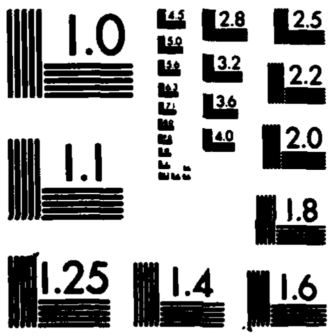
AD-A137 667 OPTIMAL CONTROL AND FILTER GAINS FOR THE STATIONARY CONTINUOUS OR DISCRET..(U) NAVAL RESEARCH LAB WASHINGTON DC C OZIMINA 06 JAN 84 NRL-MR-5239 1/1

UNCLASSIFIED SBI-AD-E000 561 F/G 12/1 NL

END
FILMED
3-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NRL Memorandum Report 5239

2

Optimal Control and Filter Gains for the Stationary Continuous or Discrete Time LQG Problem — A FORTRAN Program

C. D. OZIMINA

Marine Systems Branch
Marine Technology Division

January 6, 1984

NAVAL RESEARCH LABORATORY
Washington, D.C.

84 02 07 122

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER NRL Memorandum Report 5239	2. GOVT ACCESSION NO. AD-A137 667	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) OPTIMAL CONTROL AND FILTER GAINS FOR THE STATIONARY CONTINUOUS OR DISCRETE TIME LQG PROBLEM — A FORTRAN PROGRAM		5. TYPE OF REPORT & PERIOD COVERED Final report covering period Oct. 1982 to Sept. 1983. 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) C. D. Ozimina		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, DC 20375		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N; RR023-01-41; 58-0257-0-0
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE January 6, 1984 13. NUMBER OF PAGES 32
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)

18. SUPPLEMENTARY NOTES

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

Linear quadratic control
Kalman filtering
Steady-state gains
Riccati equation
LQG control gains

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

The eigenvalue-eigenvector method is used to compute the constant linear quadratic control and Kalman filter gains for the stationary continuous and discrete LQG control problem. A listing of a simple Fortran program is provided.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

OPTIMAL CONTROL AND FILTER GAINS FOR THE STATIONARY CONTINUOUS OR DISCRETE TIME LQG PROBLEM - A FORTRAN PROGRAM

INTRODUCTION

In this report the solution to the continuous and discrete-time linear-quadratic regulator (LQR) and Kalman filter (KF) is presented, and a FORTRAN program is included. The LQR or KF design model for a stationary process is described as a continuous or discrete vector-matrix equation. The output is the control system or filter eigenstructure and the optimal steady state LOR or KF gain matrices. The method used to compute the gains is the classical eigenvalue-eigenvector approach.

The common requirement in the design of an optimal LQR control law or a KF is the solution to the control or filter Riccati equation [1]. In the general case, the solution is obtained by a backwards in time propagation from a known te:minal boundary for the LQR, and by a forward in time propagation for the KF.

In most practical design problems, it is assumed that the control or filter design model is time-invariant. With this assertion, the Riccati solution reaches a steady state value in a few sytem time constants. The steady state solution to the Riccati equation is subsequently used to compute the steady state LQR or KF gains, which are used to implement the candidate LQR or KF. In many cases, a KF is used to estimate the state vector in the LQR control law. This latter strategy is referred to as a linear-quadratic-gaussian (LQG) control design [2].

Manuscript approved October 11, 1983.

The soluton to the Riccati equation by integration requires the solution of n_x x $(n_x + 1)/2$ coupled differential equations, where n_x is the dimension of the plant model. When only the steady-state solution is required, an alternative is to solve the algebraic version of the Riccati equation, i.e., the derivative elements are set at zero. This results in a set of simultaneous equations, which must be solved by computer for plant models of greater than 2nd-order.

There are several possible methods of solving the algebraic Riccati equation. The method used in this effort is the classical eigenvalue-eigenvector approach, which was first described by MacFarlane [3] and Potter [4]. The MacFarlane-Potter method, which originally was applied to continuous-time plant models, was extended to discrete-time plant representations by Vaughn [5]. This latter development enables us to use the same basic approach to design a LQR or KF for a discrete or continuous-time design model. Another advantage of this approach is that the eigenstructure of the closed loop LQR or KF is a by-product of the Riccati equation solution. These results are an invaluable aid in evaluating the candidate designs. This is particularly true in the scaler input cases, where the position of closed loop eigenvalues are a direct indication of the control or filter transient response.

The selected method requires the computation of the eigenvalues and eigenvectors of the appropriate Hamiltonian matrix. This difficult computation is facilitated with the use of a few subroutines from EISPACK [6]. EISPACK is a software package which is the product of an intensive effort to develop reliable methods of computing the eigenstructure of various matrix types. The appropriate Hamiltonian matrix is easy to set up from a common set of input matrices for all design model and problem options described.

Notation

The following notation is used throughout this report:

Underlined uncapitalized letters will denote column vectors, with the dimension indicated by, for example, n_x for the vector $\underline{x}$. In the continuous time case, vectors are an implied function of time. In the discrete time case the notation (k), (k + 1), ... will denote the vector at discrete times t_k, t_{k+1}, ..., where k = 0, 1, ... n. T denotes the constant time interval $t_{k+1} - t_k$. The superscript ^{T} denotes the transpose of a matrix or vector, and • and ^ over a vector denotes the time derivative and estimate of, respectively.

A standard set of matrix symbols, which are denoted by capital letters, are used to define the plant models and design parameters for all options. This is done to simplify the input data required. The subscripts $_c$ and $_f$ in the equations indicates whether the particular matrix is associated with a LQR or KF. A ~ over the matrix denotes the discrete-time equivalent of a matrix in the continuous-time model, i.e., $\tilde{A}$.

The symbol E is the expectation operator, (i.e., the average value of), and s is the Laplace transform operator.

PROBLEM DEFINITION AND SOLUTION

A. Continuous-Time LQR Problem

The design model for the plant to be controlled is

$$\dot{\underline{x}} = A\,\underline{x} + B\,\underline{u} \tag{1a}$$

$$\underline{z} = C\,\underline{x} \tag{1b}$$

where $\underline{x}$, $\underline{u}$, and $\underline{z}$ are the state, control input, and output vectors of dimensions n_x, n_u, and n_z x 1, respectively. A, B, and C are constant real plant, control input (distribution), and output matrices, respectively. The control law to be used is of the linear state variable feedback (LSVF) form

$$\underline{u} = G_c \, \underline{x} \tag{1c}$$

where G_c is a constant $n_u \times n_x$ matrix of control gains (to be selected). The LQR design procedure, which results in a control law in the form of (1c) is to be used to select an optimal gain matrix in the linear quadratic (LQ) sense. We note that there are various other design procedures for selecting non-optimal gains, i.e., pole-placement, classical frequency domain methods, etc.

In the LQR design procedure used here, a quadratic cost function of one of the following forms is selected.

(a) state regulator cost

$$2\,J_1 = \int_0^\infty (\underline{x}^T \, Q_c \, \underline{x} + \underline{u}^T \, R_c \, \underline{u})dt \tag{1d}$$

(b) output regulator cost

$$2\,J_2 = \int_0^\infty (\underline{z}^T \, C^T \, Q_c \, C \, \underline{z} + \underline{u}^T \, R_c \, \underline{u})dt \tag{1e}$$

Q_c and R_c in (1d, 1e) are symmetric, real state and control input weighting matrices, respectively, with Q_c restricted to be positive semi-definite (≥ 0) and R_c restricted to be positive definite (> 0). The optimal gain which minimizes J_1 or J_2 is

$$G_c = - R^{-1} \, B^T \, K \tag{1f}$$

where K is a constant matrix (> 0) in the time invariant case. K is obtained by solving the algebraic control Riccati equation

$$A^T K + K A + Q_c' - K B R_c^{-1} B^T K = 0 \tag{1g}$$

where $Q_c' = Q_c$ for the state regulator, and

$Q_c' = C^T Q_c \, C$ for the output regulator

Remark on the constant LQ gain strategy:

An alternate method of computing K is to solve the differential control Riccati equation

$$\dot{K}(t) = A^T K(t) + K(t) \, A + Q_c' - K(t) \, B \, R_c^{-1} \, B^T \, K(t) \tag{1h}$$

backwards in time from $K(\infty) = 0$. $K(t)$ converges to K in a few system time constants. The use of K to compute a constant optimal gain is referred to as the infinite-time LQR in the literature, cf [7, Chap. 9]. If A, B, C, and Q_c and (or) R_c are not constant, the solution of $K(t)$ is indicated. However, the time histories of these matrices are rarely known a-prior. Hence the strategy in the time-varying case is to select a set of stationary plant models corresponding to expected operating points, i.e., various equilibrium conditions. The corresponding set of LQR gains are computed and scheduled as a function of some convenient measured variable, i.e., dynamic pressure in an aircraft. An alternative is to select a constant gain matrix which satisfies the set of plant models (i.e., by simulation studies). See [7, Chap. 9] for an excellent discussion on the impracticalness of implementing $G_c(t)$.

B. Algebraic Riccati Equation Solution

The Hamiltonian matrix associated with the LQR problem [3] is

$$H_c = \begin{bmatrix} -A & S \\ Q_c' & A^T \end{bmatrix}$$

where $S = B\, R_c^{-1}\, B^T$

H_c, which is a $2\, n_x \times 2\, n_x$ matrix, has $2\, n_x$ eigenvalues. For each eigenvalue λ, which appear in conjugate pairs if complex, $-\lambda$ is also an eigenvalue. Those eigenvalues with negative real parts are the closed loop eigenvalues of the LQR, that is, they are poles of the system characteristic equation

$$|sI - (A + B\, G_c)| = 0 \qquad (1j)$$

We note that the LQR design procedure guarantees a stable control law, with certain gain and phase margins, if the plant is controllable [9] [10].

Define the block matrix of eigenvectors associated with H_c

$$W = \begin{bmatrix} W_{11} & W_{21} \\ W_{21} & W_{22} \end{bmatrix} \tag{1k}$$

Where $\begin{matrix} W_{11} \\ W_{21} \end{matrix}$

contains the upper and lower half elements of the eigenvectors associated with the eigenvalues of H_c with <u>positive</u> real parts. The eigenvectors can be arranged in any order, except that those due to complex conjugate eigenvalues are adjacent. The solution to the algebraic Riccati equation is [3, 4]

$$K\, W_{11} = W_{21} \tag{1l}$$

Instead of solving for K by computing the inverse of W_{11}, it is best to manipulate (2d) into the linear system form

$$W_{11}^T\ K = W_{21}^T \tag{1m}$$

K is computed with the use of any a linear system solver. In this effort we use the method described in [11].

Remark on the Hamiltonian:

The Hamiltonian matrix is associated with the Euler-Lagrange system of linear equations [7]

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\underline{p}} \end{bmatrix} = H_c \begin{bmatrix} \underline{x} \\ \underline{p} \end{bmatrix}$$

where $\dot{\underline{p}}$ is the n_x x 1 costate vector. The solution of the above set is a 2-point boundary value problem with $\underline{x}(0)$ and $\underline{p}(\infty)$ known. The solution of the state equation which minimize J_1 or J_2 is

$$\dot{x} = A\underline{x} - B\, R_c^{-1}\, B^T\, \underline{p}$$

where $\underline{p} = K(t)\ \underline{x}$.

C. Discrete-Time LQR Problem

In the discrete-time case, the (normally continuous) plant design model is described in the form

$$\underline{x}(k+1) = \tilde{A}\,\underline{x}(k) + \tilde{B}\,\underline{u}(k) \tag{2a}$$

$$\underline{z}(k) = C\,\underline{x}(k) \tag{2b}$$

where $\tilde{A}$ and $\tilde{B}$ are discrete-time equivalents of A and B in (1a), and $\underline{u}(k)$ is assumed to be piecewise continuous in the constant time interval

$$T = t_{k+1} - t_k, \quad k = 0, 1, \ldots$$

$\tilde{A}$ and $\tilde{B}$ are given by

$$\tilde{A} = \exp(AT) \tag{2c}$$

$$\tilde{B} = \int_0^T \tilde{A}(T, t) B \, dt \tag{2d}$$

Since (1b) and (2b) are algebraic, C does not change.

We have used various methods, which are not presently included in the software described here, to compute $\tilde{A}$ and $\tilde{B}$. The potential methods and possible problems are an interesting subject, c.f. [8]. If T is selected to be sufficiently small, in comparison with the plant time constants, the following first order approximation are useful:

$$\tilde{A} \approx I + AT \tag{2e}$$

$$\tilde{B} \approx BT \tag{2f}$$

The LQR design procedure [12] for the discrete plant is outlined below:

LQ Control Law:

$$\underline{u}(k) = \tilde{G}_c\,\underline{x}(k) \tag{2g}$$

where $\tilde{G}_c$ is a constant LQ gain matrix

Quadratic Cost Functions:

state regulator

$$2\,J_S = \sum_{k=0}^{\infty} \left[\underline{z}^T(k)\ \tilde{Q}_c\ \underline{z}(k) + \underline{u}^T(k)\ \tilde{R}_c\ \underline{u}(k)\right] \tag{2h}$$

output regulator

$$2 J_c = \sum_{k=0}^{\infty} \left[\underline{z}^T(k)\, C^T \tilde{Q}_c\, C\, \underline{z}(k) + \underline{u}^T(k)\, \tilde{R}_c\, \underline{u}(k) \right] \tag{2i}$$

where $\tilde{Q}_c$ and $\tilde{R}_c$ are the discrete time equivalents of Q_c and R_c.

LQR gain:

$$\tilde{G}_c = -\left[\tilde{R}_c + \tilde{B}^T \tilde{K} \tilde{B}^T \right] \tilde{B}^T \tilde{K}\ \tilde{A} \tag{2j}$$

where $\tilde{K}$ is the solution to the discrete algebraic Riccati equation.

Discrete algerbraic Riccati equation:

$$\tilde{A}^T \tilde{K} \tilde{A} - \tilde{A}^T \tilde{K} \tilde{B} \left[\tilde{B}^T \tilde{K} \tilde{B} + \tilde{R}_c \right]^{-1} \tilde{B}^T \tilde{K}\ \tilde{A} + \tilde{Q}_c' - \tilde{K} = 0 \tag{2k}$$

where $\tilde{Q}_c' = \tilde{Q}_c$ (state regulator)

or $C^T \tilde{Q}_c C$ (output regulator)

Hamiltonian:

$$\tilde{H}_c = \begin{bmatrix} \tilde{A}^{-1} & \tilde{A}^{-1} \tilde{S} \\ \tilde{Q}_c' \tilde{A}^{-1} & \tilde{A}^T + \tilde{Q}_c' \tilde{A}^{-1} \tilde{S} \end{bmatrix} \tag{2l}$$

where $\tilde{S} = \tilde{B}\ \tilde{R}^{-1}\ \tilde{B}^T$

$\tilde{H}_c$ has $2\ n_x$ eigenvalues with the following property: For each eigenvalue λ (assumed to be within the unit circle), λ^{-1} is also an eigenvalue (outside the unit circle). Complex eigenvalues appear in conjugate pairs. The eigenvalues within the unit circle of the z-plane are the closed loop poles of the discrete LQR characteristic equation.

Discrete Riccati equation solution:

The method follows that of Vaughn [5].

Let $\begin{bmatrix} \tilde{W}_{11} \\ \tilde{W}_{21} \end{bmatrix}$ be a partitioned matrix of eigenvectors corresponding to those eigenvectors outside the unit circle, i.e., the unstable poles. An eigenvalue λ is outside the unitcircle if λ, or its vector sum, is > 1.0. The organization of the eigenvectors is as in the continuous time case. The solution to (2k) is

$$\tilde{W}_{11}^T \tilde{K} = \tilde{W}_{21}^T$$

Where $\bar{K}$ is solved as in the continuous-time case.

D. <u>Kalman Filter Problem</u>

Continuous Time Filter

In the KF problem the objective is to estimate the state of the linear stochastic plant

$$\dot{\underline{x}} = A\,\underline{x} + B\,\underline{u} + \underline{w} \tag{3a}$$

$$\underline{z} - C\,\underline{x} + \underline{v} \tag{3b}$$

where $\underline{x}$, $\underline{u}$, $\underline{z}$, A, B, and C are as previously defined, and $\underline{w}$ and $\underline{v}$ are independent, zero-mean, gaussian white plant and output (measurement) noise processes, respectively. The intensity matrices (spectral densities) of $\underline{w}$ and $\underline{v}$ are Q_f (≥ 0) and R_f (> 0).

The Kalman filter for this system (c.f. [1] is

$$\dot{\hat{\underline{x}}} = A\,\hat{\underline{x}} + B\,\underline{u} + G_f\,[\underline{z} - C\,\hat{\underline{x}}] \tag{3c}$$

Where $\hat{\underline{x}}$ is the optimal estimate of $\underline{x}$ and G_f is a $n_x \times n_z$ matrix of constant Kalman gains. The state error covariance matrix P is defined as

$$P = E\,\{\delta\underline{x}\,\delta\underline{x}^T\}$$

where $\delta\underline{x} = x - \hat{\underline{x}}$

The initial state error covariance P (t=0) = P_o is assumed to be known.

The Kalman gain matrix is given by

$$G_f = P\,C^T\,R_f^{-1} \tag{3d}$$

Where the symmetric matrix P, > 0, is obtained by solving the algebraic covariance Riccati equation

$$AP + PA^T + Q_f - P\,C^T\,R_f^{-1}\,C\,P = 0 \tag{3e}$$

Note that the dimension of G_c and G_f are different.

The KF can be viewed as a closed loop control sytem. The eigenvalues of the KF are the roots of

$$|sI - A + G_f\ C| = 0$$

Covariance Riccati equation solution:

The Hamiltonian matrix for the KF is

$$H_f = \begin{bmatrix} -A^T & R^{-1} \\ 0_f & A \end{bmatrix} \tag{3f}$$

Where H_f is $2\ n_x \times 2\ n_x$. H_f has the same properties as the control Hamiltonian H_c, Vaughn [5]. Hence the steady state covariance matrix is obtained from

$$W_{11}^T\quad P = W_{21}^T \tag{3g}$$

where the partitioned matrix

$$\begin{bmatrix} W_{11} \\ W_{21} \end{bmatrix}$$

contains the eigenvectors of H_f associated with the positive eigenvalues.

Comment on the time varying Kalman gain:

The time varying state error covariance matrix P(t) is obtained by integrating the differential Riccati equation

$$\dot{P}(t) = A\ P\ (t) + P(t)\ A + Q_f - P(t)\ C^T\ R_f^{-1}\ C\ P(t)$$

forward in time from P_o. Hence it is feasible to compute $G_f(t)$ in real time. The only advantage to this approach in the case of a stationary plant is that $\hat{\underline{x}}$ will converge to $\underline{x}$ more quickly. In the case of a time-varying plant, it is normal to implement the time varying Kalman filter.

Discrete time Kalman filter:

Since the continuous KF equation (3c) contains the deterministic portion of the plant model, it is quite complex to implement with analog circuitry. This indicates the use of a digital computer, and the discrete version of the KF. For this reason, (3c) is seldom used. The discrete version of the continuous stochastic plant model is

$$\underline{x}(k+1) = \tilde{A}\,\underline{x}(k) + \tilde{B}\,\underline{u}(k) + \underline{w}(k) \tag{4a}$$

$$\underline{z}(k) = C\,\underline{x}(k) + \underline{v}(k) \tag{4b}$$

Where $\underline{w}(k)$, $\underline{v}(k)$ are discrete white noise sequences representing plant and measurement noise, and $\tilde{Q}_f$ and $\tilde{R}_f$ are the covariances of $\underline{w}(k)$ and $\underline{v}(k)$, respectively. There are two common forms of the KF in general use. These are (i) the filter form, where

$$\hat{\underline{x}}(k+1) = E\{\underline{x}(k+1) \mid z(o), \ldots, z\,(k+1)\}$$

and (ii) the one step ahead predictor form where

$$\hat{\underline{x}}(k+1) = E\{\underline{x}(k+1) \mid z\,(o), \ldots, z\,(k)\}$$

However both forms are referred to as a "filter" in the literature.

(i) Filter algorithm

The filtered $\underline{x}$ is normally computed in time and measurement update stages, where $\hat{\underline{x}}^-(k)$ and $\hat{\underline{x}}^+(k)$ will denote the time and measurement updated state estimates

Time update

$$\hat{x}^-(k+1) = \tilde{A}\,\hat{\underline{x}}^-(k) + \tilde{\underline{B}}\,u(k) \tag{4c}$$

Measurement update

$$\hat{x}^+(k+1) = \hat{x}^-(k+1) + \tilde{G}_f\left[\underline{z}\,(k+1) - C\,\hat{x}^-(k+1)\right] \tag{4d}$$

where $\tilde{G}_f$ is a matrix of constant filter gains.

(ii) Predictor Algorithm

The one-step ahead predictor algorithm is

$$\hat{\underline{x}}(k+1) = \tilde{A}\,\hat{\underline{x}}(k) + \tilde{B}\,\underline{u}(k) + \tilde{G}_p\left[\underline{z}(k) - C\,\hat{\underline{x}}(k)\right] \tag{4e}$$

where $\tilde{G}_p$ is a constant matrix of predictor gains.

Remarks:

The notation $(k+1|k)$, $(k|k-1)$, etc is often used to denote the one-step ahead predicted estimate of $\underline{x}$.

Optimal Gains

The filter and predictor gains are given by

$$\tilde{G}_f = \tilde{P}\, C^T \left[C\, \tilde{P}\, C^T + \tilde{R}_f \right]^{-1} \tag{4f}$$

and
$$\tilde{G}_p = \tilde{A}\, \tilde{G}_f \tag{4g}$$

Where $\tilde{P}$ is the steady-state, discrete, state error covariance, and is obtained by solving the discrete algebraic covariance Riccati equation

$$\tilde{A}\, \tilde{P}\, \tilde{A}^T + \tilde{Q}_f - \tilde{A}\, \tilde{P}\, C^T \left[C\, \tilde{P}\, C^T + \tilde{R}_f \right]^{-1} C\, \tilde{P}\, \tilde{A}^T - \tilde{P} = 0 \tag{4h}$$

Riccati Equation Solution

The Hamiltonian associated with the discrete KF is

$$\tilde{H}_f = \begin{bmatrix} \tilde{A}^{-T} & \tilde{A}^{-T}\, \tilde{R}_f^{-1} \\ \tilde{Q}_f\, \tilde{A}^{-T} & \tilde{A} + \tilde{Q}_f\, \tilde{A}^{-T}\, \tilde{R}_f^{-1} \end{bmatrix}$$

Where the $2\, n_x \times 2\, n_x$ matrix $\tilde{H}_f$ contains $2\, n_x$ eigenvalues with the same properties as the discrete LQR Hamiltonian $\tilde{H}_c$. The eigenvalues λ of $\tilde{H}_f$ which are inside the unit circle are the closed loop poles of the discrete KF. $\tilde{P}$ is computed using the same method as in the discrete LQR equation (2k), and as first described in [5].

FORTRAN PROGRAM DESCRIPTION

This section describes a FORTRAN progrm which computes the steady state LQR or KF gains, as given in the previous section. A program listing is given in Appendix A. Several separate programs, which have been used by this author over a period of several years, were combined to produce the version described. Hence the result is not as optimal, from a viewpoint of work vectors and matrices, execution time, and modularity, as the program could be if we had started from scratch. Subroutines were used only to eliminate obvious duplications. In some cases, we use a loop instead of an available subroutine, if only 2 or 3 statements are required.

All real FORTRAN variables or constants are in double precision, as defined in the IMPLICIT statement. This allows for easy change from double to single precision computations (with appropriate subroutine changes). In general, matrix and vector names end in M and V, respectively. The exceptions are the real vectors TV1, TV2, and integer vectors IV1, IV2. All integer variables and constants begin with I and loop counters begin with I or J.

The maximum dimension of the state vector is set to 15 in the listed version. This limit can be expanded by changing the integer INXMX to the desired value, and by expanding the matrix and vector dimensions accordingly. All subroutines use variables dimensions.

Data Input

The FORTRAN input file (FT05F001) for each run consists of the following types of data cards:

(a) Control/title card (No. 1)

The integers in the first five columns are used to set the status of the five internal option control flags shown in Table 1. The characters in columns 6 through 65 are used for run identification. The format of this card is (5I1, 15A4). Note that the flag IDATAF is used to terminate the run (= 0), i.e., a blank card.

(b) Plant dimension card (No. 2)

The dimensions of $\underline{x}$, $\underline{u}$, and $\underline{z}$ are input in columns 1-2, 3-4, and 5-6, respectively (FORMAT = 3I2). These integers determine the dimensions of the matrices to follow.

(c)Matrix data cards

Each matrix to be read in is preceded by a card containing a read (1)/no-read (0) matrix flag in column 1. We use columns 6 on to identify the matrix; however, this is not printed out. The coefficients of the matrix to

Table 1 - Control/Title Card Option Flags

Flag	Card Column	Option
IDATAF	1	1 = run, 0 = stop
IOREGF	2	0 = state regulator, 1 = output regulator (LQR option only)
IDISCF	3	0 = continuous plant 1 = discrete plant model
IPRTF	4	0 = normal print 1 = extended print option
ITYPE	5	0 = LQR option 1 = Kalman filter option

be input are read in by rows on cards following the read/no-read flag card. The matrix coefficient card format is 6D12.8. The order and dimensions of the matrices are:

A (n_x x n_x), B (n_x x n_u), C (n_z x n_x), Q (n_x x n_x), and R, which is n_u x n_u for the LQR option, and n_z x n_z for the Kalman filter option. Prior to the first run, A, B, and C are cleared, and Q and R are set to identify matrices. If a matrix is not input, its previous setting does not change. Note that a read/no-read card must be provided for each matrix.

Output Data

The print flag (IPRTF) setting is used to select the normal or extended print-out options. A brief description of the print out data follows:

(i) Normal output (IPRTF = 0)

1. Title field characters and control flag settings (Input card No. 1)
2. Input Data (A, B, C, Q, and R matrices)
3. Eigenvalues of A
4. Q' (= C^T Q C) if IOREGF = 1 (output regulator)
5. Eigenvalues of the Hamiltonian (closed loop eigenvalues)
6. LQR or KF gains

(ii) Extended Output (IPRTF = 1)

Items 1-6 in normal output

7. Hamiltonian matrix
8. Packed eigenvectors of H-matrix from EISPACK subroutine HQR2
9. Normalized eigenvectors
10. Positive eigenvector matrices W_{11}^T, W_{21}^T
11. Riccati equation solution

Computational Details

The A-matrix eigenvalues are computed by calling the EISPACK subroutine BALANC, ELMHES, ELTRAX1, and HOR2. The latter subroutine also computes the eigenvectors, which are not needed. HQR, which computes the eigenvalues only, can be used in place of HQR2 (with the call to ELTRAN deleted).

All eigenvalues and eigenvectors of the Hamiltonian (HAM) are computed by calling the EISPACK subroutines BALANCE, ELMHES, ELTRAN HOR2 and BALANC. However, only the positive (or negative) eigenvalues and eigenvectors of HAM need actually to be computed. An alternate strategy could be used, for example, to reduce storage requirements or execution time (see [6] for details). Note that the subroutine HQR2 will fail for repeated eigenvalues.

The eigenvectors of HAM are normalized for print-out only. This step is not actually required.

The FORTRAN name KM is used to denote the control (K) or filter (P) Riccati equation variable. The linear system to be solved is

W11M x KM = W21M

where W11M = W_{11}^T ,, W21M = W_{21}^T

KM is computed by using the method and subroutines described in [11]. The subroutines used are DECOMP and SOLVE, which were supplied by Dr. L. R. Anderson of Virginia Polytechnic Institute.

These subroutines are variable dimension, double-precision versions of the subroutines DECOMP and SOLVE given in [11, Chap. 17]. Note that the subroutine SING, which is called by DECOMP, is also required (see reference). Note also that one could use the appropriate subroutines from LINPACK [13] to replace DECOMP and SOLVE.

Additional Subroutines Called

The following matrix handling subroutines are used throughout the program and are included in the listing.

PRTMAT - prints out a matrix by rows

READM - reads in a matrix by rows

MULMAT - multiplies two matrices M1 x M2 and stores the result in M3

MATINV - inverts matrix M1 and stores the result in M2. MATINV calls DECOMP and SOLVE

Limitations

If the LQR or KF has repeated eigenvalues, HQR2 will fail. This limitation can be removed by using the method described by Laub [14]. This method, which uses the Schur vector approach, is dependent on the subroutines ORTHES and ORTRAN, which are in EISPACK, and HQR3, which is not (see reference for details).

In the set up of the H-matrix, we assumed that R was diagonal. This assumption is generally true, because it is difficult to determine what the off-diagonal elements of R should be in the LQR option. In the KF option the output noise elements are assumed to be uncorrelated. When the inverse of R is called for, it is computed by inverting the diagonal elements of R. This limitation can be easily removed by using the subroutine MATINV to invert R.

There are no checks for proper dimensions, plant model controllability and observability (required for Riccati equation solution to exist) or for proper matrix characteristics.

CONCLUDING REMARKS

We have described the methods and software for solving the LQR and Kalman filter problems for a stationary continuous or discrete-time process. The methods described have been used for aircraft and submersible control system design. We use a separate, undocumented FORTRAN program to compute the plant dynamics (A) and control input (B) matrices from the vehcle stability derivatives and, mass and geometric properties. The candidate control or filter design is tested with a linear system simulator, or a 6 degree-of-freedom simulator. The latter programs are also undocumented. However, equivalent programs are commonly used.

At present the output from one program is manually manipulated into the input for the next program in the design stage. With the recent acquistion of time-shared operating system based computers, we intend to modify and combine the separate FORTRAN programs above in order to provide an integrated interactive aircraft or submersible control system design package.

REFERENCES

1. Astrom, K. J., "Introduction to Stochastic Control Theory," Academic Press, 1970.

2. Athans, M., "The Role and Use of the Stochastic Linear-Quadratic-Gaussian Problem in Control System Design," IEEE Trans. on Auto. Cont., Vol. AC-16, No. 6, pp. 529-552, Dec. 1971.

3. MacFarlane, A. G. J., "An Eigenvector Solution to the Optimal Linear Regulator Problem," J. of Electronics and Control, Vol. 14, pp. 643-654, June 1963.

4. Potter, J. E., "Matrix Quadratic Solutions," SIAM J. of Appl. Math., Vol 14, pp. 496-501, 1966.

5. Vaughn, D. R., "A Nonrecursive Algebraic Solution for the Discrete Riccati Equation," IEEE Trans. on Auto. Cont., Vol. AC-15, pp. 597-599, 1970.

6. Smith, B. T., et al., "Matrix Eigensystem Routines - EISPACK Guide," Springer-Verlag, 1976.

7. Athans, M. and Falb, P., "Optimal Control," McGraw-Hill, Inc., New York 1966.

8. Moler, C. and Van Loan, C., "Ninteen Dubious Ways to Compute the Exponential of a Matrix," SIAM Review Vol. 20 pp. 801-836, Oct. 1978.

9. Anderson, B. D. O. and Moore, J. B., "Linear Optimal Control," Prentice-Hall, Englewood Cliffs, NJ, 1971.

10. Safanov, M. G. and Athans, M., "Gain and Phase Margins for Multiloop LQG Regulators," IEEE Trans. on Auto. Cont., Vol. AC-22, No. 2, pp. 173-179, April 1977.

11. Forsythe, G. E. and Moler, C. B., "Computer Solution of Linear Algebraic Systems," Prentice-Hall, Englewood Cliffs, NJ, 1967.

12. Dorato, P. and Levis, A., "Optimal Linear Regulators: The Discrete-Time Case," IEEE Trans. on Auto Cont., Vol. AC-16, No. 6, pp. 613-620, Dec. 1971.

13. Dongarra, J. J., et al., "LINPACK Users Guide," SIAM, Philadelphia, 1979.

14. Laub, A. J., "A Schur Method for Solving Algebraic Riccati Equation," IEEE Trans. on Auto. Cont., Vol. AC-24, No. 6, Dec. 1979. p. 913.

APPENDIX A

FORTRAN Program Listing

```
C         LINEAR QUADRATIC GAUSSIAN CONTROLLER DESIGN PROGRAM
C         VERS = 1B,  9/21/82
C         CUT OUT SOME PRINT IN "A" VERS
C         THIS VERS IS A COMBINATION OF SEVERAL SEPARATE VERSIONS
C         AND THEREFOR IS A BIT LENGTHY & MESSY, IE.,COULD BE MODULARIZED
C         WITH SOME ADD'L EFFORT; NO EFFORT WAS MADE TO OPTIMIZE FORT CODE
C         ADDED FILTER & PREDICTOR GAINS FOR DISCRETE KF ON 6/30
C
C         DISCRETE AND CONTINUOUS LQR & KAL FILT COMBINED
C         CHET DZIMINA, D58,B20,767-2171
C         COMPUTES: STEADY STATE GAIN FOR A LINEAR QUADRATIC
C                   REGULATOR OR KALMAN FILTER
C
C****** ITYPE = 0 ( LINEAR QUADRATIC REGULATOR ) ******
C
C ********   CONTINUOUS SYSTEM  *********
C         PLANT DYNAMICS  XD = A * X + B * U
C         OUTPUT PROCESS   Z = C * X
C         COST FUNCTION   J = .5 * INTEGRAL OF ( X(T) * Q * X +
C                              U(T) * R * U ) DT )
C                     Q MUST BE POS SEMI-DEF,SYMMETRIC
C                     R MUST BE POS DEF,DIAGONAL(IN THIS PROG)
C         STATE OR OUTPUT REGULATOR OPTIONS AVAILABLE
C         IOREGF = 0/1 ( STATE/OUTPUT REGULATOR )
C         SEE ATHANS & FALB,CHAP 9 FOR DETAILS
C         NOTATION GENERALLY FOLLOWS THAT OF ATHANS
C         MATRICES IN FORT CODE GENERALLY END IN M,IE.  A = AM,  ETC.
C         VECTORS GEN END IN V
C         SS SOL TO THE RICCATI EQN IS OBTAINED VIA MACFARLANE-
C         POTTER METHOD
C         EISPACK IS USED TO COMPUTE THE REQ'D E-VECTORS
C         THIS VERS USES DP EISPACK
C         DECOMP & SSOLVE ARE USED TO SOLVE LINEAR SYST IN PLACE
C         OF MATRIX INVERSION
C         NOTE: EISPACK SR HQR2 FAILS FOR REPEATED E-VALUES
C **********   DISCRETE VERSION ( IDISF = 1 ) *******
C         PLANT DYNAMICS X(K+1) = A * X(K) + B * U(K)
C         OUTPUT PROCESS   Z(K) = C * X(K)
C         COST FUNCT J = .5 * SUM( X(K)(TRANS) * Q * X(K) +
C                        U(K)(TRANS) * R * U(K) ), K = 0 TO N
C         STATE OR OUTPUT REG OPTION ( IOREGF = 0/1 )
C         NOTE: A,B,Q,R ARE DISCRETE EQUIVALENTS OF CONT PLANT
C         REF: PAPER BY DORATO & LEVIS,IEEE TRANS ON AUTO CNTRL,
C         PP 613-620, DEC. 1971
C******* ITYPE = 1 ( STEADY STATE KALMAN FILTER ) *******
C
C         PLANT & OBSERV MODELS:
C         XD = A * X + W; Z = C * X + V, WHERE W,V ARE
C         INDEPENDENT GAUSIAN WHITE NOIPROCESSES
C        WITH INTENSITY MATRICES OF Q => 0,R > 0 .
C         SS KAL GAIN G = K * C(T) * R(INV), WHERE
C         K = SS SOL TO THE FILTER RICCATI EQN
C         DISCRETE FILTER CASE:
C         X(K+1) = A * X(K) + W(K); Z(K) = C * X(K) + V(K)
C         WHERE A,B,C,Q,R ARE DISCR EQUIVALENTS OF CONT PROCESS
C         SS FILTER GAIN IS: GF(SS)= K(SS)*C(T)*[C*K(SS)*C(T)+R](INV)
C         SS KAL PREDICTOR GAIN IS GP(SS) = A * GF(SS)
C         WHERE K(SS) IS THE SS SOL TO THE DISCR RICCATI EQN
C         NOTE: THE DUALITY THEOREM IS USED TO SOLVE THE FILTER
C         PROBLEM VIA THE SAME METHOD AS THE LQR PROBLEM
C         REFS: ASTROM,K.J."INTRODUCTION TO STOCHASTIC CONTROL
```

```
C          THEORY", OR KALMAN'S ORIG PAPERS, OR THE EXCELLENT
C          TUTORIAL BY I. B. RHODES IN IEEE TRANS ON AUTO CONTR,
C          DEC 1971,PP. 688-706.
C
C********* IPRTF = 1,   ADD'L PRINT OUT FOR DEBUGGING *****
C
      IMPLICIT REAL*8 (A-H,K-Z)
      DIMENSION AM(15,15),BM(15,15),CM(15,15),GM(15,15),RM(15,15)
      DIMENSION HAM(30,30),KM(15,15),GPM(15,15),SM(15,15),TM(15,15)
      DIMENSION AIM(15,15),W11M(15,15),W21M(15,15),WM(30,30),ZM(30,30)
      DIMENSION WRV(30),WIV(30),TV(30),TV1(15),TV2(15)
      INTEGER IV1(30),IV2(30)
      INTEGER INX,INU,INZ,INH,INXMX,INUMX,INZMX,INHMX
        REAL*4 TITLE(15)
C
C
C         CLEAR SOME VARIABLES
C
      ICASE = 0
      IDATAF = 0
      IRAF = 0
      IRBF = 0
      IRCF = 0
      IRGF = 0
      IRRF = 0
      IOREGF = 0
      IPRTF = 0
      IDISF = 0
      ITYPE = 0
C
      INX = 0
      INU = 0
      INZ = 0
      INH = 0
        INR = 0
C
C         SET MAX DIMENSIONS
C
      INXMX = 15
      INUMX = INXMX
      INZMX = INXMX
      INHMX = INXMX * 2
C
C           CLEAR SOME ARRAYS
C          SET GM,RM TO UNITY MATRICES
C
      DO 10 J=1,INXMX
      DO 5 I=1,INXMX
        AM(I,J) = 0.0D0
        BM(I,J) = 0.0D0
        CM(I,J) = 0.0D0
        GM(I,J) = 0.0D0
        GPM(I,J) = 0.0D0
        RM(I,J) = 0.0D0
    5 CONTINUE
        GM(J,J) = 1.0D0
        RM(J,J) = 1.0D0
   10 CONTINUE
C
C         READ ALL INPUT DATA HERE
C
C          IDATAF = READ DATA FLAG(1/0)
C          IOREGF = OUTPUT REGULATOR FLAG, NEED C-MTRX IF ON(1)
C          IDISF = DISCRETE REGULATOR FLAG
C          IPRTF = OPTIONAL PRINT FLAG,PROVIDES MORE DATA FOR
C          DEBUGGING
C          ITYPE = KALMAN FILTER FLG, COMPUTE SS KALMAN GAIN IF
C          ON(1)
C          IRAF = READ A-MTRX, IRBF = READ B-MTRX, ETC.
C          NOTE: 1'ST 2 INPUT CARDS ARE MANDATORY FOR EVERY CASE
C          COMMENTS ON CARD #1 BEGINNING AT COL 6 ON ARE PRINTED
C          INPUT # OF ELEMENTS IN X,U,Z ON CARD #2
C          CARD CONTAINING INPUT/NO INPUT FLAG MUST PRECEDE EACH MATRIX.
```

```
C          WHETHER OR NOT THE MATRIX IS INPUT; ORDER OF MATRIX INPUTS IS:
C          A,B,C,G,R; ANY COMMENTS ON MATRIX INPUT/NO INPUT FLAG CARD
C          FROM COL 6 ON ARE NOT READ IN
CC         MATRIX READ FORMAT IS 6E(D)12.8,DEP ON VERS. IE.,SINGLE OR DP
C          NEW CARD FOR EACH ROW
C
    1 READ(5,901) IDATAF,IOREGF,IDISF,IPRTF,ITYPE,(TITLE(I),I=1,15)
  901 FORMAT(5I1,15A4)
C         TEST FOR NEW CASE
      IF(IDATAF .EQ. 0 ) GO TO 9999
C         READ DIMENSIONS OF AM,BM,CM
      READ(5,702) INX,INU,INZ
  902 FORMAT(5I2)
C         TEST FOR NEW A-MTRX
        READ(5,901) IRAF
      IF(IRAF .EQ. 0 )GO TO 25
      CALL READM(INXMX,INX,INX,AM)
C         TEST FOR NEW B-MTRX
   25 READ(5,901) IRBF
      IF(IRBF .EQ. 0 ) GO TO 35
      CALL READM(INXMX,INX,INU,BM)
C         TEST FOR C-MTRX
   35 READ(5,901) IRCF
      IF(IRCF .EQ. 0 ) GO TO 45
      CALL READM(INZMX,INZ,INX,CM)
C          TEST FOR NEW G-MTRX
   45 READ(5,901) IRGF
      IF(IRGF .EQ. 0 ) GO TO 55
      CALL READM(INXMX,INX,INX,GM)
C         TEST FOR NEW R-MTRX
C          INR IS DIMENSION OF R-MTRX TO BE READ IN
C          SET INR = INU FOR LQR CASE, THEN TEST FOR KAL FILT
   55 INR = INU
        IF( ITYPE .NE. 0 ) INR = INZ
      READ(5,901) IRRF
      IF(IRRF .EQ. 0 ) GO TO 65
      CALL READM(INUMX,INR,INR,RM)
C         DATA IS READ IN
   65 ICASE = ICASE + 1
C
C         NOW PRINT INPUT DATA(OR DEFAULT DATA)
C
      WRITE(6,715) ICASE
  915 FORMAT('1',10X,'INPUT DATA FOR CASE NO. ',I4)
        WRITE(6,717) (TITLE(I),I=1,15)
  917 FORMAT('0',5X,15A4)
      WRITE(6,720)
  920 FORMAT('0',5X,'FLAGS(1=YES,0=NO)')
      WRITE(6,925) IRAF,IRBF,IRCF,IRGF,IRRF
  925 FORMAT(' NEW A=',I2,5X,'NEW B=',I2,5X,'NEW C=',I2,5X,'NEW G=',I2,
     1  5X,'NEW R=',I2)
      WRITE(6,930) IOREGF,IDISF,IPRTF,ITYPE
  930 FORMAT(' OUTPUT REG FLG=',I2,5X,'DISCRETE REG FLG=',I2,
     X  /,' OPTIONAL PRINT FLG =',I2,5X,'KALMAN FILTER FLG =',I2)
      IF( IRAF. EQ. 0 ) GO TO 67
      WRITE(6,933)INX,INX
  933 FORMAT('0',5X,'A-MTRX   ',I2,' BY ',I2)
      CALL PRTMAT(INXMX,INX,INX,AM)
   67 IF( IRBF .EQ. 0 ) GO TO 69
      WRITE(6,935) INX,INU
  935 FORMAT('0',5X,'B-MTRX   ',I2,' BY ',I2)
      CALL PRTMAT(INXMX,INX,INU,BM)
   69 IF( IRCF .EQ. 0 ) GO TO 71
      WRITE(6,938) INZ,INX
  938 FORMAT('0',5X,'C-MTRX   ',I2,' BY ',I2)
      CALL PRTMAT(INZMX,INZ,INX,CM)
   71 IF( IRGF .EQ. 0 ) GO TO 73
      WRITE(6,740) INX,INX
  940 FORMAT('0',5X,'G-MTRX   ',I2,'  BY  ',I2)
      CALL PRTMAT(INXMX,INX,INX,GM)
   73 IF( IRRF . EQ. 0 ) GO TO 75
      WRITE(6,942) INU,INU
```

```
  942 FORMAT('0',5X,'R-MTRX   ',I2,'  BY   ',I2)
      CALL PRTMAT(INUMX,INR,INR,RM)
      WRITE(6,945)
  945 FORMAT(' CAUTION!!! R-MTRX MUST BE DIACONAL DUE TO METHOD OF',
     1  ' INVERTING!!!')
C
C******* COMPUTE E-VAL'S OF A-MTRX ***********
C         SET TM = AM TO PRESERVE A-MTRX
C         USE EISPACK TO COMPUTE E-VAL'S OF TM
C         NOTE: WE ARE USING HQR2 INSTEAD OF HQR FOR CONVENIENCE
   75   DO 85 I=1,INX
        DO 85 J=1,INX
   85 TM(I,J) = AM(I,J)
        CALL BALANC(INXMX,INX,TM,ILOW,IHI,TV)
        CALL ELMHES(INXMX,INX,ILOW,IHI,TM,IV1)
        CALL ELTRAN(INXMX,INX,ILOW,IHI,TM,IV1,SM)
        CALL HQR2(INXMX,INX,ILOW,IHI,TM,WRV,WIV,SM,IERR)
        IF( IERR .EQ. 0 ) GO TO 90
        WRITE(6,948) IERR
  948 FORMAT('0',5X,'EIGENVALUE COMPUTATION FAILURE, IERR =',I2)
        GO TO 100
   90 WRITE(6,949)
  949 FORMAT('0',5X,'THE EIGENVALUES OF THE A-MTRX ARE:')
        DO 100 I=1,INX
          WRITE(6,960) WRV(I),WIV(I)
  100 CONTINUE
C
C******** BEGIN COMPUTATIONS TO SET UP HAMILTONIAN **********
C
C         COMPUTE GPM; STATE REG: GPM = G
C                      OUTPUT REG: GPM = C(T) * G * C
C                      KAL FILTER: GPM = G
C
C         TEST FOR KAL FILT
        IF( ITYPE .NE. 0 ) GO TO 130
C        TEST IOREGF
      IF(IOREGF .NE. 0 ) GO TO 150
C        SET GPM = GM FOR KAL FILT OR  STATE REGULATOR
  130 DO 140 J=1,INX
      DO 140 I=1,INX
        GPM(I,J) = GM(I,J)
  140 CONTINUE
      GO TO 180
  150 CONTINUE
C        COMPUTE GPM = CM(T) * GM * CM
      DO 160 I=1,INX
      DO 160 J=1,INX
      WM(I,J) = 0.0D0
      DO 160 IK =1,INX
      WM(I,J) = WM(I,J) + GM(I,IK) * CM(IK,J)
  160 CONTINUE
C        NOW COMPUTE GPM = CM(T) * WM
      DO 170 I=1,INX
      DO 170 J=1,INX
      GPM(I,J) = 0.0D0
      DO 170 IK=1,INX
        GPM(I,J) = GPM(I,J) + CM(IK,I) * WM(IK,J)
  170 CONTINUE
C
C         SET DIMENSION OF HAMILTONIAN
C
  180 INH = INX * 2
C
C         TEST FOR KALMAN FILTER
        IF( ITYPE .NE. 0 ) GO TO 110
C         COMPUTE SM = BM * RM(INV) * BM(T)
C        FIRST, COMPUTE TM = RM(INV) * BM(T), SAVE FOR USE LATER
      DO 175 I=1,INU
      DO 175 J=1,INX
        TM(I,J) =  BM(J,I) / RM(I,I)
  175 CONTINUE
C         NEXT COMPUTE SM = BM * TM
        CALL MULMAT(INXMX,INX,INU,BM,INXMX,INX,TM,INXMX,SM)
        GO TO 135
```

```
C
C          SM = CM(T) * RM * CM FOR KALMAN FILTER
C
  110  DO 120 I=1,INZ
        DO 120 J=1,INX
          TM(J,I) = CM(I,J) / RM(I,I)
  120 CONTINUE
C         COMPUTE SM = TM * CM
        CALL MULMAT(INXMX,INX,INZ,TM,INXMX,INX,CM,INXMX,SM)
C
C         TEST FOR KALMAN FILTER
  135 IF( ITYPE .NE. 0 ) GO TO 460
C********* REGULATOR PROBLEM *********
C         TEST FOR CONTINUOUS/DISCRETE LQR
C
      IF( IDISF .NE. 0 ) GO TO 400
C
C ********    CONTINUOUS PLANT LQR  *******
C          FORM THE CONTINUOUS PLANT HAMILTONIAN
C
C        HAM =  -AM    ,  SM
C               GPM   ,  AM(T)
C        WHERE SM =  BM * RM(INV) * BM(T)
C              GPM = GM OR CM(T) * GM * CM, DEPENDING ON IOREGF
C
C         MOVE  -AM INTO UPPER LEFT BLOCK OF HAM,
C              GPM INTO LOWER LEFT BLOCK,
C              SM INTO UPPER RT BLOCK
C              AM(T) INTO LOWER RIGHT BLOCK
C
      DO 190 I=1,INX
        I1=I + INX
      DO 190 J=1,INX
        J1 = J + INX
        HAM(I,J) = -AM(I,J)
        HAM(I,J1) = SM(I,J)
        HAM(I1,J) = GPM(I,J)
        HAM(I1,J1) = AM(J,I)
  190 CONTINUE
        GO TO 450
C
C******* FORM DISCRETE PLANT HAMILTONIAN *********
C
C         HAM = AM(INV)             AM(INV) * SM
C               GPM * A(INV)        A(T) + GPM * AM(INV) * SM
C
C
C               WHERE SM = BM * RM(INV) * BM(T)
  400 CONTINUE
C         COMPUTE AM(INV) BY LU DECOMPOSITION
C         SEE FURTHER DOWN IN LISTING FOR REF & DETAILS
C
        CALL MATINV(INXMX,INXMX,INXMX,INXMX,INX,AM,KM,W11M,
     X  W21M,IV1,IV1)
C         AM(INV) = KM
C         COMPUTE AM(INV) * SM = W11M
          CALL MULMAT(INXMX,INX,INX,KM,INXMX,INX,SM,INXMX,W11M)
C         COMPUTE GPM * AM(INV) * SM = W21M
        CALL MULMAT(INXMX,INX,INX,GPM,INXMX,INX,W11M,INXMX,W21M)
C         COMPUTE AM(T) + GPM * AM(INV) * SM,
C         STORE IT IN LOWER RT BLOCK OF HAM
      DO 420 I=1,INX
          I1 = INX + I
        DO 420 J=1,INX
          J1 = INX + J
          HAM(I,J) = KM(I,J)
          HAM(I,J1) = W11M(I,J)
          HAM(I1,J1) = AM(J,I) + W21M(I,J)
  420 CONTINUE
C         COMPUTE GPM * AM(INV)
        CALL MULMAT(INXMX,INX,INX,GPM,INXMX,INX,KM,INXMX,W21M)
C         STORE IN LOWER LEFT BLK OF HAM
        DO 430 I=1,INX
          I1 = INX + I
```

```
        DO 430 J=1,INX
          HAM(I1,J) = W21M(I,J)
  430   CONTINUE
C         PRINT OUT GPM IF IOREGF IS SET
  450 IF(IOREGF .EQ. 0 ) GO TO 500
      IF(IRGF.AND.IRRF.EQ. 0) GO TO 500
      WRITE(6,946) INX,INX
  946 FORMAT('0',5X,'C(T) * G * C MATRIX   ',I2,'  BY   ',I2)
      CALL PRTMAT(INXMX,INX,INX,GPM)
        GO TO 500
C*****  KALMAN FILTER HAMILTONIAN *******
C
C         TEST FOR DISCRETE KAL FILT
  460 IF( IDISF .NE. 0 ) GO TO 480
C ********    CONTINUOUS PLANT KAL FILT   *******
C         FORM THE CONTINUOUS PLANT HAMILTONIAN
C
C        HAM = -AM(T),   SM
C               GPM  ,   AM
C        WHERE SM = CM(T) * RM(INV) * CM
C
C         MOVE -AM(T) INTO UPPER LEFT BLOCK OF HAM,
C              GPM INTO LOWER LEFT BLOCK,
C              SM INTO UPPER RT BLOCK
C              AM INTO LOWER RIGHT BLOCK
C
      DO 470 I=1,INX
        I1=I + INX
      DO 470 J=1,INX
        J1 = J + INX
        HAM(I,J) = -AM(J,I)
        HAM(I,J1) = SM(I,J)
        HAM(I1,J) = GPM(I,J)
        HAM(I1,J1) = AM(I,J)
  470 CONTINUE
        GO TO 500
C******* FORM DISCRETE PLANT HAMILTONIAN *********
C
C         HAM = AM(INV-T)             AM(INV-T) * SM
C               GPM * A(INV-T)        A + GPM * AM(INV-T) * SM
C
C               WHERE SM = BM * RM(INV) * BM(T)
C
C         INVERT AM BY LU DECOMPOSITION
  480 CALL MATINV(INXMX,INXMX,INXMX,INXMX,INX,AM,AIM,W11M,
     Y W21M,IV1,IV1)
C       AM(INV) = AIM
C         FIRST, TRANSPOSE AIM
        DO 485 I=2,INX
          II=I-1
        DO 485 J=1,II
        DUM = AIM(I,J)
      AIM(I,J) = AIM(J,I)
  485 AIM(J,I) = DUM
C         COMPUTE GPM * AM(INV-T)
        CALL MULMAT(INXMX,INX,INX,GPM,INXMX,INX,AIM,INXMX,KM)
C         COMPUTE AM(INV-T) * SM
          CALL MULMAT(INXMX,INX,INX,AIM,INXMX,INX,SM,INXMX,W11M)
C         COMPUTE GPM * AM(INV-T) * SM = W21M
        CALL MULMAT(INXMX,INX,INX,GPM,INXMX,INX,W11M,INXMX,W21M)
C         COMPUTE AM + GPM * AM(INV-T) * SM,
C         STORE IT IN LOWER RT BLOCK OF HAM
C         STORE REMAINING ELEMENTS IN HAM IN SAME LOOP
      DO 490 I=1,INX
          I1 = INX + I
        DO 490 J=1,INX
          J1 = INX + J
          HAM(I,J) = AIM(I,J)
          HAM(I,J1) = W11M(I,J)
          HAM(I1,J) = KM(I,J)
          HAM(I1,J1) = AM(I,J) + W21M(I,J)
  490 CONTINUE
C         OPTIONAL PRINT OUT OF HAMILTONIAN MATRIX
C
```

```
  500 IF(IPRIF .EQ. 0 ) GO TO 195
      WRITE(6,947) INH,INH
  947 FORMAT('0',5X,'HAMILTONIAN MATRIX  ',I2,'  BY  ',I2)
      CALL PRIMAT(INHMX,INH,INH,HAM)
  195 CONTINUE
C        NOW COMPUTE THE E-VALUES AND E-VECTORS OF HAM
C        VIA EISPACK SUBROUTINES
C
      CALL BALANC(INHMX,INH, HAM,ILOW,IHI, TV)
      CALL ELMHES(INHMX,INH,ILOW,IHI, HAM,IV1)
      CALL ELTRAN(INHMX,INH,ILOW,IHI, HAM,IV1, ZM)
      CALL HQR2(INHMX,INH,ILOW,IHI, HAM, WRV, WIV, ZM,IERR)
      IF(IERR .EQ. 0 ) GO TO 200
      WRITE(6,950) IERR
  950 FORMAT('0',5X,'EIGENVALUE COMPUTATION FAILURE,IERR = ',I2)
      GO TO 1
  200 CALL BALBAK(INHMX,INH,ILOW,IHI, TV,INH, ZM)
C
C        PRINT E-VALUES
      WRITE(6,955)
  955 FORMAT('0',5X,'THE EIGENVALUES OF THE HAMILTONIAN ARE:')
      DO 210 I=1,INH
C        PRINT NEG E-VAL'S ONLY
        IF( WRV(I) .GT. 0.0 ) GO TO 210
        WRITE(6,960)WRV(I),WIV(I)
  960 FORMAT(' ',1PD15.7,10X,1PD15.7)
  210 CONTINUE
C         OPTIONAL PRINT OF  PACKED E-VECTORS FROM EISPACK
      IF(IPRIF .EQ. 0 ) GO TO 212
C        PRINT THE PACKED E-VECTORS OF HAM
      WRITE(6,962)
  962 FORMAT('0',5X,'THE E-VECTORS OF HAM ARE:')
      CALL PRIMAT(INHMX,INH,INH,ZM)
  212 CONTINUE
C
C         NORMALIZE THE E-VECTORS
C
      IK = 0
  214 IK = IK + 1
      IF( WIV(IK) .NE. 0.0D0 ) GO TO 220
      SUM = 0.0D0
      DO 215 I=1,INH
      SUM = SUM + ZM(I,IK) * ZM(I,IK)
  215 CONTINUE
      SUM = SQRT( SUM )
      DO 218 I=1,INH
      ZM(I,IK) = ZM(I,IK) / SUM
  218 CONTINUE
      GO TO 230
C
  220 SUM = 0.0D0
      DO 225 I=1,INH
      SUM = SUM + ZM(I,IK) * ZM(I,IK) + ZM(I,IK+1) * ZM(I,IK+1)
  225 CONTINUE
      SUM = SQRT( SUM )
      DO 228 I=1,INH
      ZM(I,IK) = ZM(I,IK) / SUM
      ZM(I,IK+1) = ZM(I,IK+1) / SUM
  228 CONTINUE
      IK = IK + 1
  230 IF( IK .LT. INH ) GO TO 214
C         OPTIONAL PRINT OF NORMALIZED E-VECTORS FROM EISPACK
      IF(IPRIF .EQ. 0 ) GO TO 235
      WRITE(6,961)
  961 FORMAT('0',5X,'THE NORMALIZED E-VECTORS OF HAM ARE:')
      CALL PRIMAT(INHMX,INH,INH,ZM)
  235 CONTINUE
C         RE-ARRANGE THE E-VECTORS
C         TEST FOR CONTINUOUS/DISCRETE CASE
        IF( IDISF .NE. 0 ) GO TO 260
C
C         SELECT THE E-VECTORS ASSOCIATED WITH POS E-VALUES
C         AND PUT THEM INTO THE PARTITIONED MATRICES W11M,W21M
C         TRANSPOSE W11M,W12M
```

```
      J = 0
      DO 250 IK=1,INH
        IF(WRV(IK) .LT. 0.0D0 ) GO TO 250
C         NO POS E-VALUE
C          IF E-VALUE IS COMPLEX, THEN TWO E-VECTOR
C          2 COL'S ARE REQ'D.HQR2 COMPUTES POS PART ONLY
C          SINCE NEG PART IS CONJUGATE,WE WILL PICK UP 2ND
C          COL ON NEXT PASS
      J=J+1
      DO 240 I=1,INX
        I1 = I + INX
        W11M(J,I) = ZM(I,IK)
        W21M(J,I) = ZM(I1,IK)
  240 CONTINUE
  250 CONTINUE
        GO TO 280
C          RE-ARRANGE E-VECTORS DIFFERENTLY FOR DISCRETE CASE
C          REF: VAUGHN, IEEE TRANS ON AUTO CONT.,OCT 1970
C
  260 J = 0
        DO 270 IK = 1,INH
          COMPUTE MAG OF E-VAL
C          SUM = WRV(IK) * WRV(IK) + WIV(IK) * WIV(IK)
          SUM = DSQRT( SUM )
C          TEST FOR E-VAL OUTSIDE OF UNIT CIRCLE
        IF( SUM .LT. 1.0D0 ) GO TO 270
C          E-VAL IS OUTSIDE OF UNIT CIRCLE, GET & STORE E-VECT
C          IF E-VAL IS COMPLEX, WE WILL PICK UP OTHER PART OF
C          E-VECT AUTOMATICALLY ON NEXT PASS
        J = J + 1
        DO 270 I = 1,INX
          I1 = I + INX
          W11M(J,I) = ZM(I,IK)
          W21M(J,I) = ZM(I1,IK)
  270 CONTINUE
C
C          OPTIONAL PRINT OF POS E-VAL EIG VECTORS
  280 IF(IPRIF .EQ. 0 ) GO TO 290
C         PRINT W11M,W21M
      WRITE(6,970)
  970 FORMAT('0',5X,'W11(T) IS')
      CALL PRIMAT(INXMX,INX,INX,W11M)
      WRITE(6,972)
  972 FORMAT('0',5X,'W21(T) IS')
      CALL PRIMAT(INXMX,INX,INX,W21M)
  290 CONTINUE
C
C         LQR SS GAIN  G = -R(INV) * B(T) * K
C         CONT KAL FILT SS GAIN = K * C(T) * R(INV)
C         WHERE K = SS SOL TO ALG RICCATI EQN
C          VIA POTTER'S METHOD
C          W11(T) * K = W21(T)
C         SOLVE FOR COLS OF K BY LU DECOMP METHOD
C          REF: FORSYTHE & MOLER "COMPUTER SOLUTION OF LINEAR
C          ALGEBRAIC SYSTEMS"
C          FIRST DO LU DECOMPOSITION
      CALL DECOMP(INX,W11M,INXMX,ZM,INHMX,IV1,TV1)
C          NOW COMPUTE COL'S OF KM
C
      DO 300 J=1,INX
      CALL SSOLVE(INX,ZM,INHMX,W21M(1,J),KM(1,J),IV1)
  300 CONTINUE
C          TEST FOR KAL FILT
        IF( ITYPE .NE. 0 ) GO TO 310
C          TEST FOR ADD'L PRINT OUT
      IF( IPRIF .EQ. 0 ) GO TO 318
C          PRINT OUT R(INV) * B(T)
      WRITE(6,975) INU,INX
  975 FORMAT('0',5X,'R( INV ) * B( TRANS ) MATRIX',I2,'  BY  ',I2)
      CALL PRIMAT(INXMX,INU,INX,TM)
      GO TO 315
C           TEST FOR ADD'L PRINT OUT
  310 IF( IPRIF .EQ. 0 ) GO TO 318
```

```
C         PRINT OUT C(T) * R(INV) MATRX IN TM
      WRITE(6,976) INX,INZ
  976 FORMAT('0',5X,'C(T) * R(INV)  MATRIX',I2,' BY ',I2)
        CALL PRTMAT(INXMX,INX,INZ,TM)
C         NOW PRINT OUT SS K-MTRX
  315 WRITE(6,980)
  980 FORMAT('0',5X,'SS K-MTRX(RICCATI EQN SOLUTION IS')
      CALL PRTMAT(INXMX,INX,INX,KM)
  318 WRITE(6,985)
  985 FORMAT('0',5X,'THE SS GAIN MTRX G IS :')
C         TEST FOR KAL FILT
        IF( ITYPE .NE. 0 ) GO TO 320
C         COMPUTE SS LQR GAIN:  R(INV) * B(T) IS IN TM
        CALL MULMAT(INXMX,INU,INX,TM,INXMX,INX,KM,INHMX,WM)
      CALL PRTMAT(INHMX,INU,INX,WM)
        GO TO 1
  320   IF( IDISF .NE. 0 ) GO TO 325
C         COMPUTE SS KAL FILT GAIN: C(T) * R(INV) IN TM
      CALL MULMAT(INXMX,INX,INX,KM,INXMX,INZ,TM,INHMX,WM)
C         PRINT SS GAIN MTRX
      WRITE(6,788)
  988 FORMAT(' ',5X,'G(SS) = K(SS) * C(T) * R(INV) FOR CONT KF')
      CALL PRTMAT(INHMX,INX,INZ,WM)
      GO TO 1

C         COMPUTE DISCR KF SS GAIN G=K*C(T)*[C*K*C(T)+R](INV)
C         FIRST COMPUTE & SAVE K*C(T)
  325   DO 330 I=1,INX
        DO 330 J=1,INZ
          W11M(I,J) = 0.0D0
        DO 330 IK=1,INX
  330     W11M(I,J) = W11M(I,J) + KM(I,IK) * CM(J,IK)
        CALL MULMAT(INXMX,INZ,INX,CM,INXMX,INZ,W11M,INXMX,W21M)
        DO 335 I=1,INZ
  335   W21M(I,I) = W21M(I,I) + RM(I,I)
C         INVERT W21M
      CALL MATINV(INXMX,INXMX,INHMX,INHMX,INX,W21M,AIM,WM,ZM,
     X  TV1,IV1)
C         COMPUTE GF(SS) = W11M * AIM
        CALL MULMAT(INXMX,INX,INZ,W11M,INXMX,INZ,AIM,INXMX,W21M)
        WRITE(6,990)
  990 FORMAT('0',5X,'SS FILTER GF(SS) IS:')
        CALL PRTMAT(INXMX,INX,INZ,W21M)
C         COMPUTE KF PREDICTOR SS GAIN GP(SS) = A * GF(SS)
        CALL MULMAT(INXMX,INX,INX,AM,INXMX,INZ,W21M,INXMX,W11M)
        WRITE(6,992)
  992 FORMAT('0',5X,'SS PREDICTOR GAIN IS:')
        CALL PRTMAT(INXMX,INX,INZ,W11M)
      GO TO 1
 9999 CONTINUE
      END
      SUBROUTINE PRTMAT(IRMAX,IR,IC,MAT)
C
C         PRINTS OUT A MATRIX BY ROWS
C
      IMPLICIT REAL*8 (A-H,K-Z)
      DIMENSION MAT(IRMAX,1)
      DO 10 I=1,IR
        WRITE(6,90) (MAT(I,J),J=1,IC)
   10 CONTINUE
   90 FORMAT('0',1P5D15.7/(' ',1P5D15.7))
      RETURN
      END
      SUBROUTINE READM(IRMAX,IR,IC,MAT)
C
C         READS IN A MATRIX BY ROWS
C
      IMPLICIT REAL*8 (A-H,K-Z)
      DIMENSION MAT(IRMAX,1)
      DO 10 I=1,IR
      READ(5,90) (MAT(I,J),J=1,IC)
   10 CONTINUE
```

```
   90 FORMAT(5D12.8)
      RETURN
      END
        SUBROUTINE MULMAT(IRAMX, IRA, ICA, A, IRBMX, ICB, B, IRCMX, C)
C         C = A * B,  MULTIPLIES 2 MATRICES
C
        IMPLICIT REAL*8 (A-H,K-Z)
        DIMENSION A(IRAMX,1),B(IRBMX,1),C(IRCMX,1)
        DO 10 I=1,IRA
        DO 10 J=1,ICB
          C(I,J) = 0.0D0
        DO 10 IK=1,ICA
   10 C(I,J) = C(I,J) + A(I,IK) * B(IK,J)
        RETURN
        END
        SUBROUTINE MATINV(INAMX, INBMX, INCMX, INDMX, INA, A, B, C, D, TV, IV)
C
C         INVERTS THE MATRIX A BY SOLVING THE SET:  A * B = C FOR B.
C         WHERE C = A UNIT MATRIX;  B = A(INV)
C         METHOD: LU DECOMPOSITION;  REF: FORSYTHE & MOLER,
C         'COMPUTER SOLUTIONS OF LINEAR ALGEBRAIC SYSTEMS'
C         SUBROUTINES: DECOMP & SOLVE , SUPPLIED BY DR. L
C         ANDERSON OF VPI
C
      IMPLICIT REAL*8 (A-H,K-Z)
        DIMENSION A(INAMX,1),B(INBMX,1),C(INCMX,1),D(INDMX,1),TV(1)
        INTEGER IV(1)
C         DECOMPOSE A INTO LU FACTORS
        CALL DECOMP(INA,A,INAMX,D,INDMX,IV,TV)
C         SOLVE THE LINEAR SET A * B(1,J) = I FOR J=1,INA
C         SET C TO IDENTITY MTRX
        DO 10 J=1,INA
        DO 5  I=1,INA
    5 C(I,J) = 0.0D0
        C(J,J) = 1.0D0
   10 CALL SSOLVE(INA,D,INDMX,C(1,J),B(1,J),IV)
C         NOTE:  C IS A MATRIX FOR CONVENIENCE
      RETURN
      END


      SUBROUTINE DECOMP(N,A,NA,UL,NU,IPS,SCALES)
      REAL*8 A(NA,1),UL(NU,1),SCALES(1)
      REAL*8 ROWNRM,SIZE,BIG,PIVOT,EM
      INTEGER IPS(1)
C   INITIALIZE IPS, UL, AND SCALES
      DO 5 I=1,N
      IPS(I) = I
      ROWNRM = 0.0D0
      DO 2 J=1,N
      UL(I,J) = A(I,J)
      IF(ROWNRM-DABS(UL(I,J)))1,2,2
    1 ROWNRM = DABS(UL(I,J))
    2 CONTINUE
      IF(ROWNRM)3,4,3
    3 SCALES(I) = 1./ROWNRM
      GO TO 5
    4 CALL SING(1)
      SCALES(I) = 0.0D0
    5 CONTINUE
C   GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
      NM1 = N-1
      DO 17 K=1,NM1
      BIG = 0.0D0
      DO 11 I=K,N
      IP = IPS(I)
      SIZE = DABS(UL(IP,K))*SCALES(IP)
      IF(SIZE-BIG)11,11,10
   10 BIG = SIZE
      IDXPIV= I
   11 CONTINUE
      IF(BIG)13,12,13
```

```
   12 CALL SING(2)
      GO TO 17
   13 IF(IDXPIV-K)14,15,14
   14 J=IPS(K)
      IPS(K) = IPS(IDXPIV)
      IPS(IDXPIV) = J
   15 KP = IPS(K)
      PIVOT = UL(KP,K)
      KP1 = K + 1
      DO 16 I=KP1,N
      IP = IPS(I)
      EM = -UL(IP,K)/PIVOT
      UL(IP,K) =  -EM
      DO 16 J=KP1,N
      UL(IP,J) = UL(IP,J) + EM*UL(KP,J)
   16 CONTINUE
   17 CONTINUE
      KP = IPS(N)
      IF(UL(KP,N))19,18,19
   18 CALL SING(2)
   19 RETURN
      END
      SUBROUTINE SSOLVE(N,UL,NU,B,X,IPS)
      REAL*8 UL(NU,1),B(1),X(1),SUM
      INTEGER IPS(1)
      NP1 = N + 1
      IP = IPS(1)
      X(1) = B(IP)
      DO 2 I=2,N
      IP = IPS(I)
      IM1 = I - 1
      SUM = 0.0D0
      DO 1 J=1,IM1
    1 SUM = SUM + UL(IP,J)*X(J)
    2 X(I) = B(IP) - SUM
      IP = IPS(N)
      X(N) = X(N)/UL(IP,N)
      DO 4 IBACK = 2,N
      I = NP1 - IBACK
      IP = IPS(I)
      IP1 = I + 1
      SUM = 0.0D0
      DO 3 J=IP1,N
    3 SUM = SUM + UL(IP,J)*X(J)
    4 X(I) = (X(I)-SUM)/UL(IP,I)
      RETURN
      END
      SUBROUTINE SING(IWHY)
      DATA NOUT/6/
      GO TO(1,2,3),IWHY
    1 WRITE(NOUT,11)
      RETURN
    2 WRITE(NOUT,12)
      RETURN
    3 WRITE(NOUT,13)
      RETURN
   11 FORMAT(' MATRIX WITH ZERO ROW IN DECOMPOSE.')
   12 FORMAT(' SINGULAR MATRIX IN DECOMPOSE.  ZERO DIVIDED IN SOLVE.')
 13      FORMAT(' NO CONV IN IMPROVE, MATRIX IS NEARLY SINGULAR')
         END
```

END

FILMED

3-84

DTIC