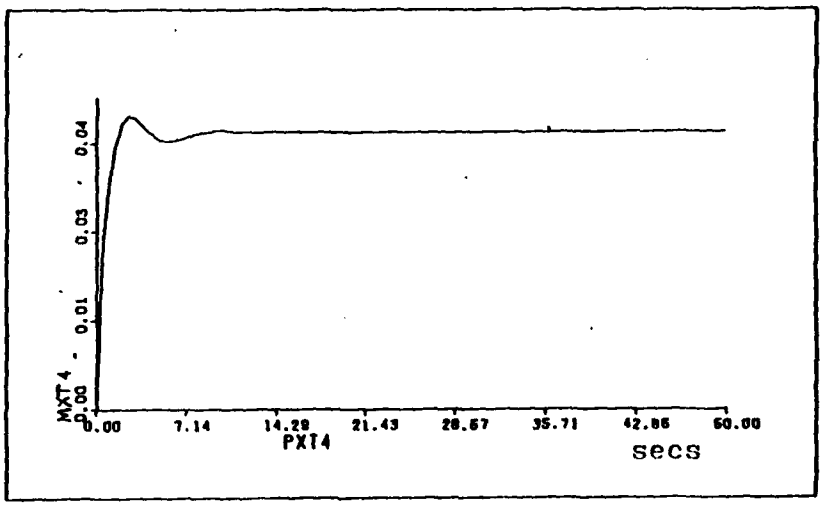
a) mean of state 1



c) mean of state 4

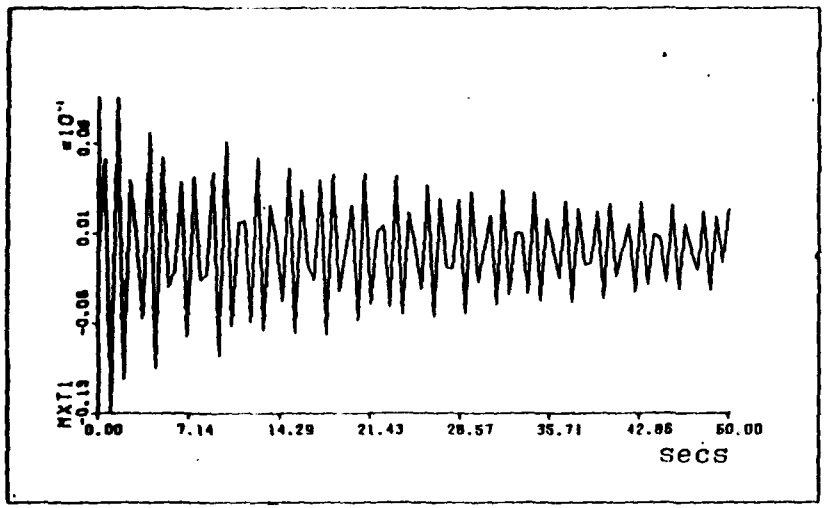


b) variance of state 1

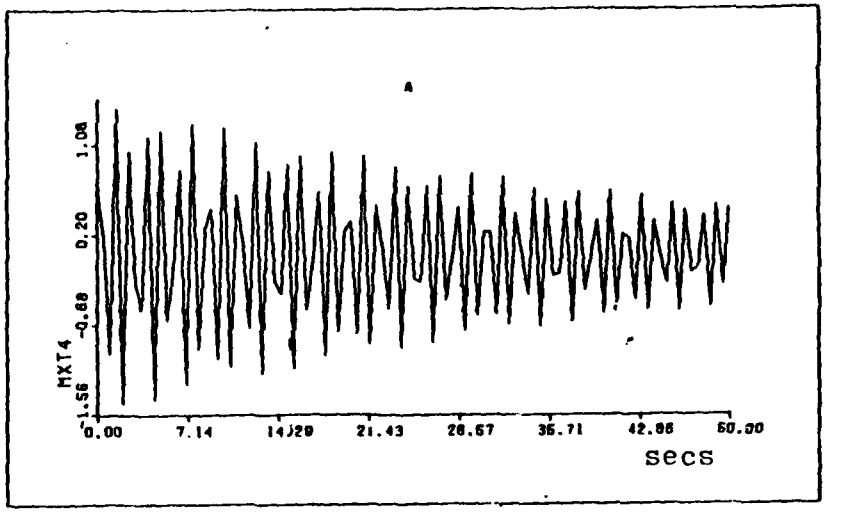


d) variance of state 4

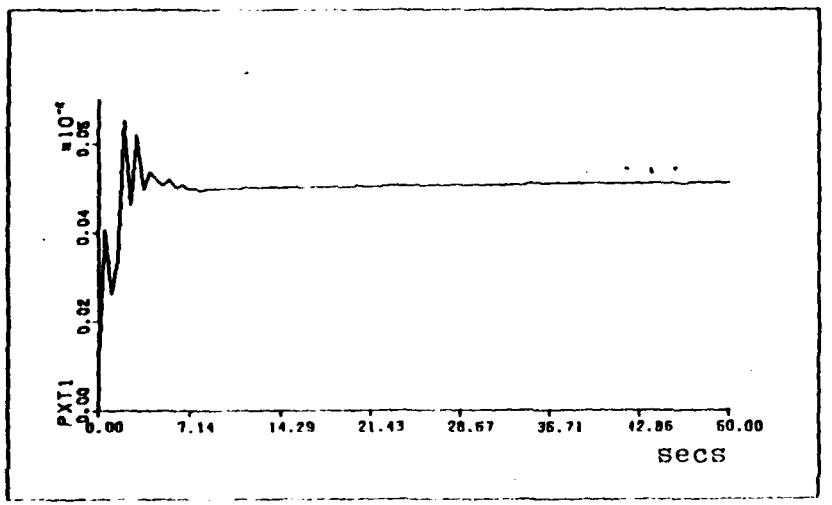
Fig F.8 Sampled-data performance with $\omega_b^2=400$ and $q^2=100$



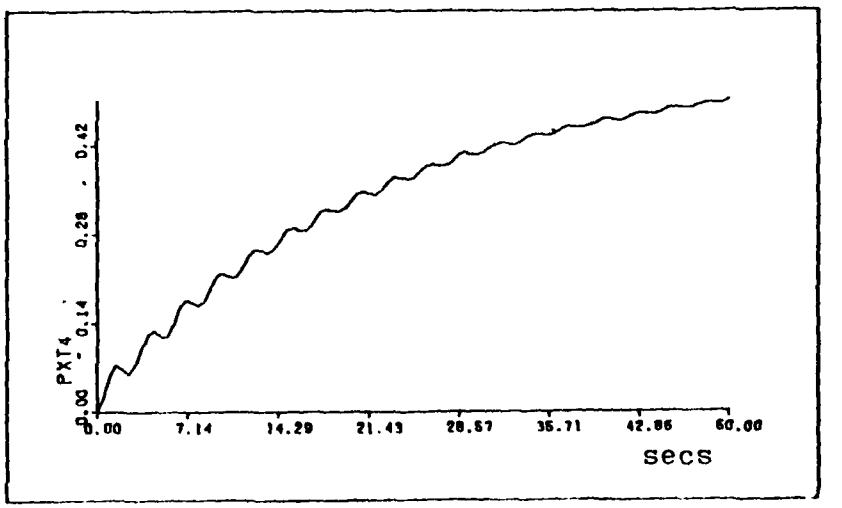
a) mean of state 1



c) mean of state 4

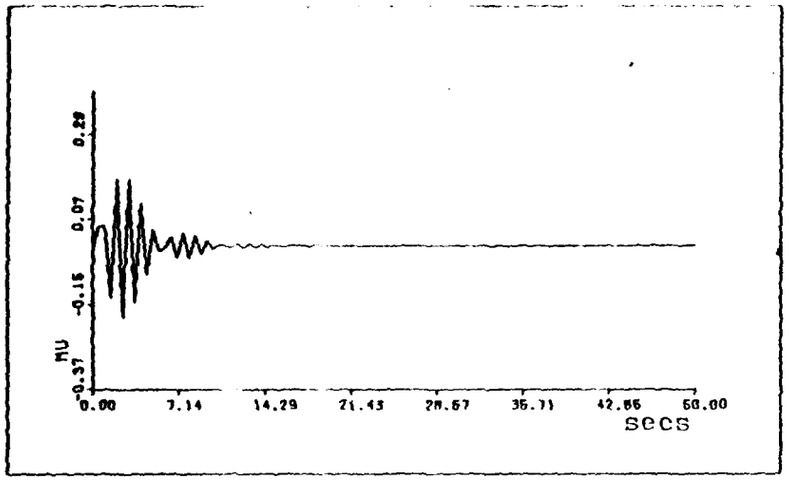


b) variance of state 1

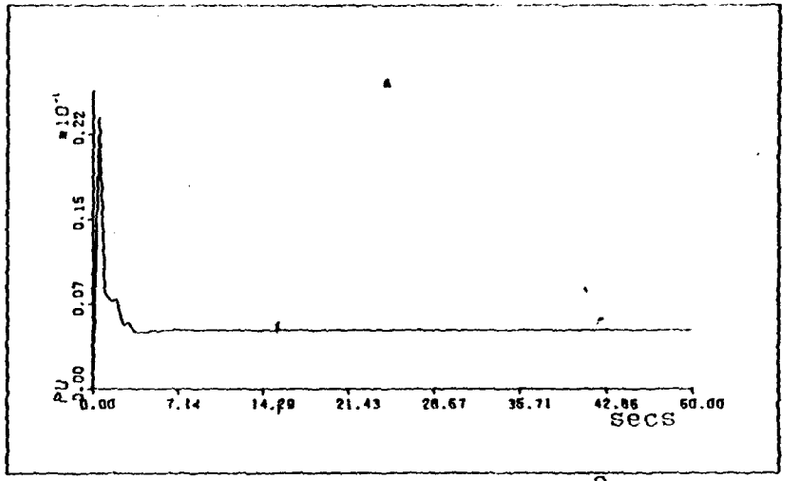


d) variance of state 4

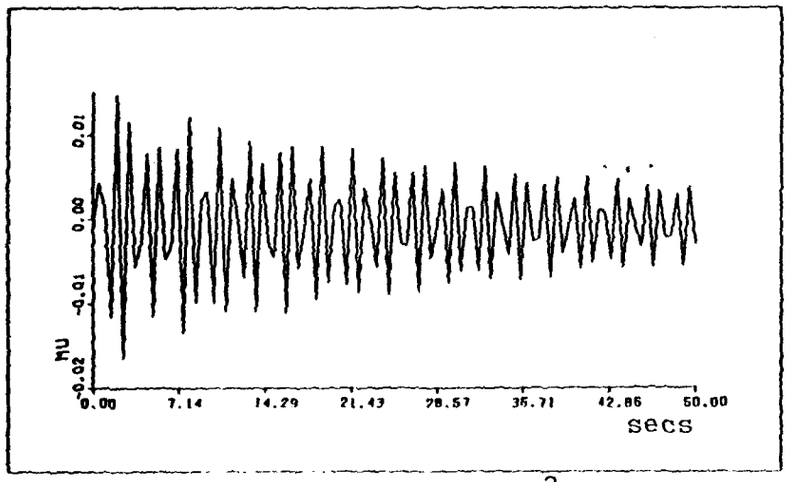
Fig F.9 Sampled-data performance with ω_b^2=400 and q^2=0.0001



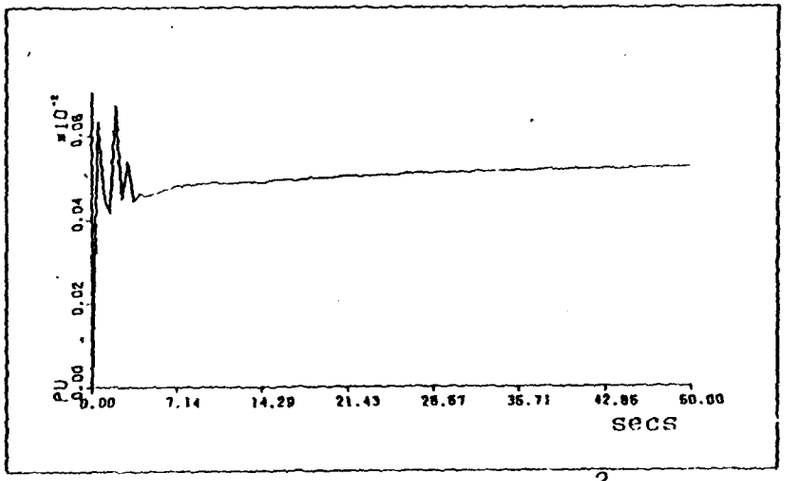
a) *mean* of control, q^2=0.01



b) variance of control, q^2=0.01



c) mean of control, q^2=0.0001



d) variance of control, q^2=0.0001

Fig F.10 Sampled-data performance with ω_b^2=400 and q^2=0.01 and 0.0001 (control)

VITA

Eric Dean Lloyd was born on 26 August 1951 in Key West Florida. He graduated from high school in Glendale, Arizona in 1969 and then attended Glendale Community College. In June 1970 he enlisted in the USAF and was trained as a computer repair specialist. In 1971 he applied for and was accepted in the Airman Education and Commissioning Program. Through this program he earned the degree of Bachelor of Science in Electrical Engineering from the University of Missouri - Columbia, Missouri, in 1974. After his commissioning through the Officers Training School he was assigned as an instrumentation engineer and ground systems test controller for the Minuteman III Weapon System Research and Development testing at Vandenberg AFB, California. Following that, he was assigned in a two year Space Shuttle Test Director training program with the National Aeronautics and Space Administration at the Kennedy Space Center, Florida, until entering the School of Engineering, Air Force Institute of Technology, in June 1980.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/81D-36	2. GOVT ACCESSION NO. AD-A115 478	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ROBUST CONTROL SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED MS THESIS
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Eric D. Lloyd Capt USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE DEC 1981
		13. NUMBER OF PAGES 202
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

15 APR 1982

Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

~~FREDERIC C. LYNCH, Major, USAF~~
~~Director of Public Affairs~~

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Robust Control Systems, LQG Controllers, Kalman Filters, Optimal Stochastic Controllers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Doyle and Stein robustness enhancement technique for continuous-time LQG stochastic controllers was investigated in application to simple examples and a realistic Apollo Command Service Module/Lunar Module Thrust Vector Control System that exhibited severe robustness problems in its initial design. This technique was then extended to discrete-time

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

systems in two ways. First, the continuous-time controller to which the Doyle and Stein technique had been applied was discretized using first order approximations. Second, an approximation to their continuous-time technique was developed for sampled-data control systems. In addition, an attempt was made to enhance the robustness of sampled-data systems by directly picking the gain of the Kalman filter within the controller structure based on an approach similar to that of Doyle and Stein.

Sampled-data controllers were designed using each of these approaches. The resulting performance analysis for each closed-loop system was based on the time histories of the mean and covariance of the "truth model" states and controls as well as on the eigenvalues of the closed-loop system. In both the discretized continuous-time and sampled-data cases, significant steady-state robustness enhancement was observed. Results for picking the Kalman filter gain directly were inconclusive. General purpose interactive software for developing robustified LQG controllers was also produced and documented.

END

DATE
FILMED

7 82

DTIC