A FORTRAN PROGRAM TO SOLVE THE STEADY-STATE MATRIX RICCATI EQUATION OF OPTIMAL CONTROL

Donald Edward Heath

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A Fortran Program to Solve the Steady-State Matrix Riccati Equation of Optimal Control

by

Donald Edward Heath

Thesis Advisor: R. A. Hess

June 1972

Approved for public release; distribution unlimited.

A Fortran Program to Solve the Steady-State Matrix

Riccati Equation of Optimal Control

by

Donald Edward Heath
Ensign, United States Navy
B.A.A.E., The Ohio State University, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1972

ABSTRACT

This thesis presents a Fortran program that numerically solves the steady-state matrix Riccati equation of the quadratic cost optimal control problem. Each step of the program is presented, analytically and computationally. The check points incorporated in the program and the input parameters that can be used to assure a correct solution are identified and discussed. Difficulties encountered when verifying the program, and the suggested solutions, are also presented.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I gratefully acknowledge my thesis advisor, Assistant Professor Ron Hess, and the staffs of the W.R. Church Computer Center and the Dudley Knox Library for their assistance in preparing this thesis. I thankfully acknowledge Professor Louis B. Schmidt for serving as the second reader of this thesis. I also acknowledge my wife for her patience and understanding during preparation of this thesis.

I. INTRODUCTION

The Fortran computer program presented in this thesis provides a relatively quick and reliable means for determining the unique, symmetric, steady-state solution P of the nonlinear matrix Riccati equation:

$$\dot{P} = 0 = - PA - A'P - C'C + PBR^{-1}B'P \tag{1}$$

occurring in optimal control theory. The remaining equation variables are defined in the following statement of the quadratic cost optimal control problem.

Given the linear, time-invariant, completely controllable system defined by the state equations:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2}$$

$$x(0) = x_0 \tag{3}$$

where:

$x(t)$ is an $n \times 1$ state vector

$u(t)$ is an $m \times 1$ unconstrained control vector

A is an $n \times n$ matrix

B is an $n \times m$ matrix,

determine the control vector $u^*(\cdot)$ which minimizes the quadratic cost functional:

$$J(x_0;u(\cdot)) = \int_0^\infty [\, x'(t)C'Cx(t) + u'(t)Ru(t) \,]\, dt \tag{4}$$

where:

C is an $m \times n$ matrix

R is an $m \times m$ symmetric, positive definite matrix

and the matrix pair (A,C) is completely observable.

It has been shown [Ref. 1] that $u^*(\cdot)$ is given by the linear feedback law:

$$u^*(t) = -R^{-1}B'Px(t) = -L^*x(t) \tag{5}$$

where P is the unique solution to matrix equation (1).

For the system of equation (2) to be completely controllable, the n x mn augmented matrix G,

$$G = (B \mid AB \mid A^2B \mid \cdots \mid A^{n-1}B\ ), \tag{6}$$

must contain n linearly independent column vectors, or equivalently, the rank of G must be n. For the matrix pair (A,C) to be completely observable, the n x mn augmented matrix H,

$$H = (\ C' \mid A'C' \mid (A')^2C' \mid \cdots \mid (A')^{n-1}C'\ ),$$

must contain n linearly independent column vectors or the rank of H must be n.

When $u^*(\cdot)$ is given by equation (5), equation (2) can be rewritten as:

$$\dot{x}(t) = Ax(t) - BL^*x(t) = (A - BL^*)x(t) \tag{7}$$

which, using equation (3), has the general solution:

$$x(t) = e^{(A - BL^*)t}\ x_0\ . \tag{8}$$

Because the system is completely controllable, as time approaches infinity x(t) must remain bounded. To satisfy this requirement, the matrix (A - BL*) must be a stable matrix or, alternatively, all the eigenvalves of the matrix (A - BL*) must have negative real parts. As a consequence of this:

$$\lim_{t\to\infty} x(t) = x_0 \lim_{t\to\infty} e^{(A - BL^*)} = 0\ . \tag{9}$$

To evaluate the quadratic cost associated with the optimal control $u^*(t)$, substitute equation (5) into equation (4) and write:

$$J^* = J(x_o;u^*(t)) = \int_o^\infty [x'(t)C'Cx(t) + x'(t)L^{*\prime}RL^*x(t)]\, dt. \qquad (10)$$

Substituting equation (8) for $x(t)$, expanding $L^{*\prime}RL^*$, and factoring out the common terms:

$$J^* = x_o' \left[\int_o^\infty e^{(A - BL^*)'t}[C'C + PBR^{-1}B'P]\, e^{(A - BL^*)t}\, dt \right] x_o. \qquad (11)$$

From equation (1) $C'C = -PA - A'P + PBR^{-1}B'P$ or:

$$C'C + PBR^{-1}B'P = -P(A - BL^*) - (A - BL^*)'P\ . \qquad (12)$$

Substituting equation (12) into equation (11) and integrating by parts the first term of the resulting integral expression, we get:

$$\int_o^\infty e^{(A - BL^*)'t}\, [-P(A - BL^*)]\, e^{(A - BL^*)t}\, dt$$

$$= - e^{(A - BL^*)'t}\, [P]\, e^{(A - BL^*)t} \Big|_{t=0}^{\infty}$$

$$+ \int_o^\infty (A-BL^*)'\, e^{(A - BL^*)'t}\, [P]\, e^{(A - BL^*)t}\, dt\ . \qquad (13)$$

Using equation (9) to evaluate the upper limit of the first term, the first term reduces to P. Recognizing:

$$Ze^{Zt} = Z \sum_{i=0}^{\infty} \frac{1}{i!} Z^i t^i = e^{Zt}Z \qquad (14)$$

we see that the second term in equation (13) will cancel the second term in equation (11) when equation (12) is substituted. Thus:

$$J^* = x_o'Px_o\ . \qquad (15)$$

II. ANALYTICAL METHODS OF SOLUTION

A. ANALYTICAL METHOD OF KLEINMAN

The main computer program of this thesis is based upon a method for solving the steady-state Riccati equation published by David L. Kleinman [Ref. 2] in 1968. The iteration scheme for solving euqation (1) is presented in the following theorem by Kleinman.

1. Kleinman's Theorem

Let V_k, k = 0, 1, 2, ... be the n x n (unique) positive definite matrix solution of the linear algebraic matrix equation:

$$0 = A_k'V_k + V_kA_k + C'C + L_k'RL_k \qquad (16)$$

where, recursively,

$$L_k = R^{-1}B'V_{k-1} \qquad k = 1,2,3,\ldots$$

$$A_k = A - BL_k \qquad k = 0,1,2,\ldots$$

and where L_0 is chosen such that the matrix $A_0 = A - BL_0$ has eigenvalues with negative real parts.

Then: 1) $P \leq V_{k+1} \leq V_k \leq \cdots \qquad k = 0,1,2,\ldots$

2) $\lim_{k\to\infty} V_k = P$.

The notation $X > Y$ $[X \geq Y]$ means that the matrix $X - Y$ is positive [semi] definite.

2. The Cost Matrix

The proof of this theorem requires the introduction and definition of a cost matrix V_L. Assume that $u_L(x(t)) = - Lx(t)$ is an arbitrary feedback law, with feedback gains of L, and $u_L(x(t))$ is applied to the

system of equation (2). Following a development similar to equations (7), (8) and (15), the resulting quadratic cost function is:

$$J(x_o;u_L(x(t))) = x_o'V_Lx_o$$

where V_L is the cost matrix associated with the feedback gains L and is given by:

$$V_L = \int_0^\infty e^{(A - BL)'t} [C'C + L'RL] e^{(A - BL)t} dt . \tag{17}$$

V_L is bounded if and only if the closed-loop system control matrix (A - BL) is stable. If V_L is bounded, it becomes the unique (positive definite) solution of the linear matrix equation:

$$0 = (A - BL)'V_L + V_L(A - BL) + C'C + L'RL . \tag{18}$$

Examining the first term of equation (18) and substituting equation (17) for V_L, we can verify this relationship:

$$(A - BL)'V_L = \int_0^\infty (A - BL)' e^{(A - BL)'t}[C'C + L'RL] e^{(A - BL)t} dt.$$

Integrating by parts and using equation (14) we have:

$$(A - BL)'V_l = e^{(A - BL)'t} [C'C + L'RL] e^{(A - BL)t}\Big|_{t=0}^{\infty}$$

$$- \int_0^\infty e^{(A - BL)'t}[C'C + L'RL] e^{(A - BL)t} (A - BL) dt.$$

Using equation (9) to evaluate the upper limit of the first term, the first term reduces to C'C + L'RL and we have,

$$(A - BL)'V_L = -C'C - L'RL - V_L(A - BL) ,$$

the desired result.

Recalling the result of equation (15) we see that for the optimal control of equation (5) we have:

$$V_{L*} = P \quad . \tag{19}$$

If L_1 and L_2 are the gain matrices associated with the cost matrices V_1 and V_2, it can be shown [Ref. 3] that:

$$\begin{aligned} V_1 - V_2 = \int_0^\infty e^{(A - BL_2)'t} [&(L_1 - L_2)'R(L_1 - L_2) \\ &- (L_1 - L_2)'(B'V_1 - RL_2) \\ &- (B'V_1 - RL_2)'(L_1 - L_2)] \, e^{(A - BL_2)t} \, dt \end{aligned} \tag{20}$$

or:

$$\begin{aligned} V_1 - V_2 = \int_0^\infty e^{(A - BL_1)'t} [&(L_1 - L_2)'R(L_1 - L_2) \\ &- (L_1 - L_2)'(B'V_2 - RL_2) \\ &- (B'V_2 - RL_2)'(L_1 - L_2)] \, e^{(A - BL_1)t} \, dt. \end{aligned} \tag{21}$$

If either matrix $(A - BL_2)$ or matrix $(A - BL_1)$ is unstable, V_1 or V_2, respectively, will be unbounded and care must be exercised in using the above formulas.

3. Proof of Kleinman's Theorem

1) $P \le V_{k+1} \le V_k \le \cdots \quad k = 0,1,2,\cdots$. Let V_o be the cost matrix for L_o, the initial guess that yields a stable closed-loop system control matrix, and let V_1 be the cost matrix for $L_1 = R^{-1}B'V_o$. Substituting V_o and V_1 into equation (20) and noting:

$$e^{A'_k t} = (e^{A_k t})'$$

$$R = Y'Y, \text{ where } Y \text{ is unique}$$

$$B'V_o - RL_1 = 0$$

we have:

$$V_o - V_1 = \int_0^{\infty} e^{A_1't} [(L_o - L_1)'R(L_o - L_1)] e^{A_1 t} dt .$$

Define $Z(t) = Y(L_o - L_1)e^{A_1 t}$. It has been shown [Ref. 4] that for $Z(t)$ a real matrix:

$$Z'(t)Z(t) \geq 0 \qquad \text{for all } t \geq 0.$$

Thus $V_o - V_1 = \int_0^{\infty} Z'(t)Z(t)\, dt \geq 0$ or:

$$V_o \geq V_1 \quad . \tag{22}$$

Let V_∞ be the cost matrix associated with L^*, use equation (19) and substitute into equation (21). Noting that:

$$B'P - RL^* = 0$$

we have:

$$V_1 - P = \int_0^{\infty} e^{A_1't} [(L_1 - L^*)'R(L_1 - L^*)] e^{A_1 t} \, dt.$$

This time define $Z(t) = Y(L_1 - L^*)e^{A_1 t}$ and we have:

$V_1 - P = \int_0^{\infty} Z'(t)Z(t)\, dt \geq 0$ or:

$$V_1 \geq P \quad . \tag{23}$$

Combining the results of equations (22) and (23) we get:

$$V_o \geq V_1 \geq P$$

meaning that V_1 is bounded above by P and below by V_o. Thus, A_1 is a stable matrix and V_1 satisfies equation (16) with $k = 1$. Similar arguments can be made for $k = 2,3,4,\ldots$ yielding:

$$V_o \geq V_1 \geq V_2 \geq \cdots \quad V_k \geq V_{k+1} \geq P \quad ,$$

the desired result.

2) $\lim_{k\to\infty} V_k = P$. The $\lim_{k\to\infty} V_k = V_\infty$ exists by a theorem on monotonic convergence of positive operators [Ref. 5]. Thus, in the limit $V_k = V_{k+1}$ and $A_k = A - BL_k = A - BR^{-1}B'V_k = A - BR^{-1}B'V_{k+1}$. Since $A_k' = A' - V_kBR^{-1}B'$ equation (16) becomes, in the limit as k approaches infinity,

$$0 = A'V_k - V_kBR^{-1}B'V_k + V_kA - V_kBR^{-1}B'V_k + C'C + V_kBR^{-1}B'V_k$$

which is equation (1), the desired result.

B. BASS'S METHOD OF DETERMINING A STABLE INITIAL GUESS

Kleinman's method requires an initial guess of the feedback gain that yields a stable closed-loop system control matrix. Kleinman remarks [Ref. 2] that, if the system of equation (2) is completely controllable, then the desired initial guess will always exist. A method that could be programmed for a general, controllable system to yield this correct initial guess was sought.

Bass [Ref. 6] presented, but did not publish, a general method for determining a stable initial guess for a completely controllable system in 1961. The method was published in a paper by Wonham and Cashman [Ref. 7] and a proof can be found in a subsequent paper by Bass [Ref.8].

Given the controllabe matrix pair (A,B), the matrix:

$$A_0 = A - BL_0$$

will be stable when $L_0 = B'X^{-1}$, where X is the (unique) positive definite solution to the linear matrix equation:

$$-(A + \beta I)X - X(A + \beta I)' + 2BB' = 0 \qquad (24)$$

where β is defined by:

$$\beta > ||A||$$

where $||\cdot||$ is the Euclidean norm. According to Wonham [Ref. 7] the results are also valid if:

$$\beta > \sqrt{n} \max_{j} \sum_{i=1}^{n} | a_{ij} |$$

where a_{ij} are the elements of A.

C. SOLVING THE MATRIX EQUATION Y'X + XY = Z

Equations (16) and (24) are in the form of the Lyapunov equation:

$$Y'X + XY = Z \tag{25}$$

The methods found in the literature for solving this equation fall into two general categories: series solutions and simultaneous linear equation solutions.

In the series solutions, X is found from the sum of a converging matrix series, i.e., Ref. 9. For the series to converge the matrix Y must be a stable matrix. In solving equation (16) this condition is met; A_k is stable by the definition of a bounded cost matrix. In general, however, the matrix $(A + \beta I)'$ of equation (24) is not stable, and the series method fails to solve equation (24) properly.

Since the unique solution, X, is symmetric, equation (25) represents $n(n + 1)/2$ unknowns. The second category of solutions expands equation (25) into a set of $n(n + 1)/2$ simultaneous linear equations. An economical way of recursively expanding equation (25) was given by Bingulac [Ref. 10], using an integer coefficient matrix to expand the equation. This is the method used to expand equations (16) and (24) to the form Ax = B, which can be solved by a variety of simultaneous equation solvers.

D. ALTERNATE ANALYTICAL METHODS

There are several alternate methods found in the literature for solving the steady-state matrix Riccati equation (1).

A method developed by Bass [Ref. 11] obtains the solution from a 2n-dimensional Hamiltonian, H, and the terms of the polynomial expansion of the stable roots of H. This scheme is considered too sensitive to finite numerical computations to be of practical use.

MacFarlane [Ref. 12] shows that the solution can be obtained from the eigenvectors corresponding to the unstable eigenvalues of a similar Hamiltonian. The scheme requires that the system have distinct eigenvalues and that the corresponding eigenvectors be determined (this is difficult for high order systems).

Blackburn [Ref. 13] programmed a method based on a Newton-Raphson iterative solution for simultaneous nonlinear equations. This scheme requires an initial guess that yields a stable closed-loop system control matrix. The user must determine this initial guess so that it is close enough to the final solution for the local Newton-Raphson method to converge.

A fourth method integrates the full, nonsteady-state Riccati equation (1) backwards in time, from a set of zero initial conditions, until each element of the $\dot{P}$ matrix reaches a satisfactory, small value. For systems of even moderate order, this method is prohibitive with respect to computation time.

III. COMPUTATIONAL METHOD OF SOLUTION

A. SUBROUTINE RICATS

1. General

Subroutine RICATS is the subroutine that the user will call to solve equation (1). The program language is FORTRAN IV for the Operating System / 360 which is compatible with, and encompasses USASA FORTRAN. The calling arguments, in the required sequence, are explained in the comment cards at the beginning of the subroutine. It should be noted that IA = n and JB = m. Basically the subroutine iterates on equation (16), using equation (24) to calculate the initial guess, until the solution converges.

For first order systems (n = 1), the nonlinear equation (1) becomes a quadratic equation. For these systems, subroutine RICATS solves the resulting quadratic and returns the largest root in the first element of the P matrix .

2. The System Controllability Check

An analytical requirement of Bass's method of determining the initial guess is that the matrix pair (A,B) be completely controllable. To check this requirement the augmented matrix G of equation (6) is formed and subroutine GMRANK is called to determine the rank of G. If the rank equals n, execution continues, otherwise subroutine RICATS returns with IER = 2.

3. The Initial Guess Stability Check

Kleinman's iteration scheme requires that the eigenvalues of the closed-loop system control matrix of the initial guess have negative real parts in order for the initial cost matrix V_0 to be bounded.

In subroutine RICATS, when the stability check is requested, the matrix $A - BL_0$ is copied into a work matrix and then passed to the IBM/SSP subroutines HSBG and ATEIG. The eigenvalues computed by ATEIG are then individually tested to be algebraically less than minus the absolute value of EIGMAX, where EIGMAX is set by the user. If any eigenvalue fails the check RICATS returns with IER = 3.

4. The Solution Positive Definite Check

Kleinman's iteration scheme is supposed to converge to a positive definite solution. The iterations can converge to a nonpositive definite solution if all of Kleinman's theorem requirements are not numerically met. Therefore, for the user's convenience, a check to verify that the solution is positive definite is included. In subroutine RICATS, when the positive definite check is requested, the solution is factored by the Cholesky square-root method. A nonpositive term on the diagonal of the factorized matrix constitutes a nonpositive definite solution and RICATS returns with IER = 4 + KL, where KL = 0 is for a converged solution and KL = 1 is for NTRY iterations without convergence.

5. The Steps of Subroutine RICATS

Calling subroutine RICATS causes the following sequential steps to be executed.

1. Check input parameters IA, JB, IER, NTRY to insure they are within proper bounds. If check fails IER = 6, NN = 0 and RICATS returns.

2. If the user requests, check the controllability of the inputed system. If check fails IER = 2, NN = 0 and RICATS returns.

3. Set NN = 0.

4. If this is a first order system (n = 1), go to step 30.

5. Form $E = - BR^{-1}B'$.

6. Form $F = 2BB'$.

7. Form $P = (A + \beta I)'$ using equation (24) to define β where the "greater than" magnitude is provided by the variable FIX or:

$$\beta = \text{FIX}\sqrt{n}\ \max_{j} \sum_{i=1}^{n} | a_{ij} | \cdot$$

8. Solve $(A + \beta I)X + X(A + \beta I)' = 2BB'$.

9. If subroutine SIMQ, through subroutine MLIAPS, returns a nonzero error flag, NN = 1.

10. Form $L_0 = BX^{-1}$.

11. Form $P = A - BL_0$.

12. If the user requests, check the stability of the system matrix of the initial guess. If the check fails, IER = 3 and RICATS returns.

13. Form $F = - Q - L_0'RL_0$.

14. Solve $(A - BL_0)'V_1 + V_1(A - BL_0) = - Q - L_0'RL_0$.

15. If subroutine SIMQ, through subroutine MLIAPS, returns a nonzero error flag, NN = NN + 1.

16. Set k = 1, KL = 0.

17. Form $P = EV_k$.

18. Form $F = - Q + V_kP$.

19. Form $P = A + P$.

20. Solve $(A + EV_k)'V_{k+1} + V_{k+1}(A + EV_k) = - Q + V_kEV_k$.

21. If subroutine SIMQ, through subroutine MLIAPS, returns a nonzero error flag, NN = NN + 1. If NN > (n + 1)/2, go to step 29.

22. Check each element of V_{k+1} by ABS ($(v_k - v_{k+1})/v_k$) $\leq$ TOLER. If all elements of V_{k+1} pass this test, go to step 27.

23. Form $V_k = V_{k+1}$.

24. If k $\geq$ NTRY; go to step 26.

25. Set k = k+1; go to step 17.

26. Set KL = 1.

27. If the user requests, check to see if V_{k+1} is a positive definite matrix. If check fails IER = 4 + KL and RICATS returns.

28. Set IER = KL and RICATS returns.

29. Set IER = 7 and RICATS returns.

30. If the user requests the controllability check and B(1) is zero, IER = 2, NN = 0 and RICATS returns.

31. Set $P = AR/BB + \sqrt{(AR/BB)^2 + C'CR/BB}$

32. If the user requests the positive definite check and P $\leq$ 0.1E-35, IER = 4, NN = 0 and RICATS returns.

33. Set IER = 0 and RICATS returns.

B. SUBROUTINE MLIAPS

Subroutine MLIAPS is an auxiliary routine used by subroutine RICATS. The subroutine expands the equation in steps 8, 14 and 20 of subroutine RICATS into n(n + 1)/2 simultaneous linear equations of the form Ax = B. The method used was presented by Bingulac [Ref. 10] in 1970. These n(n = 1)/2 simultaneous linear equations are then solved by subroutine SIMQ. Upon return from subroutine SIMQ, MLIAPS immediately returns to subroutine RICATS and any error codes returned by SIMQ are passed, unchanged, to RICATS.

C. SUBROUTINE GMRANK

Subroutine GMRANK is a general subroutine for determining the minimum row or column rank of an arbitary real matrix. The method used is simple row and column interchanges for maximum pivoting, with successive

reduction on the remaining matrix elements [Ref. 14]. When the absolute value of a pivot element is less than 0.1E-35, or when the final pivot element has been determined, the subroutine returns the integer K, where K is the number of successfully determined nonzero pivot elements or, equivalently, the rank of the inputed matrix.

IV. OPERATIONAL ASPECTS OF USING SUBROUTINE RICATS

A. CONVERGENCE PROPERTIES

Kleinman suggests that equation (16) will converge to a satisfactory solution within seven to ten iterations, using an initial guess generated by hand or from a previous solution. Subroutine RICATS, using a correct set of input parameters and Bass's method of generating an initial guess, converged within forty iterations for every system tested.

To test subroutine RICATS, over seventy-five systems were run and a satisfactory solution was returned for each one. As a test of how accurate the solutions were, the average absolute value of the matrix $\dot{P}$ of equation (1) was determined. For the satisfactory solutions the average values were of the order of 10^{-6}. The standard input parameters were:

NTRY = 50
TOLER = 0.1E-03
FIX = 1.1
EIGMAX = 0.001
IER = 0

and the usual error flag returned was IER = 0 (see the discussion of the parameter IER for high order systems).

B. INPUT PARAMETERS

Through its input parameters subroutine RICATS was written to be as flexible as possible for the user. For any system, the five parameters NTRY, TOLER, FIX, EIGMAX, IER influence the final returned solution. It is hoped that the user can tailor these input parameters to meet his particular needs.

In the following discussion, "high" or "higher order systems," refer to systems of the eighth order and above ($n \geq 8$).

1. NTRY

NTRY is the maximum number of iterations the user desires to be attempted, before returning to the user's program without a converged solution. A recommended value is fifty and the user is usually assured that a solution that is going to converge, will converge within fifty iterations. It should be noted that some initial guesses generated may yield marginally stable system control matrices, and these solutions will require a larger number of iterations to converge. From experience, one hundred and fifty iterations was considered to be a sufficient practical upper limit.

2. TOLER

TOLER is the maximum percentage difference, between each element of the solution, on successive iterations for the convergence criterion to be satisfied. The accuracy of single-precision computations is about five significant digits, so decreasing TOLER beyond 0.0001 will not result in an appreciably more accurate solution. Decreasing the magnitude of TOLER (up to this limit) will tend to increase the accuracy and the number of iterations required for the desired solution.

3. FIX

FIX is the constant used to insure the magnitude of β in equation (24) is greater than the Euclidian norm of the A matrix. Analytically then, FIX should be greater than one and the suggested value of 1.1 worked well for the majority of systems. However, for some systems equation (24) generates a singular initial guess, evidenced by an underflow error message from subroutine MINV. These singular initial guesses can still yield quite satisfactory solutions within five or six iterations. However, the user can, by varying FIX through

the suggested range, remove this underflow from his execution. From practical experience, the range of FIX is from 0.1 up to about three or four.

4. EIGMAX

Once the initial guess L_0 is calculated, the user has the option through IER of verifying that the associated control matrix is stable. If the user asks for the stability check, the eigenvalues of the control matrix are determined. Each must have its real part more negative than minus the absolute value of EIGMAX. From the discussion on cost matrices, theoretically EIGMAX could be set to zero. Numerically though, an eigenvalue that is very close to zero can induce numerical instability in the iteration scheme. From experience, the suggested value is 0.001. The user can also use FIX to modify the control matrix of the initial guess in an attempt to make the real parts of the eigenvalues negative enough to pass the stability check.

5. IER

The most informative parameter of the group is IER. On return from RICATS, IER can inform the user of the validity of the solution. When speicfying IER as in input parameter, the user can, by bypassing various combinations of the three checks available in RICATS, overcome some system deficiencies, save execution time, or obviate decks for the subroutines GMRANK, ATEIG, HSBG (see note 6 in comment cards at the beginning of subroutine RICATS). By bypassing any or all of the three checks (if the user is fairly certain that the circumvented checks would have been passed) the user can save execution time, although the savings are neglibile except for high order systems.

Care must be exercised when not checking controllability and or stability. For systems that are uncontrollable, neglecting the controllability check may lead to a convergent solution, if the stability check were passed. Such a solution could be positive definite and yield a stable final-solution control matrix, but the user should keep in mind that there may be modes of the system that cannot be controlled. For the case of unstable control matrices corresponding to the initial guess, the stability check should be bypassed only after FIX has been varied through the suggested range of values without success. The danger in attempting to generate a solution for a system that would fail the stability check is that successive iterations are no longer bounded by a lower positive [semi] definite iteration. The iterations will probable not converge to a satisfactory answer. This is the most dangerous of the three checks to bypass, by far.

The positive definite check of the solution is included as a convenience for the user, to verify that the final solution is indeed positive definite. A note of caution should be sounded when solving high order systems. The positive definite check is subject to numerical problems when evaluating the high order solutions, and thus a solution will be flagged as not being positive definite, when for all practical purposes it is. Refinements in the solution, from introducing iterative routines for solving equation (16) (see subroutine SIMQ below), have shown that the difference between passing and failing the check can depend solely on numerically insignificant digits in a few elements of the solution. Therefore, to give confidence to a solution that has been flagged as not being positive definite, the user can: (1) determine the eigenvalues of the final solution, closed-loop system control

matrix and check for all negative real parts; or (2) calculate the value of the optimal quadratic cost function for an arbitrary set of initial conditions using equation (15), and check for a positive, finite result. The positive definite check will yield the same result as the IBM/SSP subroutine MFSD.

C. HIGH ORDER SYSTEMS

As the order of the system increases, the problems due to finite numerical calculations increase considerably. One means of trying to maintain sufficient accuracy is to have a double-precision version of subroutine RICATS. Since this would nearly double the storage requirements (prohibitive for systems of order higher than twenty) it was felt that this was not a feasible means of achieving the goal. The suggested procedure for systems of higher order is for the user to introduce his own dummy subroutines MINV and SIMQ. Care must be exercised when writing your own subroutines. The variables passed to and returned from your subroutine must correspond exactly to the variables as handled by the subroutine you are replacing.

1. Subroutine MINV

Using a common statement from the user's main program to provide the additional storage required, the user can write a routine to convert the matrix to be inverted to double-precision storage, invert the matrix in double-precision, then convert the inverted matrix back to the single-precision mode. When this is done, the inverted matrix is passed back to the subroutine RICATS as if the IBM/SSP subroutine MINV had done the inverting. A logical way to invert the matrix in the double-precision mode is to use the IBM/SSP subroutine DMINV. It might appear

that this type of routine would make no noticeable change in the execution of subroutine RICATS. However, for some systems tested, particularly those with singular initial guesses (FIX = 1.1), the number of iterations required for solution was halved.

2. Subroutine SIMQ

Again using a common statement, recognizing that work areas can be declared single-precision in one subroutine and double-precision in another, the user can write his own simultaneous linear-equation-solving routine. The IBM/SSP subroutine SIMQ has been found to yield satisfactory results for a maximum of about forty-five equations (n = 9). For systems of higher order, the solutions did converge, but they were usually accompanied by an error flag indicating that they were not positive definite solutions. However, iterative refinement of the solution to equation (16) at each iteration can yield error-free results. The user can either write his own routine for the iterative refinement, or pass the set of equations to the IBM/SSP subroutines FACTR and RSLMC. It is suggested that the common statement also contain a relative tolerance parameter, not necessarily the same as TOLER inputed to RICATS, in either name or magnitude. As was previously mentioned, this subroutine technique was used to remove error flags from the solutions returned by subroutine RICATS.

```
C
C     ..................................................................
C
C        SUBROUTINE RICATS
C
C        PURPOSE
C           TO ITERATIVELY DETERMINE THE UNIQUE, SYMMETRIC,
C           POSITIVE-DEFINITE SOLUTION P TO THE NONLINEAR,
C           STEADY-STATE MATRIX RICCATI EQUATION
C           .                                   -1
C           P = 0 = - P*A - A'*P - C'*C + P*B*R   *B'*P
C           OF OPTIMAL CONTROL THEORY. SEE NOTE 10.
C
C        USAGE
C           CALL RICATS (A,B,Q,R,P,D,E,F,V,L,LI,MI,IA,JB,KQ,
C          1             NTRY,TOLER,FIX,EIGMAX,IER,NN)
C
C        DESCRIPTION OF PARAMETERS
C           A      - GENERAL IA BY IA INPUT MATRIX. SEE NOTE
C                    3.
C           B      - GENERAL IA BY JB INPUT MATRIX.
C           Q      - LOWER TRIANGULAR (OR UPPER TRIANGULAR,
C                    SEE KQ) PART OF A SYMMETRIC, POSITVE
C                    SEMIDEFINITE IA BY IA INPUT MATRIX.
C                    Q = C'*C.
C           R      - GENERAL PART OF A SYMMETRIC, POSITIVE
C                    DEFINITE JB BY JB INPUT MATRIX.
C           P      - IA BY IA WORK MATRIX. ON RETURN P CON-
C                    TAINS THE GENERAL FORM OF THE SOLUTION.
C           D      - MM BY MM WORK MATRIX. SEE NOTE 1.
C           E      - MM BY  1 WORK VECTOR.
C           F      - MZ BY  1 WORK VECTOR. SEE NOTE 1.
C           V      - MM BY  1 WORK VECTOR.
C           L      - IA BY IA INTEGER WORK MATRIX.
C           LI     - IA BY  1 INTEGER WORK VECTOR.
C           MI     - IA BY  1 INTEGER WORK VECTOR.
C           IA     - ROW AND COLUMN DIMENSION OF MATRICES A,
C                    Q AND RETURNED SOLUTION P. ROW
C                    DIMENSION OF MATRIX B.
C           JB     - COLUMN DIMENSION OF MATRIX B. ROW AND
C                    COLUMN DIMENSION OF MATRIX R. JB <= IA.
C           KQ     - INTEGER INPUT CONSTANT.
C                    = 0 - Q MATRIX IS STORED COLUMNWISE IN
C                          LOWER TRIANGULAR FORM.
C                    = 1 - Q MATRIX IS STORED COLUMNWISE IN
C                          UPPER TRIANGULAR FORM (SEE
C                          IBM/SSP SUBROUTINE MSTR).
C           NTRY   - MAXIMUM NO. OF ITERATIONS TO BE TRIED.
C                    2 > NTRY > 151.
C           TOLER  - INPUT CONSTANT WHICH IS USED AS A
C                    RELATIVE TOLERANCE FOR TEST OF
C                    CONVERGENCE.
C           FIX    - INPUT CONSTANT. THEORETICALLY SHOULD BE
C                    >= 1.0. SUGGESTED VALUE IS 1.1. SEE
C                    NOTE 7.
C           EIGMAX - MINIMUM ACCEPTABLE MAGNITUDE OF THE
C                    NEGATIVE REAL PARTS OF THE EIGENVALUES
C                    OF THE CONTROL MATRIX ( A - B*V(0) ) OF
C                    THE INITIAL GUESS V(0). SUGGESTED VALUE
C                    OF EIGMAX IS OF ORDER 0.001 . SEE NOTE
C                    5. EIGMAX USED ONLY FOR IER = 0,1,4,5.
C           IER    - INTEGER INPUT/OUTPUT PARAMETER.
C                    THE FOLLOWING NOTATION IS USED BELOW :
C              C.A,B. - FOR NUMERICAL CONTROLLABILITY
C                       OF THE MATRIX PAIR (A,B).
C                       SEE NOTES 4,6.
C              S.C.M. - FOR A STABLE CONTROL MATRIX FOR
C                       THE INITIAL GUESS. SEE NOTES 5,6.
C              P.D.S. - FOR A POSITIVE-DEFINITE SOLUTION.
C                       SEE NOTE 8.
C
```

```
C                INPUT - THE VALUE OF IER DETERMINES WHICH
C                        OF THE 3 ABOVE PROPERTIES RICATS
C                        IS TO CHECK DURING EXECUTION. SEE
C                        NOTE 9.
C                = 0 - CHECK C.A,B. ; S.C.M. ; P.D.S.
C                = 1 - CHECK C.A,B. ; S.C.M.
C                = 2 - CHECK C.A,B. ;          P.D.S.
C                = 3 - CHECK C.A,B.
C                = 4 - CHECK          S.C.M. ; P.D.S.
C                = 5 - CHECK          S.C.M.
C                = 6 - CHECK                   P.D.S.
C                = 7 - MAKE NONE OF THE ABOVE 3 CHECKS.
C
C                OUTPUT - ON RETURN IER SERVES AS AN ERROR
C                         FLAG.
C                = 0 - ALL CHECKS REQUESTED WERE PASSED,
C                      SOLUTION CONVERGED WITHIN NTRY
C                      ITERATIONS.
C                = 1 - ALL CHECKS REQUESTED WERE PASSED,
C                      SOLUTIONS DID NOT CONVERGE WITHIN
C                      NTRY ITERATIONS.
C                = 2 - C.A,B. CHECK FAILED. NO SOLUTION.
C                = 3 - S.C.M. CHECK FAILED. NO SOLUTION.
C                = 4 - P.D.S. CHECK FAILED. SOLUTION DID
C                      CONVERGE WITHIN NTRY ITERATIONS.
C                = 5 - P.D.S. CHECK FAILED. SOLUTION DID
C                      NOT CONVERGE BY NTRY ITERATIONS.
C                = 6 - IMPROPER PARAMETERS PASSED TO
C                      RICATS. SEE NOTE 2. NO SOLUTION.
C                = 7 - MORE THAN (IA+1)/2 ERROR FLAGS WERE
C                      RETURNED BY SUBROUTINE SIMQ INDI-
C                      CATING SINGULAR INPUTS TO THAT SUB-
C                      ROUTINE. INCORRECT SOLUTION. SEE
C                      NOTE 7.
C          NN    - INTEGER OUTPUT PARAMETER.
C                  FOR RICATS RETURN WITH IER .NE. 2, NN
C                      IS THE NO. OF ERROR FLAGS RETURNED
C                      BY SUBROUTINE SIMQ.
C                  FOR RICATS RETURN WITH IER .EQ. 2, NN
C                      IS THE RANK OF THE AUGMENTED
C                      CONTROLLABILITY MATRIX. SEE NOTE 4.
C
C      NOTES
C          1.  MM = IA*(IA+1)/2   :   MZ = MAXO( MM,IA*JB ).
C          2.  THE INPUT PARAMETERS IA,IB,IER,NTRY MUST
C          MEET THE FOLLOWING REQUIREMENTS:
C          0 < JB <= IA ; -1 < IER < 8 ; 2 < NTRY < 151  .
C          3.  INPUT MATRICES A,B,Q,R AND OUTPUT MATRIX P
C          ARE ASSUMED STORED IN THEIR RESPECTIVE MODES
C          (GENERAL OR TRIANGULAR) COLUMNWISE IN IBM/SSP
C          COMPRESSED FORM. MATRICES A,B,Q,R ARE RETURNED
C          UNCHANGED.
C          4.  THE METHOD OF OBTAINING THE INITIAL GUESS
C          REQUIRES THAT THE MATRIX PAIR (A,B) IS CONTROLL-
C          ABLE (I.E. THE AUGMENTED MATRIX
C                             2           IA-1
C                  ( B|A*B|A  *B|...|A        *B )
C          HAS RANK IA). THE USER CAN HAVE THIS PROPERTY
C          CHECKED (IER=0,1,2,3) AND IF THE CHECK FAILS,
C          RICATS IMMEDIATELY RETURNS WITH IER = 2.
C          5.  THE ITERATION SCHEME REQUIRES THE CONTROL
C          MATRIX OF THE INITIAL GUESS BE STABLE. THE USER
C          CAN HAVE THIS PROPERTY CHECKED (IER=0,1,4,5).
C          THE REAL PARTS OF THE EIGENVALUES OF THIS MATRIX
C          ARE TESTED TO BE <= -ABS(EIGMAX) . IF ANY CHECK
C          FAILS RICATS IMMEDIATELY RETURNS WITH IER = 3.
```

```
C         6.  IF THE USER DOES NOT HAVE C.A,B. CHECKED,
C         SUBROUTINE GMRANK IS NOT NEEDED. IF THE USER
C         DOES NOT HAVE S.C.M. CHECKED, SUBROUTINES ATEIG
C         & HSBG ARE NOT NEEDED. THE USER CAN OBVIATE UN-
C         NECESSARY DECKS BY SUPPLYING HIS OWN DUMMY SUB-
C         ROUTINES. AS AN EXAMPLE, GMRANK IS SHOWN BELOW.
C                SUBROUTINE GMRANK (D,IA,KL,TOLER,K)
C                IA = IA
C                RETURN
C                END
C         THE REQUIRED HEADINGS FOR ATEIG AND HSBG ARE:
C                SUBROUTINE ATEIG (IA,D,V,F,LL,IA)
C                SUBROUTINE HSBG (IA,D,IA) .
C         7.  CHANGING FIX WILL CHANGE THE INITIAL GUESS.
C         VARYING FIX FROM 0.1 TO 4.0 HAS BEEN FOUND TO
C         YIELD SATISFACTORY INITIAL GUESSES,THUS REMOVING
C         ERROR FLAGS IER = 3 AND 7, IN MOST CASES.
C         8.  THE ITERATION SCHEME SHOULD CONVERGE TO A
C         POSITIVE DEFINITE SOLUTION. THE USER CAN HAVE
C         THIS PROPERTY CHECKED (IER=0,2,4,6). FOR THIS
C         CHECK THE SOLUTION IS FACTORED BY THE CHOLESKY
C         SQUARE-ROOT METHOD. A NONPOSITIVE DIAGONAL ELE-
C         MENT OF THE FACTORED MATRIX CONSTITUTES A NON-
C         POSITIVE DEFINITE SOLUTION, RICATS RETURNS WITH
C         IER = 4 OR 5. HIGH ORDER ( IA>7 ) SOLUTIONS WITH
C         ERROR FLAGS IER = 4 OR 5 MATHEMATICALLY MAY BE
C         POSITIVE DEFINITE. TO CHECK SOLUTIONS; USE PRIOR
C                                                 -1
C         EXPERIENCE; FIND EIGENVALUES OF A - B*R   *B'*P,
C         ALL SHOULD HAVE NEGATIVE REAL PARTS;OR CALCULATE
C         Z'*P*Z FOR ARBITRARY IA BY 1 VECTOR Z, RESULTS
C         SHOULD BE POSITIVE AND FINITE.
C         9.  IF ANY OF THE 3 CHECKS FAIL (SEE NOTES 4,5,
C         7,8) THE USER SHOULD BYPASS THE FAILED CHECK AND
C         ATTEMPT TO FORCE A SOLUTION. BEFORE BYPASSING
C         THE S.C.M. CHECK, FIX SHOULD BE VARIED FROM 0.1
C         TO 4 AS SUGGESTED IN NOTE 7.
C         10. THE STEADY-STATE MATRIX RICCATI EQN. ARISES
C         FROM A SYSTEM DEFINED BY THE DIFFERENTIAL EQN.
C         .
C         X(T) = A*X(T) + B*U(T) , WHERE U(T) IS CHOSEN
C         TO MINIMIZE THE QUADRATIC COST FUNCTIONAL J;
C         J = INTEGRAL FROM ZERO TIME TO INFINITE TIME OF
C         ( X'(T)*C'*C*X(T) + U'(T)*R*U(T) )DT. U(OPTIMAL)
C                                         -1
C         IS GIVEN BY U(OPTIMAL) = - R   *B'*P*X(T).
C
C      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED.
C         MINV                   ATEIG  SEE NOTE 6.
C         SIMQ                   HSBG   SEE NOTE 6.
C                                GMRANK SEE NOTE 6.
C
C      REFERENCES
C         1.  HEATH,D.E., "A FORTRAN PROGRAM TO SOLVE THE
C         STEADY-STATE MATRIX RICCATI EQUATION OF OPTIMAL
C         CONTROL," THESIS, NPS, JUNE 1972.
C         2. KLEINMAN,D.L., "ON AN ITERATIVE TECHNIQUE
C         FOR RICCATI EQUATION COMPUTATIONS", I.E.E.E.
C         TRANSACTIONS ON AUTOMATIC CONTROL, AC-13, FEB.
C         1968, P. 114,AND BASS,R.W., "LECTURE NOTES ON
C         CONTROL SYNTHESIS AND OPTIMIZATION", PRESENTED
C         AT NASA/LARC, AUGUST 1967.
C
C    ..................................................
C
C
```

```
      SUBROUTINE RICATS (A,B,Q,R,P,D,E,F,V,L,LL,MI,IA,JB,KQ,
     1                   NTRY,TOLER,FIX,EIGMAX,IER,NN)
      DIMENSION A(1),B(1),R(1),Q(1),V(1),E(1),P(1),F(1),
     1          D(1),L(1),LL(1),MI(1)
      IF (    JB.LT.1 .OR.    JB.GT.IA   ) GO TO 6
      IF (   IER.LT.0 .OR.   IER.GT.7    ) GO TO 6
      IF ( NTRY.LT.2 .OR. NTRY.GT.150 ) GO TO 6
      GO TO 15
    6 IER = 6
      NN  = 0
      RETURN
   15 CONTINUE
      DIVCK  = 0.1E-30
      NN     = 0
      IZ     = (IA+1)/2
      IAI    = IA + 1
      IA2 = IA + 2
      MM     = IA*(IA+1)/2
      IAS    = IA*IA
      MMP = MM*MM
      EIGMAX = - ABS(EIGMAX)
      IF ( IA.EQ.1 ) GO TO 805
      IF ( IER.GT.3 ) GO TO 375
C
C         IF CONTROLLABILITY CHECK REQUESTED, FORM AUGMENTED
C    CONTROLLABILITY MATRIX AND DETERMINE ITS RANK.
      KL = IA*JB
      DO 300 I=1,KL
      D(I) = B(I)
300   P(I) = B(I)
      IR = KL
      DO 320 M=2,IA
      LI = 0
      LJ = - IA
      DO 310 J=1,JB
      LJ = LJ + IA
      DO 310 I=1,IA
      KI = I - IA
      KJ = LJ
      LI = LI + 1
      F(LI) = 0.0E0
      DO 310 K=1,IA
      KI = KI + IA
      KJ = KJ + 1
310   F(LI) = F(LI) + A(KI)*P(KJ)
      DO 320 I=1,KL
      IR = IR + 1
      D(IR) = F(I)
320   P(I) = F(I)
      CALL GMRANK (D,IA,KL,TOLER,K)
      IF ( K.EQ.IA ) GO TO 375
      NN   = K
    2 IER = 2
      RETURN
C                END CHECK ROUTINE
C
  375 IF ( KQ.EQ.0 ) GO TO 35
      DO 20 I=1,MM
20    D(I) = Q(I)
      IR = 0
      KJ = 1
      DO 30 J=1,IA
      KJ = KJ + J - 1
      KI = KJ
      DO 30 I=J,IA
      KI = KI + I - 1
      IR = IR + 1
30    Q(IR) = D(KI)
```

```
C          Q = (LOWER TRIANGULAR) Q
C
   35 IR = 0
      KL = -2*IA
      KJ = 0
      DO 40 J=1,IA
      KL = KL + IAI
      KI = KL
      KJ = KJ + J - 1
      DO 40 I=J,IA
      IR = IR + 1
      KI = KI + IA
      L(KI) = IR
40    L(IR+KJ) = IR
      IF ( JB.GT.1 ) GO TO 45
      D(1) = 1.0E0/R(1)
      GO TO 55
   45 KL = JB*JB
      DO 50 I=1,KL
50    D(I) = R(I)
      CALL  MINV (D,JB,DF,LL,MI)
   55 IR = 0
      DO 60 J=1,IA
      LJ = J - IA
      DO 60 I=J,IA
      LI = I - IA
      KJ = LJ
      IR = IR + 1
      E(IR) = 0.0E0
      KL = 0
      DO 60 M=1,JB
      KI = LI
      KJ = KJ + IA
      DO 60 K=1,JB
      KI = KI + IA
      KL = KL + 1
60    E(IR) = E(IR) - B(KI)*D(KL)*B(KJ)*1.0D0
C          E = (LOWER TRIANGULAR) - B*R(INVERSE)*B'
C
      IR = 0
      DO 70 J=1,IA
      LJ = J - IA
      DO 70 I=J,IA
      KI = I - IA
      KJ = LJ
      IR = IR + 1
      F(IR) = 0.0E0
      DO 70 K=1,JB
      KI = KI + IA
      KJ = KJ + IA
70    F(IR) = F(IR) + B(KI)*B(KJ)*1.0D0
      DO 80 I=1,MM
80    F(I) = 2.0E0*F(I)
      IR = 0
      V(1) = 0.0E0
      SAVE = 0.0E0
      DO 90 J=1,IA
      KI = J - IA
      IF ( V(1).LT.SAVE ) V(1) = SAVE
      SAVE = 0.0E0
      DO 90 I=1,IA
      KI = KI + IA
      IR = IR + 1
      P(IR) = A(KI)
90    SAVE = SAVE +  ABS(A(KI))
      IF ( V(1).LT.SAVE ) V(1) = SAVE
      SAVE = FIX*V(1)* SQRT( FLOAT(IA))
C
C SAVE = FIX*SQRT(IA)*(MAX OVER I OF SUM OVER J ABS(A(I,J)))
C
```

```
      KI = - IA
      DO 100 I=1,IA
      KI = KI + IAI
100   P(KI) = P(KI) + SAVE
C                         SOLVE
C          (A+SAVE*I)*S + S*(A+SAVE*I)' = 2*B*B'
C          INITIAL GUESS V(0) = B'*S(INVERSE)
C
      CALL MLIAPS (P,D,L,F,IA,MM,IS,MMP,IA2)
      IF ( IS.NE.0 ) NN = 1
      DO 110 I=1,IAS
110   P(I) = F(L(I))
      CALL  MINV (P,IA,DF,LL,MI)
      IR = IAS
      LJ = - IA
      DO 120 J=1,IA
      LJ = LJ + IA
      KI = 0
      DO 120 I=1,JB
      KJ = LJ
      IR = IR + 1
      D(IR) = 0.0E0
      DO 120 K=1,IA
      KI = KI + 1
      KJ = KJ + 1
120   D(IR) = D(IR) + B(KI)*P(KJ)*1.0D0
      IR = 0
      LJ = IAS - JB
      DO 130 J=1,IA
      LJ = LJ + JB
      DO 130 I=1,IA
      KI = I - IA
      KJ = LJ
      IR = IR + 1
      P(IR) = A(IR)
      DO 130 K=1,JB
      KI = KI + IA
      KJ = KJ + 1
130   P(IR) = P(IR) - B(KI)*D(KJ)*1.0D0
      IF ( IER.GT.1 .AND. IER.LT.4 .OR. IER.GT.5 ) GO TO 525
C
C         IF INITIAL GUESS STABILITY CHECK REQUESTED, FORM
C    CONTROL MATRIX AND CHECK ITS EIGENVALUES.
      KI = - IA
      V(2) = 0.0E0
      DO 500 I=1,IA
      KI = KI + IAI
500   V(2) = V(2) + P(KI)
      IF ( V(2).GT.IA*EIGMAX ) GO TO 3
      DO 510 I=1,IAS
510   D(I) = P(I)
      CALL  HSBG  (IA,D,IA)
      CALL  ATEIG (IA,D,V,F,LL,IA)
      DO 520 I=1,IA
      IF ( V(I).GT.EIGMAX ) GO TO 3
520   CONTINUE
      GO TO 525
    3 IER = 3
      KL  = 3
      GO TO 275
C              END CHECK ROUTINE
C
  525 CONTINUE
      DO 140 I=1,MM
140   Q(I) = -Q(I)
```

```
      IR = 0
      LJ = IAS - JB
      DO 150 J=1,IA
      LJ = LJ + JB
      DO 150 I=J,IA
      LI = (I-1)*JB + IAS
      KJ = LJ
      IR = IR + 1
      F(IR) = Q(IR)
      KL = 0
      DO 150 M=1,JB
      KI = LI
      KJ = KJ + 1
      DO 150 K=1,JB
      KI = KI + 1
      KL = KL + 1
150   F(IR) = F(IR) - D(KI)*R(KL)*D(KJ)*1.0D0
C                         SOLVE
C   (A-B*V(0))'*V(1) + V(1)*(A-B*V(0)) = - Q - V(0)*R*V(0)
C
      CALL MLIAPS (P,D,L,F,IA,MM,IS,MMP,IA2)
      IF ( IS.NE.0 ) NN = NN + 1
      DO 160 I=1,MM
      IF ( F(I).EQ.0.0E0 ) F(I) = DIVCK
      IF (  ABS(F(I)).LT.DIVCK ) F(I) =  SIGN(DIVCK,F(I))
160   V(I) = F(I)
      DO 255 M=2,NTRY
      KL = 0
      IR = 0
      LJ = - IA
      DO 210 J=1,IA
      LJ = LJ + IA
      KI = 0
      DO 210 I=1,IA
      KJ = LJ
      IR = IR + 1
      P(IR) = 0.0E0
      DO 210 K=1,IA
      KI = KI + 1
      KJ = KJ + 1
210   P(IR) = P(IR) + E(L(KI))*V(L(KJ))*1.0D0
      IR = 0
      LJ = - IA
      DO 220 J=1,IA
      LJ = LJ + IA
      DO 220 I=J,IA
      KI = (I-1)*IA
      KJ = LJ
      IR = IR + 1
      F(IR) = Q(IR)
      DO 220 K=1,IA
      KJ = KJ + 1
      KI = KI + 1
220   F(IR) = F(IR) + V(L(KI))*P(KJ)*1.0D0
      DO 230 I=1,IAS
230   P(I) = A(I) + P(I)
C                         SOLVE
C (A+E*V(M))'*V(M+1) + V(M+1)*(A+E*V(M)) = - Q + V(M)*E*V(M)
C
      CALL MLIAPS (P,D,L,F,IA,MM,IS,MMP,IA2)
      IF ( IS.EQ.0 ) GO TO 235
      NN = NN + 1
      IF ( NN.GT.IZ ) GO TO 7
  235 DO 240 I=1,MM
      IF ( F(I).EQ.0.0E0 ) F(I) = DIVCK
      IF (  ABS(F(I)).LT.DIVCK ) F(I) =  SIGN(DIVCK,F(I))
      IF (  ABS(1.0E0 - F(I)/V(I)).GT.TOLER ) GO TO 245
240   V(I) = F(I)
      GO TO 265
```

```
  245 DO 250 J=I,MM
      IF ( F(J).EQ.0.0E0 ) F(J) = DIVCK
      IF (  ABS(F(J)).LT.DIVCK ) F(J) =  SIGN(DIVCK,F(J))
250   V(J) = F(J)
  255 CONTINUE
      KL = 1
  265 CONTINUE
      DO 270 I=1,MM
270   Q(I) = -Q(I)
  275 IF ( KQ.EQ.0 ) GO TO 295
      DO 280 I=1,MM
280   D(I) = Q(I)
      IR = 0
      KJ = 1
      DO 290 J=1,IA
      KJ = KJ + J - 1
      KI = KJ
      DO 290 I=J,IA
      KI = KI + I - 1
      IR = IR + 1
290   Q(KI) = D(IR)
  295 CONTINUE
      DO 999 I=1,IAS
999   P(I) = V(L(I))
      IF ( IER-2*(IER/2).NE.0 ) GO TO 645
C
C        IF POSITIVE-DEFINITE CHECK REQUESTED, ATTEMPT TO
C    FACTOR THE SOLUTION BY CHOLESKY SQUARE ROOT METHOD.
      IF ( P(1).LT.DIVCK ) GO TO 4
      IF ( IA.GT.2 ) GO TO 605
      IF ( P(4)-P(3)*P(3)/P(1).LT.DIVCK ) GO TO 4
      GO TO 645
  605 IR = 1
      SAVE =  SQRT(P(1))
      DO 610 J=2,IA
      IR = IR + IA
610   D(IR) = P(IR)/SAVE
      IR = 1
      I = 2
  615 IR = IR + IAI
      SAVE = P(IR)
      LI = IR - I
      DO 620 K=2,I
      LI = LI + 1
620   SAVE = SAVE - D(LI)*D(LI)
      IF ( SAVE.LT.DIVCK ) GO TO 4
      IF ( I.GE.IA ) GO TO 645
      SAVE =  SQRT(SAVE)
      LI = IR - I
      LJ = LI
      IZ = IR
      M  = I + 1
      DO 640 J=M,IA
      KI = LI
      LJ = LJ + IA
      KJ = LJ
      IZ = IZ + IA
      D(IZ) = P(IZ)
      DO 630 K=2,I
      KI = KI + 1
      KJ = KJ + 1
630   D(IZ) = D(IZ) - D(KI)*D(KJ)*1.0D0
640   D(IZ) = D(IZ)/SAVE
      I = I + 1
      GO TO 615
    4 IER = 4 + KL
      RETURN
C             END CHECK ROUTINE
C
```

```
645 IER = KL
    RETURN
  7 IER = 7
    KL = 7
    GO TO 265
C
C     FOR IA = 1, SOLVE THE RESULTING QUADRATIC EQUATION.
805 KL = 0
    IF ( IER.LT.4 .AND.  ABS(B(1)).LT.DIVCK ) GO TO 2
    D(1) = R(1)/B(1)*B(1)
    D(2) = A(1)*D(1)
    P(1) = D(2) + SQRT( D(2)*D(2) + Q(1)/D(1) )
    IF ( IER-2*(IER/2).EQ.0 .AND. P(1).LT.0.0E0 ) GO TO 4
    IER = KL
    RETURN
    END
```

```
      SUBROUTINE MLIAPS (P,D,L,F,IA,MM,IS,MMP,IA2)
      DIMENSION P(1),D(1),L(1),F(1)
      DO 5 I=1,MMP
5     D(I) = 0.0E0
      IR = 0
      DO 10 I=1,IA
      LI = I - IA
      DO 10 J=1,IA
      KJ = J - IA
      KI = LI
      IR = IR + 1
      DO 10 K=1,IA
      KI = KI + IA
      KJ = KJ + IA
      LJ = L(KI) + (L(KJ)-1)*MM
10    D(LJ) = D(LJ) + P(IR)
      KI = - IA
      DO 15 I=1,IA
      KI = KI + IA2 - I
      KJ = KI - MM
      DO 15 J=1,MM
      KJ = KJ + MM
15    D(KJ) = 2.0E0*D(KJ)
      CALL SIMQ (D,F,MM,IS)
      RETURN
      END
```

```
C
C     ..................................................................
C
C        SUBROUTINE GMRANK
C
C        PURPOSE
C           TO DETERMINE THE NUMERICAL ROW (COLUMN) RANK OF
C           A GENERAL MATRIX.
C
C        USAGE
C           CALL GMRANK (D,IA,KL,TOLER,K)
C
C        DESCRIPTION OF PARAMETERS
C           D      - GENERAL INPUT MATRIX (DESTROYED).
C           IA     - ROW DIMENSION OF MATRIX D.
C           KL     - COLUMN DIMENSION OF MATRIX D.
C           TOLER  - INPUT CONSTANT USED AS A RELATIVE
C                    TOLERANCE FOR LOSS OF SIGNIFICANCE.
C           K      - ON RETURN, K = RANK OF MATRIX D.
C
C        REMARKS
C           NONE
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        REFERENCES
C           PENNINGTON,RALPH H., "INTRODUCTORY COMPUTER
C           METHODS AND NUMERICAL ANALYSIS", PUBLISHED BY
C           MACMILLAN COMPANY, NEW YORK, 1965.
C
C     ..................................................................
C

      SUBROUTINE GMRANK (D,IA,KL,TOLER,K)
      DIMENSION D(1)
      IAI = IA + 1
      NN = MINO(IA,KL)
      IR = - IA
      K = 1
  325 CONTINUE
      IR = IR + IAI
      SAVE =  ABS(D(IR))
      LI = K
      LJ = K
      KJ = K - 1
      KI = IR - K
      DO 330 J=K,KL
      KI = KI + KJ
      DO 330 I=K,IA
      KI = KI + 1
      IF (  ABS(D(KI)).LE.SAVE ) GO TO 330
      LI = I
      LJ = J
      SAVE =  ABS(D(KI))
330   CONTINUE
      IF ( SAVE.LT.0.1E-35 ) GO TO 365
      IF ( K.GE.NN ) GO TO 375
      IF ( LI.LE.K ) GO TO 345
      KI = IR - IA
      KJ = KI + LI - K
      DO 340 J=K,KL
      KJ = KJ + IA
      KI = KI + IA
      SAVE  = D(KI)
      D(KI) = D(KJ)
340   D(KJ) = SAVE
  345 CONTINUE
      IF ( LJ.LE.K ) GO TO 355
```

```
      KI = IR - 1
      KJ = KI + (LJ-K)*IA
      DO 350 I=K,IA
      KI = KI + 1
      KJ = KJ + 1
      SAVE  = D(KI)
      D(KI) = D(KJ)
350   D(KJ) = SAVE
  355 CONTINUE
      LI = K + 1
      LJ = IR
      KJ = IR + IA - K
      DO 360 J=LI,KL
      LJ = LJ + IA
      KI = IR
      KJ = KJ + K
      D(LJ) = D(LJ)/D(IR)
      DO 360 I=LI,IA
      KI = KI + 1
      KJ = KJ + 1
      SAVE  = D(LJ)*D(KI)
      D(KJ) = D(KJ) - SAVE
      IF (  ABS(D(KJ)).GE.TOLER* ABS(SAVE) ) GO TO 360
      D(KJ) = 0.0E0
360   CONTINUE
      K = K + 1
      GO TO 325
  365 CONTINUE
      K = K - 1
  375 CONTINUE
      RETURN
      END
```

LIST OF REFERENCES

1. Athans, M. and Falb, P.F., Optimal Control, p. 771-776, New York: McGraw-Hill, 1966.

2. Kleinman, D.L., "On an Iterative Technique for Riccati Equation Computations," IEEE Transactions on Automatic Control, vol. 13, p. 114-115, February 1968.

3. M.I.T. Electronic Systems Lab., Cambridge, Mass., Report 297, Suboptimal Design of Linear Regulator Systems Subject to Computer Storage Limitations, by D.L. Kleinman, 1967.

4. Marcus, M. and Ming, H., A Survey of Matrix Theory and Matrix Inequalities, p. 69, Allyn and Bacon, Inc., 1964.

5. Kantorovich, L.V. and Akilov, G.P., Functional Analysis in Normed Spaces, p. 189, New York: MacMillan, 1964.

6. Bass, R.W., Lecture Notes on Control Synthesis and Optimization, Presented at NASA/LaRC, August 1961.

7. Wonham, W.M. and Cashman, W.F., A Computational Approach to Optimal Control of Stochastic Saturating Systems, Preprints of Technical Papers of the Ninth Joint Automatic Control Conference, p. 13, 26 June 1968.

8. Aeronca Manufacturing Corporation, Air Force Office of Scientific Research 2754, Aspects of General Control Theory, by R.W. Bass and P. Mendelson, p. 72-74, August 1962.

9. Smith, P.G., "Numerical Solution of the Matrix Equation AX + XA' + B = 0," IEEE Transactions on Automatic Control, vol. 16, p. 278, June 1971.

10. Bingulac, S.P., "An Alternate Approach to Expanding PA + A'P = -Q," IEEE Transactions on Automatic Control, vol. AC-15, p. 135, February 1970.

11. Douglas Paper No. 4538, Machine Solution of High Order Matrix Riccati Equations, by R.W. Bass, 1967.

12. MacFarlane, A.G.J., "An Eigenvector Solution of the Optimal Linear Regulator Problem," Journal of Electronics and Control, vol. 14, no. 6, p. 643-654, June 1963.

13. Blackburn, T.R., Solution of the Algebraic Matrix Riccati Equation via Newton-Raphson Iteration, Preprints of technical papers on the Ninth Joint Automatic Control Conference, p. 940, 26 June 1968.

14. Pennington, R.H., Introductory Computer Methods and Numerical Analysis, p. 331-333, New York: McGraw-Hill, 1965.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Asst. Professor Ron A. Hess, Code 57 He Department of Aeronautics Naval Postgraduate School Monterey, California 93940	5
4.	ENS Donald Edward Heath, USN 1131 Hillridge Road Reynoldsburg, Ohio 43068	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

A Fortran Program to Solve the Steady-State Matrix Riccati Equation of Optimal Control.

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis (June 1972)

5. AUTHOR(S) (First name, middle initial, last name)

Donald Edward Heath

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1972	42	14.

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

This thesis presents a Fortran program that numerically solves the steady-state matrix Riccati equation of the quadratic cost optimal control problem. Each step of the program is presented, analytically and computationally. The check points incorporated in the program and the input parameters that can be used to assure a correct solution are identified and discussed. Difficulties encountered when verifying the program, and the suggested solutions, are also presented.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Riccati Equation						
Steady-State Riccati Equation						
Lyapunov Equation						