LINEAR QUADRATIC CONTROL PACKAGE
PART I - THE CONTINUOUS PROBLEM

K. MÅRTENSSON

REPORT 6802 APRIL 1 1968
LUND INSTITUTE OF TECHNOLOGY
DIVISION OF AUTOMATIC CONTROL

LINEAR QUADRATIC CONTROL PACKAGE
PART I - THE CONTINUOUS PROBLEM

K. Mårtensson

Abstract

The numerical solution of the linear quadratic control problem is discussed. Two algorithms based on the Euler-Lagrange and the Hamilton-Jacobi approach are presented. The relative merits of the approaches are discussed. Complete FORTRAN programs for the algorithms are presented. The programs can be used to design optimal multivariable control systems and to compute optimal filters and predictors. The work has been carried out with the support of the Swedish Technical Research Council.

Report 6802 April 1 1968
Lund Institute of Technology
Division of Automatic Control

1. INTRODUCTION

In this report we discuss numerical methods to solve the so called linear quadratic control problem and present two complete program packages for the solution of the problem. All the programming is done in FORTRAN.

There are many control problems which favourably can be formulated as linear quadratic control problems, e.g. steady state control of multivariable industrial processes, optimal filtering and prediction etc.

The problem, which is well-known from the calculus of variations, can be approached in two different ways. The method associated with the names Euler-Lagrange-Pontryagin reduces the problem to a two point boundary value problem for a system of ordinary differential equations, while the Hamilton-Jacobi-Bellman method gives an initial value problem for a partial differential equation. In the special case of time-invariant systems, the two point boundary value problem can be reduced to an initial value problem, and the partial differential equation can be reduced to an ordinary non-linear differential equation, the Riccati equation.

The two different approaches to the variational problem thus immediately suggest two different algorithms. These are presented in the report, and the advantages and disadvantages are discussed and compared.

The statement of the problem is given in section 2, and the solution to the problem following the two different approaches is given in section 3. In section 4 we present and describe the routines required for the numerical solution. Section 5 contains some examples for test purpose. The listing of the programs and some typical outputs are given in the appendices.

2. STATEMENT OF THE PROBLEM

Consider a linear time-varying dynamical system given by the equation

$$\frac{dx(t)}{dt} = A(t)\ x(t) + B(t)\ u(t) \tag{2.1}$$

where the state x(t) is a vector of dimension n, the control input u(t) a vector of dimension r, A(t) an n x n matrix and B(t) an n x r matrix. A(t) and B(t) are assumed to be piecewise continuous, that is $a_{ij}(t)$ and $b_{ij}(t)$ are piecewise continuous. We now specify the objective of the system in the following way. Let t_o and t_1 be given timepoints. Form the so called cost functional

$$V(u) = \frac{1}{2}\{x^T(t_1)\cdot Q_o x(t_1)\} + \frac{1}{2}\int_{t_o}^{t_1}\{x^T(s)\cdot Q_1(s)x(s)+u^T(s)Q_2(s)u(s)\}ds \tag{2.2}$$

where we assume Q_o and $Q_1(s)$ to be symmetric nonnegative definite matrices, and $Q_2(s)$ a symmetric positive definite matrix. We also postulate that $Q_1(s)$ and $Q_2(s)$ are piecewise continuous.

The object is to determine a control signal for the system (2.1) so that the cost functional (2.2) becomes as small as possible.

We will assume that there are no constraints on the magnitude of the control vector u or the state vector x. From a physical point of view this may seem to be a rather unrealistic assumption, because we always have in physical systems some kind of restrictions on the system variables. The reason is that we under these assumptions are able to get an analytic expression for the control signal, and that this control signal will become a linear function of the state variables. We can then realize the optimal system by a linear time-varying feedback. (Thus it is not impossible that the control signal in some cases grows very large. However, by using the quadratic criteria (2.2), large signals are much more punished than small ones.) When solving the problem we will refer to well-known results from the calculus of variations, and we shall compare two possible methods to solve the problem, namely:

A. Euler-Lagrange Method

B. Hamilton-Jacobi Method

3. SOLUTION TO THE LINEAR QUADRATIC PROBLEM

A. Euler-Lagrange Method.

The method associated with the names Euler-Lagrange characterizes the optimal trajectory by examination of the variation of the loss functional in a small neighbourhood of the optimal trajectory. The method leads to a two point boundary value problem, and the solution is primarily obtained only for the initial state considered. We form the Hamiltonian

$$2\mathcal{H}(x,p,u) = x^TQ_1x + u^TQ_2u + 2p^T(Ax + Bu) \qquad (3.1)$$

The Hamiltonian shall be minimized with respect to u, and this is most easily done by completing the square.

$$2\mathcal{H}(x,p,u) = x^TQ_1x + 2p^TAx + \{u + Q_2{}^{-1}B^Tp\}^T\, Q_2\{u + Q_2{}^{-1}B^Tp\} - $$
$$- p^TB\, Q_2{}^{-1}\, B^Tp \qquad (3.2)$$

We then get the stationary value

$$2\mathcal{H}^o(x,p) = x^TQ_1x + 2p^TA_x - p^TBQ_2{}^{-1}\, B^Tp \qquad (3.3)$$

for

$$u = - Q_2{}^{-1}\, B^Tp \qquad (3.4)$$

Notice the importance of the assumption that Q_2 is positive definite. If Q_2 is only nonnegative definite, the control u that minimizes the Hamiltonian (3.1) is not unique.

The canonical equations are

$$\frac{dx}{dt} = \mathcal{H}^{o}_{p} = Ax - B\, Q_2^{-1}\, B^T p \tag{3.5}$$

$$\frac{dp}{dt} = -\mathcal{H}^{o}_{x} = -\, Q_1 x - A^T p \tag{3.6}$$

with the boundary conditions

$$x(t_o) = a \tag{3.7}$$

$$p(t_1) = Q_o \cdot x(t_1) \tag{3.8}$$

Notice that this is a two point boundary value problem for a system of 2n differential equations. n boundary values are given at time t_o, and n at time t_1. The solution is given only for the special initial state $x(t_o) = a$.

However, the linearity of the equations (3.5) and (3.6) makes it possible to reduce the two point boundary problem to an initial value problem. Introduce the 2n x 2n matrix $\Sigma(t;t_1)$ being a fundamental matrix to the canonical equations (3.5) and (3.6). Hence

$$\frac{d}{dt}\,\Sigma(t;t_1) = \begin{pmatrix} A & -BQ_2^{-1}\,B^T \\ -Q_1 & -A^T \end{pmatrix} \Sigma(t;t_1) \tag{3.9}$$

and

$$\Sigma(t_1;t_1) = I \tag{3.10}$$

where I is the 2n x 2n identity matrix.

Partition the 2n x 2n matrix $\Sigma(t;t_1)$ into four n x n submatrices in the following way

$$\Sigma(t;t_1) = \begin{pmatrix} \Sigma_{11}(t;t_1) & \Sigma_{12}(t;t_1) \\ \Sigma_{21}(t;t_1) & \Sigma_{22}(t;t_1) \end{pmatrix} \tag{3.11}$$

The solutions to the canonical equations (3.5) and (3.6) can then be written

$$x(t) = \Sigma_{11}(t;t_1)b + \Sigma_{12}(t;t_1)\ Q_o b \tag{3.12}$$

$$p(t) = \Sigma_{21}(t;t_1)b + \Sigma_{22}(t;t_1)\ Q_o b \tag{3.13}$$

where $b = x(t_1)$. From (3.7) we have $x(t_o) = a$, and from (3.12) we then get

$$b = (\Sigma_{11}(t_o;t_1) + \Sigma_{12}(t_o;t_1)\ Q_o)^{-1}a \tag{3.14}$$

provided that

$$\det(\Sigma_{11}(t_o;t_1) + \Sigma_{12}(t_o;t_1)\ Q_o) \neq 0 \tag{3.15}$$

This condition is obviously satisfied if the time difference $t_1 - t_o$ is small enough, because

$$\Sigma_{11}(t_1;t_1) = I \tag{3.16}$$

$$\Sigma_{12}(t_1;t_1) = 0 \tag{3.17}$$

and the matrix $\Sigma(t;t_1)$ is continuous because we assume the matrices A, B, Q_1, Q_2 to be piecewise continuous. It can be shown (with some effort) that the inverse always exists with the assumptions made about A, B, Q_o, Q_1 and Q_2.

Notice that the control signal u given by (3.4) is a function of time. Physically this corresponds to an open loop control of the system (2.1), and is therefore less attractive from the control point of view. However, the linearity of the canonical equations makes it possible to obtain a feedback solution. Equation (3.4) gives the value of the control signal at time t, u(t), as a function of the adjoint variable p at time t. p(t) can with (3.12) and (3.13) be expressed as a linear function of the state variables x(t). We then get

$$p(t) = S(t)\ x(t) \tag{3.18}$$

where

$$S(t) = \{\Sigma_{21}(t;t_1) + \Sigma_{22}(t;t_1)Q_o\}\{\Sigma_{11}(t;t_1) + \Sigma_{12}(t;t_1)Q_o\}^{-1} \tag{3.19}$$

It can be shown that S is always symmetric.
The control variable is now given as

$$u(t) = - Q_2^{-1}(t)\ B^T(t)\ S(t)\ x(t) \tag{3.20}$$

or

$$u(t) = - L(t)\ x(t) \tag{3.21}$$

We thus have got a feedback solution, in which the gain coefficients are functions of the time. Furthermore, the solution is no longer restricted to a special value of the initial state $x(t_o)$.

B. Hamilton-Jacobi Method.

Typical for this method is that the problem is embedded in a suite of problems, and the solution does not depend on the special initial state. Besides we will directly get the feedback solution. Introduce the functional

$$V(x,t) = \min_u \{ \frac{1}{2} x^T(t_1)Q_o x(t_1) + \frac{1}{2} \int_{t_o}^{t_1} \{x^T(s)Q_1(s)x(s) + \\ + u^T(s)Q_2(s)u(s)\}\ ds\} \tag{3.22}$$

and form the same Hamiltonian as in A

$$2\mathcal{H}(x,p,u) = x^T Q_1 x + u^T Q_2 u + 2p^T(Ax + Bu) = x^T Q_1 x + 2p^T Ax - \\ - p^T B Q_2^{-1} B^T p + (u + Q_2^{-1}B^T p)^T Q_2 (u + Q_2^{-1}B^T p) \tag{3.23}$$

The minimum of the Hamiltonian with respect to u is

$$2\mathcal{H}^o(x,p) = x^T Q_1 x + 2p^T Ax - p^T B Q_2^{-1} B^T p \tag{3.24}$$

for

$$u = - Q_2^{-1} B^T p \tag{3.25}$$

The functional V(x,t) must satisfy the Hamilton-Jacobi partial differential equation

$$V_t + \mathcal{H}^{\circ}(x, V_x^T) = 0 \qquad (3.26)$$

where $V_t = \frac{\partial V}{\partial t}$ and $V_x = \text{grad}_x V$. From (3.24) we then have

$$2V_t + x^T Q_1 x + 2V_x Ax - V_x BQ_2^{-1}B^T V_x^T = 0 \qquad (3.27)$$

(3.22) provides the boundary condition

$$V(x,t_1) = \frac{1}{2} x^T(t_1)\, Q_o x(t_1) \qquad (3.28)$$

To solve the partial differential equation (3.27), the boundary condition (3.28) suggests that we make the approach

$$V(x,t) = \frac{1}{2} x^T(t)\, S(t)\, x(t) \qquad (3.29)$$

where S is an n x n matrix. There is no loss in generality if we assume S symmetric and nonnegative definite. (3.29) is then a solution to (3.27) if S satisfies the non-linear differential equation

$$\frac{dS}{dt} + A^T S + S A - S^T BQ_2^{-1}B^T S + Q_1 = 0 \qquad (3.30)$$

with boundary conditions given at the terminal time t_1

$$S(t_1) = Q_o \qquad (3.31)$$

The matrix differential equation (3.30) contains n^2 non-linear first order differential equations, and is the so called Riccati equation. With the assumptions made about the matrices A, B, Q_o, Q_1 and Q_2, the solution to the Riccati equation always exists and is unique.

As S is a symmetric matrix, the number of differential equations to solve is then reduced from n^2 to $n(n + 1)/2$. Notice that these equations are solved backwards in time since the boundary conditions are given at the terminal time t_1.

In the minimum Hamiltonian (3.24) we have replaced the adjoint variable p with the gradient V_x^T. From (3.29) we have

$$V_x^T = Sx \tag{3.32}$$

and hence the optimal control law (3.25) becomes

$$u = - Q_2^{-1} B^T Sx \tag{3.33}$$

The value of the control variable u at time t, u(t), is thus given by

$$u(t) = - Q_2^{-1}(t) B^T(t) S(t) x(t) \tag{3.34}$$

or

$$u(t) = - L(t) x(t) \tag{3.35}$$

Notice that the Hamilton-Jacobi method directly gives the feedback control law (with time-varying parameters), and that the solution is independent of the initial state $x(t_o)$. If we compare the results above with the results we get from Euler-Lagrange method, we can see that they are the same, which obviously depends on the fact that the system is linear and the loss functional quadratic. However, the two methods suggest two possible different ways to solve the problem. By differentiating (3.19) with respect to time, it can be shown that the matrix S given by (3.19) satisfies the Riccati equation (3.30) with the boundary condition (3.31).

4. NUMERICAL SOLUTION TO THE LINEAR QUADRATIC PROBLEM

In the preceding section we have shown two different ways to solve the linear-quadratic problem, Euler-Lagrange and Hamilton-Jacobi. In this section we will present two algorithms directly based on the methods described in section 3. Thus we will be able to compare the methods from a computational point of view. We will also present the complete program packages solving the problem for time-invariant matrices A, B, Q_1 and Q_2. All the programming is done in FORTRAN (CDC-3600 FORTRAN). The results and comparisons of the methods are presented in section 5.

A. Numerical solution with Euler-Lagrange method.

From the fundamental matrix

$$\Sigma(t;t_1) = \begin{pmatrix} \Sigma_{11}(t;t_1) & \Sigma_{12}(t;t_1) \\ \Sigma_{21}(t;t_1) & \Sigma_{22}(t;t_1) \end{pmatrix} \tag{4.1}$$

satisfying

$$\frac{d}{dt}\ \Sigma(t;t_1) = \begin{pmatrix} A & -BQ_2^{-1}B^T \\ -Q_1 & -A^T \end{pmatrix} \Sigma(t;t_1) \tag{4.2}$$

we have

$$S(t) = (\Sigma_{21}(t;t_1) + \Sigma_{22}(t;t_1)Q_o)(\Sigma_{11}(t;t_1) + \Sigma_{12}(t;t_1)Q_o)^{-1} \tag{4.3}$$

In the time-invariant case we can give an explicite expression for $\Sigma(t;t_1)$

$$(t;t_1) = \exp\left\{\begin{pmatrix} A & -BQ_2^{-1}B^T \\ -Q_1 & -A^T \end{pmatrix}(t-t_1)\right\} \tag{4.4}$$

where $t_o \leq t \leq t_1$. In the numerical calculations we now proceed as follows. Form the matrix

$$F = \begin{pmatrix} A & -BQ_2^{-1}B^T \\ -Q_1 & -A^T \end{pmatrix}(t-t_1) \tag{4.5}$$

Then compute the matrix function

$$G = e^F \tag{4.6}$$

where e^F is defined through its Taylor series expansion

$$e^F = I + F + \frac{F^2}{2!} + \frac{F^3}{3!} + \ldots + \frac{F^n}{n!} + \ldots \tag{4.7}$$

The series converges for all matrices F (see subroutine MEXP7T), and we then have $\Sigma(t;t_1)$. Partition Σ according to (3.11) and form the matrices

$$(\Sigma_{21}(t;t_1) + \Sigma_{22}(t;t_1)Q_o) \tag{4.8}$$

and

$$(\Sigma_{11}(t;t_1) + \Sigma_{12}(t;t_1)Q_o) \tag{4.9}$$

(4.9) is inverted and the product

$$S(t) = (\Sigma_{21}(t;t_1) + \Sigma_{22}(t;t_1)Q_o)(\Sigma_{11}(t;t_1) + \Sigma_{12}(t;t_1)Q_o)^{-1} \tag{4.10}$$

is formed.

Finally we get the feedback matrix $L(t)$ by some simple matrix multiplications. We can see that except for the matrix exponentiation, there are only very simple computations involved. Now suppose that we want to compute the feedback matrix L at t_1, $t_1 - \Delta t$, $t_1 - 2\Delta t$, ..., $t_1 - n\Delta t$. Since the system is time-invariant, the fundamental matrix $\Sigma(s - \Delta t, s)$ only depends on the time difference Δt. We then start with computing $\Sigma(t_1 - \Delta t, t_1)$, $S(t_1 - \Delta t)$ and $L(t_1 - \Delta t)$. In (4.8), (4.9) and (4.10) we then replace Q_o with the computed $S(t_1 - \Delta t)$, which is the boundary value at $t = t_1 - \Delta t$, and get $S(t_1 - 2\Delta t)$ and $L(t_1 - 2\Delta t)$. This is repeated until $t_1 - n\Delta t = t_o$. With this iteration method we can compute $\Sigma(t_1 - \Delta t, t_1)$ once for all, thereby reducing the computations involved. In the time-varying case it is not possible to use this procedure, and at the end of this section we will give a brief discussion of the consequences of time-varying parameters. The program package (appendix A) consists of:

LIOPCON	Main program
RICCE	Subroutine
MEXP7T	Subroutine
GJRV	Subroutine
NORM	Subroutine

We give here a short description of the subroutines, its parameters and the input-output required.

SUBROUTINE NORM (A, N, IA, S)

This subroutine computes the norm of an n x n matrix. A. The norm S is taken as

$$S = \min \{ \max_i \sum_{j=1}^{n} |a_{ij}|, \max_j \sum_{i=1}^{n} |a_{ij}| \} \qquad (4.11)$$

Parameters:

A-matrix

N-order of A

IA-dimension parameter

S-resulting norm.

SUBROUTINE GJRV (A, N, EPS, IERR, IA)

Inverts asymmetric matrices by the method of Gauss-Jordan with row-pivoting.

Parameters:

A-the matrix to be inverted. A is returned containing the inverse A^{-1} if the inversion has succeeded.

N-order of A.

EPS-value to be used as a tolerance for acceptance of the singularity of the matrix.

IERR-integer variable which will contain zero upon return if the inversion is completed or -1 if any pivot element has an absolute value less than EPS, in which case the matrix is considered to be singular.

IA-dimension parameter.

SUBROUTINE MEXP7T (A, B, N, IA, NOTRACE)

Computes the matrix function $B = e^A$ where A and B are n x n matrices. We define e^A by its Taylor series expansion

$$e^A = I + A + \frac{A^2}{2!} + \ldots + \frac{A^n}{n!} + \ldots \qquad (4.12)$$

It is easy to show that this series converges for all matrices A. Introduce the vector space $L_n = \{x|x$ an n x n matrix$\}$. Then L_n is a Banach-space, and the series (4.12) is convergent if it is convergent in the norm. We have

$$\left|\left| \frac{A^n}{n!} \right|\right| \leqslant \frac{||A||^n}{n!} \qquad (4.13)$$

The series

$$\sum_{n=o}^{\infty} \left|\left| \frac{A^n}{n!} \right|\right| \qquad (4.14)$$

and

$$\sum_{n=o}^{\infty} \frac{||A||^n}{n!} \qquad (4.15)$$

are ordinary positive series, and (4.15) converges for all $||A||$ towards the scalar $e^{||A||}$. From (4.13) the series (4.14) then is dominated convergent, which finally proves that

$$\sum_{n=o}^{\infty} \frac{A^n}{n!} \qquad (4.16)$$

converges for all quadratic matrices A.

Further we notice that the relation

$$e^{2A} = e^A e^A \qquad (4.17)$$

holds for all quadratic matrices A and

$$e^{A+B} = e^A e^B = e^B e^A \qquad (4.18)$$

if the matrices A and B commutate.

From (4.18) and from the fact that

$$e^{\phi} = I \qquad (4.19)$$

where ϕ is the null matrix, we get the inverse

$$(e^A)^{-1} = e^{-A} \qquad (4.20)$$

In the numerical calculations, the great problem obviously is how many terms in the expansion we must sum up to get accurate results. We will here make a very rough estimation based on the number of significant digits that can be obtained from the computer used. Suppose we estimate the error with

$$e_n = \frac{||A||^n}{n!} \geq \frac{||A^n||}{n!} \qquad (4.21)$$

From (4.17) we can see that instead of computing e^A we compute $e^{\frac{A}{2}}$ and then take the square $(e^{\frac{A}{2}})^2 = e^A$. This means a scaling of the matrix before computing the sum, but also one more matrix multiplication. With respect to the execution time, we could then as well increase the sum with one more term. Suppose that these two methods shall give the same accuracy. We then get

$$\frac{2 \cdot ||\frac{A}{2}||^{n-1}}{(n-1)!} \cdot \frac{1}{||e^{\frac{A}{2}}||} \simeq \frac{||A||^n}{n!\ ||e^A||} \qquad (4.22)$$

If $||A||$ is small enough, then $||e^A|| \simeq 1$ and (4.17) can be reduced to

$$||A|| \simeq n \cdot 2^{-n+2} \qquad (4.23)$$

We estimate $||A||$ and e_n for some different n.

n	e_n	$\|\|A\|\|$
5	$\sim 8 \times 10^{-3}$	0.625
6	$\sim 4 \times 10^{-6}$	0.375
7	$\sim 5 \times 10^{-9}$	0.219

The computer used (CDC 3600) has an accuracy of about 10 significant digits, which means that we shall sum up 7-8 terms in the series expansion. Instead of scaling A so that $||A|| < 0.2$, we have in the subroutine chosen to compute and store $\frac{A^1}{1!}, \ldots, \frac{A^7}{7!}$, and then scale A so that $||\frac{A^7}{7!}|| < 1.0 \cdot 10^{-10}$. If the norm is too large, we scale A with a factor 2^n, where n is determined so that

$$|| \frac{(\frac{A}{2^n})^7}{7!} || < 1.0 \cdot 10^{-10} \qquad (4.24)$$

After the summation where the identity matrix is added, the result is

$$B_o = e^{\frac{A}{2^n}} \qquad (4.25)$$

The matrix is squared n times

$$B_1 = B_o^{\,2}$$
$$B_2 = B_1^{\,2}$$
$$\vdots$$
$$B_n = B_{n-1}^{\,2} = B_o^{\,2^n} = e^A \qquad (4.26)$$

By scaling with an appropriate term 2^n instead of an integer m, the number of matrix multiplications required are reduced from m to n, where $2^n \sim m$. Here again we will emphasize, that the estimations done are very rough, and not in any way mathematically rigorous. However, the subroutine has given very accurate results even for some very ill-conditioned matrices. MEXP7T has an option called NOTRACE. This gives a possibility to shift the origin to the centre of the eigenvalues. The sum of the eigenvalues is equal to tr A (trace of A), and consequently the centre is tr A/n, where n is the order of A. We then have

$$A = (A - \frac{\text{tr } A}{n} \cdot I) + (\frac{\text{tr } A}{n} \cdot I) \qquad (4.27)$$

and

$$e^A = e^{(A - \frac{\text{tr } A}{n} \cdot I) + (\frac{\text{tr } A}{n} \cdot I)} \qquad (4.28)$$

But the matrices

$$A - \frac{\text{tr } A}{n} \cdot I \qquad (4.29)$$

and

$$\frac{\text{tr } A}{n} \cdot I \qquad (4.30)$$

commutate, which implies that (4.28) can be written

$$e^A = e^{(A - \frac{\text{tr } A}{n} \cdot I)} \cdot e^{\frac{\text{tr } A}{n} \cdot I} \qquad (4.31)$$

We now compute

$$B = e^{(A - \frac{\text{tr } A}{n} \cdot I)} \qquad (4.32)$$

with the method indicated above, and finally multiply each term in B with the scalar $e^{\frac{\text{tr } A}{n}}$. The origin shift obviously makes the convergence of the series faster. Suppose we have a 2 x 2 matrix A with eigenvalues 499 and 501. After shifting the origin we have a matrix with eigenvalues -1 and 1, and the series expansion will give much faster convergence towards e^{1} and e^{-1} than towards e^{501} and e^{499}. However, as we will see below, there are cases when the trace computation is not necessary which is the reason for the option.

Parameters:

A-input matrix
B-resulting matrix $B = e^{A}$
N-order of A and B
IA-dimension parameter
NOTRACE - is set zero if no trace computation is wanted, 1 if the origin shift shall be done.

SUBROUTINE RICCE (A, B, Q0, Q1, Q2, S, N, NU, IA, IB, TD, IERR)

The inputs to RICCE are the system and loss functional matrices A, B, Q0, Q1, Q2 and the time difference TD = $t_1 - t$. RICCE arranges the matrix

$$EA = \begin{pmatrix} A & -BQ_2^{-1}B^T \\ -Q_1 & -A^T \end{pmatrix} (t - t_1) \qquad (4.33)$$

and then, by calling MEXP7T, computes

$$\Sigma(t;t_1) = e^{EA} \qquad (4.34)$$

The fundamental matrix is partitioned and

$$S(t) = (\Sigma_{21}(t;t_1) + \Sigma_{22}(t;t_1)Q_o)(\Sigma_{11}(t;t_1) + \Sigma_{12}(t;t_1)Q_o)^{-1} \tag{4.35}$$

is the output matrix. From (4.33) we have tr EA = 0 for all matrices A, B, Q_1 and Q_2, and hence when computing $\Sigma(t;t_1)$ we can skip the origin shift (which is zero) in MEXP7T, thereby avoiding meaningless computations. RICCE has an entry point named ITERATE. This gives us possibility to compute the fundamental matrix $\Sigma(t_1 - \Delta t, t_1)$ once for all, and then make use of the previously mentioned iteration technique. We then proceed as follows. At the first call to RICCE we have $Q_o = S(t_1)$ thereby computing $\Sigma(t_1 - \Delta t, t_1)$ and the output matrix $S(t_1 - \Delta t)$. The fundamental matrix is carefully stored in an internal array in RICCE, and will consequently never be destroyed by computations outside the subroutine. In the main program the feedback matrix $L(t_1 - \Delta t)$ is then computed, and Q_o is set equal to $S(t_1 - \Delta t)$. When computing $S(t_1 - 2\Delta t)$ we now call ITERATE and get

$$S(t_1 - 2\Delta t) = (\Sigma_{21} + \Sigma_{22}S(t_1 - \Delta t))(\Sigma_{11} + \Sigma_{12}S(t_1 - \Delta t))^{-1} \tag{4.36}$$

using the formerly computed Σ_{ij}:s. This is repeated until $t_1 - n\Delta t = t_o$. Notice that this iteration procedure demands the matrices A, B, Q_1 and Q_2 to be time-invariant. In RICCE we have not made use of the symmetry of S, a fact that could give possibility to decrease the execution time. The reason for this is that the computations that could be saved, are very few in comparison with those we need anyway, and besides we have here a chance to get some information about the accuracy of the computed S and L matrices. If S begins to differ very much from a symmetric matrix, the results should of course be treated very suspiciously.

Parameters:

A-system matrix of order N x N.
B-system insignal matrix of order N x NU.
Q0, Q1-loss functional matrices of order N x N.
Q2-loss functional matrix of order NU x NU.
S-output matrix of order N x N.
TD-time difference $t_1 - t$.
IERR-returned -1 if any inversion has failed.
IA,IB-dimension parameters.

PROGRAM LIOPCON

This is the main program administrating inputs and outputs. It also makes the appropriate calls to RICCE or ITERATE and computes the feedback matrix L. The inputs to LIOPCON are:

N-order of the system (max 10).
NU-number of insignals (max 10).
ITIME-number of equidistant points in which S and L are computed and printed. The terminal time t_1 is not included in ITIME.
ITER-ITER=0 means that the fundamental matrix is computed at each step. ITER=1 means that it is computed only in the first step, and the entry point ITERATE in RICCE is used in the other steps.
TIMEDIF-time difference between the points (Δt).
A-system matrix of order N x N.
B-system insignal matrix of order N x NU.
Q0, Q1-loss functional matrices of order N x N.
Q2-loss functional matrix of order NU x NU.

Concerning the input formats, see appendix A. The main program LIOPCON can only handle the time-invariant case. If the matrices A, B, Q1 and Q2 are time-varying, we approximate them with piecewise constant matrices over some properly chosen time intervall Δt. In the main program we must then update the matrices and replace Q_o with the in the previous step computed S before each call to RICCE. In (4.36) the Σ_{ij}:s now depends not only on the time difference Δt, but also on the actual time t, and the fundamental matrix must be computed at each step. A proper choice of Δt must be a compromise between the time variations in the matrices, the accuracy desired and the increase of execution time which is a consequence of a reduction of Δt.

B. Numerical solution with Hamilton-Jacobi method.

Solving the linear-quadratic problem with Hamilton-Jacobi method seems at first glance to be a much more straightforward procedure than Euler-Lagrange method. The only computations involved are the solution of the system of non-linear differential equations (the Riccati equation)

$$\frac{dS}{dt} + A^T S + SA - SBQ_2^{-1}B^T S + Q_1 = 0 \qquad (4.37)$$

with boundary conditions given at the terminal time

$$S(t_1) = Q_o \qquad (4.38)$$

and the computation of the feedback matrix

$$L(t) = Q_2^{-1}(t) \cdot B^T(t) \cdot S(t) \qquad (4.39)$$

There are many numerical methods to solve equations of the type (4.37). We have here chosen to use a fourth order Runge-Kutta method, which probably is the method that gives the best accuracy with respect to the effort required. We then solve the Riccati equation (4.37) backwards in time, and after each step the feedback matrix can be computed. In the main program (appendix B) we will only consider the time-invariant case, which makes it possible to compute the matrices $BQ_2^{-1}B^T$ in (4.37) and $Q_2^{-1}B^T$ in (4.39) once for all. The inverse Q_2^{-1} is then computed before starting to solve (4.37), and there are then no more inversions involved in the computations. We have also here neglected the fact that S is symmetric, thereby hoping to get an alarm when the accuracy is too bad.

The program package (appendix B) consists of:

RKRICCE	Main program
GJRV	Subroutine
RK1STMAT	Subroutine
FUNC	Subroutine

SUBROUTINE GJRV (A, N, EPS, IERR, IA)

This is the same routine as used in LIOPCON.

SUBROUTINE RK1STMAT (T, YIN, H, YE, N, IA)

Solves the differential equation $\frac{dS}{dt} = F(S,t)$, where S is a quadratic matrix, with fourth order Runge-Kutta method.

Parameters:

YIN-the value of S at time T.
H-integration step length
YE-the value of S at time T + H.
N-order of S.
IA-dimension parameter.

SUBROUTINE FUNC (N)

This subroutine computes the matrix function

$F(S,T) = - A^TS - SA + SBQ_2^{-1}B^TS - Q_1$

The matrices A, $BQ_2^{-1}B^T$ and Q_1 lie in a common field. The parameter N is the order of the system.

PROGRAM RKRICCE

Main program administrating inputs and outputs. The inputs to RKRICCE are:

N-order of the system (max 10).
NU-number of insignals (max 10).

ITIME-number of equidistant points in which S is computed.

NUMBDIST-distance between printouts. This parameter is required because of the step length, that often must be chosen very small to get good accuracy (see section 5), and this would give an enormous lot of useless printouts. If NUMBDIST is set equal to M, the matrices S and L will be printed at $t = t_1$, $t = t_1$ - M x TIMEDIF, $t = t_1$ - 2M x TIMEDIF etc. The feedback matrix L is computed only at those points where we want it printed out.

TIMEDIF-time difference between the points in which S is computed (integration step length).

A-system matrix of order N x N.

B-system insignal matrix of order N x NU.

Q0, Q1-loss functional matrices of order N x N.

Q2-loss functional matrix of order NU x NU.

Concerning the input formats, see appendix B. The program is easily modified to handle the time-varying case. We then approximate the time-varying matrices with matrices that are piecewise constant over the integration step length, and update A, Q_1 and $BQ_2^{-1}B^T$ before each call to RK1STMAT, and $Q_2^{-1}B^T$ before the computation of the feedback matrix L.

5. EXAMPLES

A. Double-integral plant.

The system is

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = u \qquad (5.1)$$

We choose the cost functional

$$2V = x_1^2(t_1) + \int_{t_o}^{t_1} 0.5 \cdot u^2 \, ds \qquad (5.2)$$

which corresponds to the matrices

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} ; \; B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} ; \; Q_o = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} ; \; Q_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} ; \; Q_2 = (0.5)$$

This relatively simple choice gives us possibility to calculate an explicite expression of the matrix S(t). We will not carry through the computations, but just state that the solution is

$$S(t) = \frac{1}{D} \cdot \begin{pmatrix} 1 & -(t-t_1) \\ -(t-t_1) & (t-t_1)^2 \end{pmatrix} \qquad (5.3)$$

where

$$D = 1 - \frac{2}{3} \cdot (t - t_1)^3 \qquad (5.4)$$

Notice that

$$S(t_1) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = Q_o \qquad (5.5)$$

We have now possibilities to compare the computed solution with the exact solution, and get an idea about the accuracy of the methods. We choose arbitrarily to compute S(t) from $t = t_1$ to $t = t_1 - 10.0$. In the Euler-Lagrange-based program LIOPCON, the S- and L-matrices have been computed in 50, 20, 10, 5 and 2 points, corresponding to the time differences 0.2, 0.5, 1.0, 2.0 and 5.0 between the points. For every case we have compared the difference between computing the fundamental matrix just once (iteration method), and computing it at each step. The largest deviation from the exact S-matrix was found to be two units in the tenth digit, which must be regarded as a very good accuracy, considering that the computer used (CDC-3600) gives 10-11 significant digits. The execution time varies from 1 to 8 seconds, the longest time for computing 50 points with time difference 0.2, and the fundamental matrix computed at each step. The printouts for the case 10 points (time difference 1.0) are shown in appendix C together with the exact S-matrix.

The results obtained from LIOPCON shall be compared with those from Hamilton-Jacobi-based RKRICCE. If we choose the integration step length $h = 0.1$, S must be computed in 100 points to cover the time interval. The execution time then showed to be rather short, 3 seconds, but the accuracy no better than 4-5 correct digits. When h was reduced to 0.01 (1000 points), the accuracy was improved to 7-8 correct digits, but the execution time increased to 16 seconds. This is still not as accurate as the results obtained from LIOPCON, in spite of the much longer execution time needed. It is also obvious that a further attempt to improve the accuracy by decreasing the step length will be very expensive with respect to the execution time.

B. Oscillator.

The system is

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = -x_1 + u \tag{5.6}$$

The cost functional is chosen as

$$2V = x_1^2(t_1) + \int_{t_o}^{t_1} 0.5 \cdot u^2 \, ds \tag{5.7}$$

We then have the matrices

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix};\ B = \begin{pmatrix} 0 \\ 1 \end{pmatrix};\ Q_o = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix};\ Q_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix};\ Q_2 = (0.5)$$

The exact solution is also here possible to compute and is given by

$$S(t) = \frac{1}{D} \cdot \begin{pmatrix} \cos^2 (t-t_1) & -\frac{1}{2} \cdot \sin 2(t-t_1) \\ -\frac{1}{2} \cdot \sin 2(t-t_1) & \sin^2 (t-t_1) \end{pmatrix} \tag{5.8}$$

where

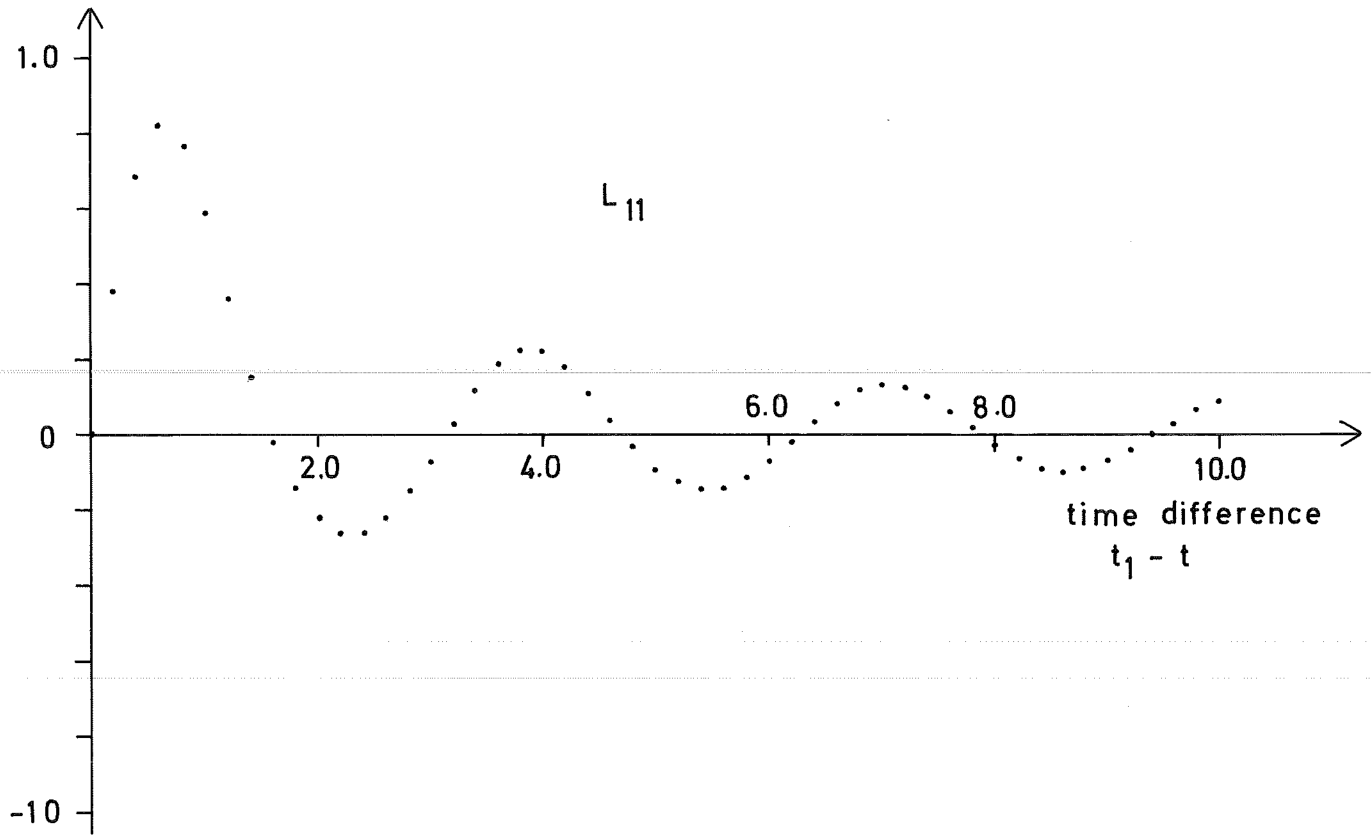
$$D = 1 - (t-t_1) + \frac{1}{2} \cdot \sin 2(t-t_1) \tag{5.9}$$

We have

$$S(t_1) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = Q_o \tag{5.10}$$

Both methods gave approximately the same results concerning accuracy and execution time as in example A, with the same different choices of the parameters ITIME and TIMEDIF.

An example of the output from LIOPCON is given in appendix D (10 points, time difference 1.0 sec.), and a computed feedback matrix L is presented in fig. 1 (50 points, time difference 0.2 sec.).



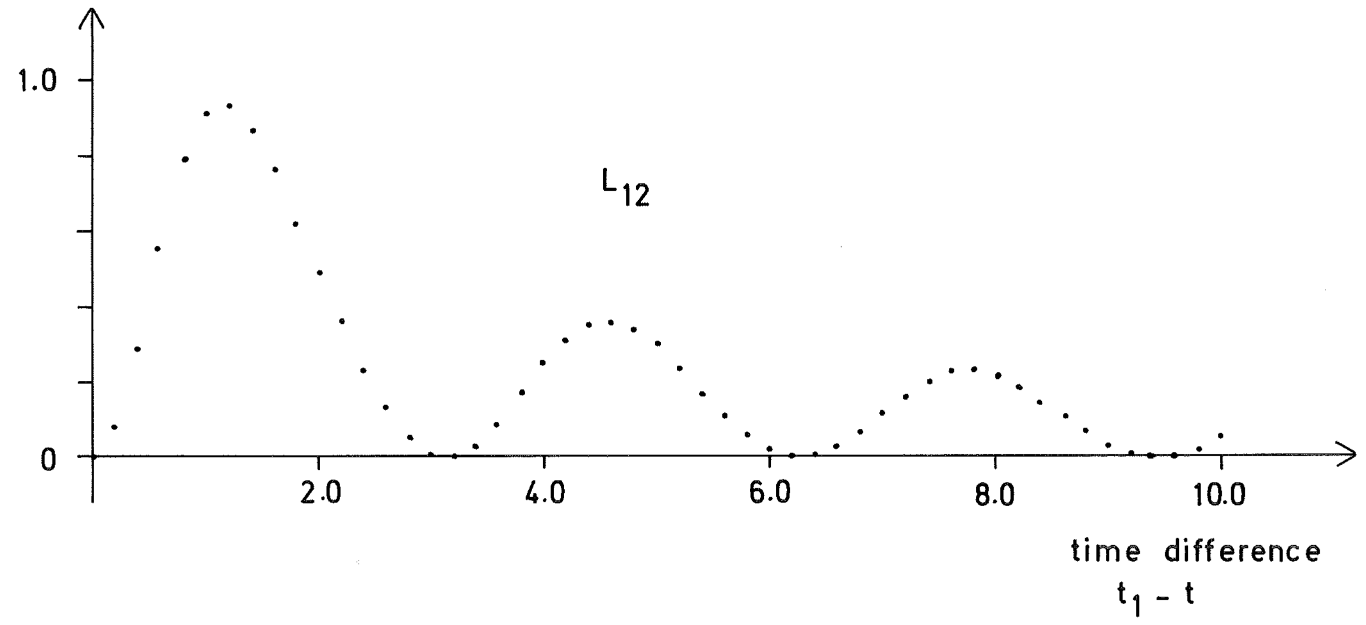


Fig. 1 Components of the feedback matrix L.
(Oscillator).

C. Double-integral plant.

Consider again the system

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = u \qquad (5.11)$$

The matrices A and B are the same as in A, but we choose now the cost functional matrices Q_1 and Q_2 as

$$Q_1 = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} ; \quad Q_2 = (1) \qquad (5.12)$$

In this case the matrix S will show to converge to a stationary value as the time difference t_1-t increases, and we will therefore consider three different boundary values of $S(t_1) = Q_o$.

We choose

$$Q_o^{\,1} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} ; \quad Q_o^{\,2} = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} ; \quad Q_o^{\,3} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \qquad (5.13)$$

The computed feedback matrices L are shown in fig. 2. We can see that independant of the boundary value $S(t_1)$, L(t) (and S(t)) reaches its stationary value for a time difference of about 5 seconds. The time needed for computing the stationary feedback matrix with program LIOPCON was 4 seconds (30 points, time difference 1.0 sec.), and the L-matrix had then converged to its stationary value with nine correct digits. When computing the fundamental matrix at each step, LIOPCON shows a weakness. Not only does the execution time increase (7 seconds), but the accuracy decreases to about 8 correct digits. This obviously depends on the fact that the norm of the matrix

$$\begin{pmatrix} A & -BQ_2^{-1}B^T \\ -Q_1 & -A^T \end{pmatrix} \cdot (t-t_1) \qquad (5.14)$$

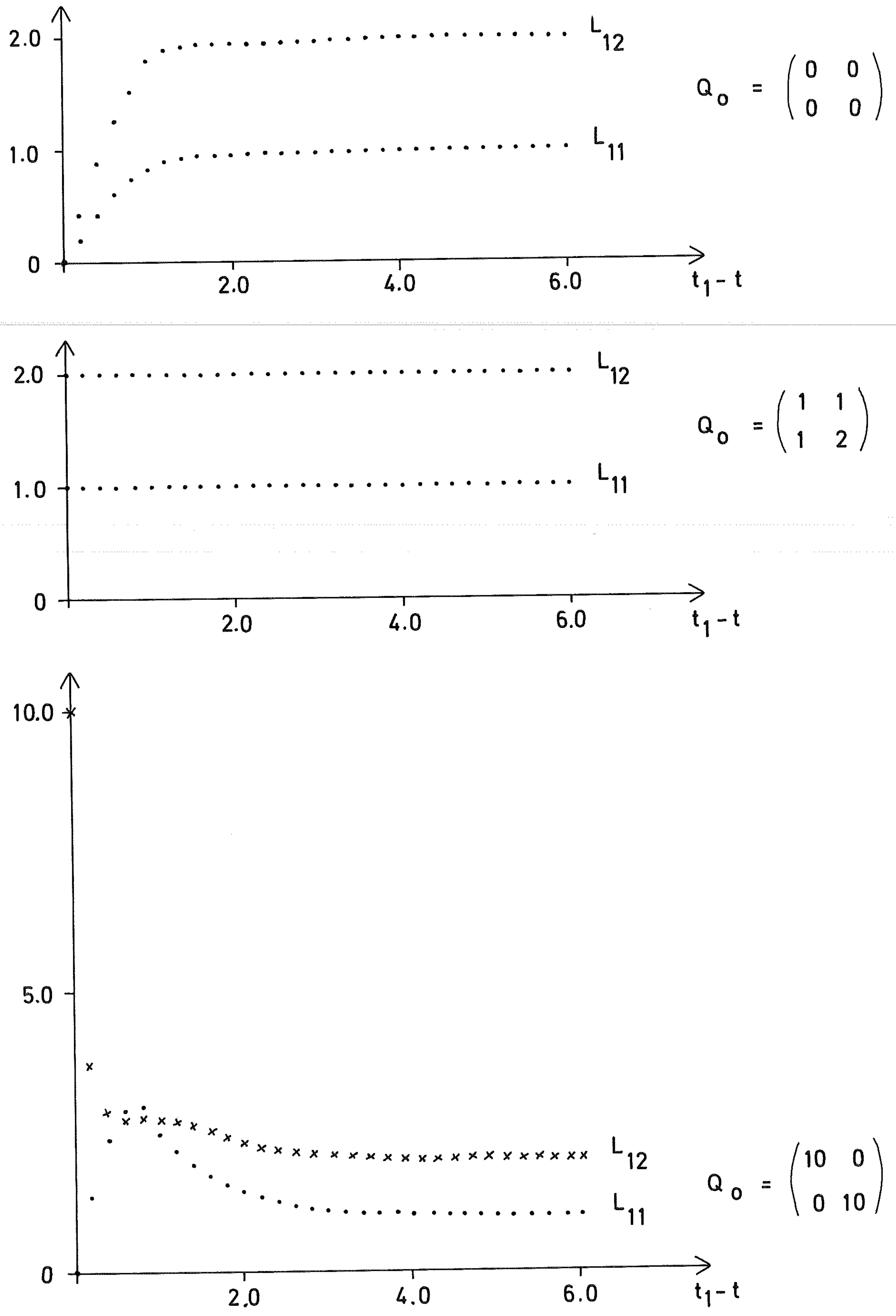


Fig. 2 Components of the feedback matrix L for various Q_o.
(Double integrator plant).

becomes too large. This means too many scalings and subsequent matrix multiplications, in which the errors grow too large.

When comparing the results above with the Hamilton-Jacobi program RKRICCE, we choose the integration step length h = 0.01, which proves to give 8-9 correct digits. However, the execution time was as long as 40 seconds, and this clearly shows the great advantage using Euler-Lagrange method when computing stationary optimal feedbacks. We will finally mention, that there may be cases when the program RKRICCE is preferred. If the system is time-varying with rapidly varying coefficient, where it is necessary to make the time interval over which we approximate the matrices very small, then it could be more economic to use the Runge-Kutta method.

Another advantage may be the storage requirement. The program LIOPCON as presented in appendix A needs about 7k depart from the system routines needed for input-output, while RKRICCE only requires about 2.5k. The main reasons for the large storage area required for LIOPCON, are the many arrays that are used, especially in subroutine MEXP7T, and that we have tried to avoid using the "common" and "equivalence" possibilities, thereby hoping to make the routines as flexible as possible. If we restrict the number of insignals and state variables allowed to five, the storage requirement will probably decrease to about 3k.

REFERENCES:

1. Athans, Optimal Control, McGraw-Hill Inc., 1966

2. Kalman, Contributions to the Theory of Optimal Control, Bol.Soc.Mat.Mex., vol 5, 1960

3. Åström, On the Choice of Smpling Rates in Optimal Linear Systems, Report RJ 243, 1963, IBM San Jose Research Laboratory, California, USA

APPENDIX A

```
      PROGRAM LIOPCON
C
C     COMPUTES THE OPTIMAL CONTROL LAW OF CONTINUOUS LINEAR DYNAMIC
C     SYSTEMS WITH QUADRATIC LOSS.
C
C     N-NUMBER OF STATES(MAX 10).
C     NU-NUMBER OF INSIGNALS(MAX 10).
C     ITIME-NUMBER OF EQUIDISTANT POINTS IN WHICH S AND L ARE COMPUTED.
C     THE FINAL TIME T1 IS NOT INCLUDED.
C     TIMEDIF-TIME DIFFERENCE BETWEEN THE POINTS.
C     ITER-ITER=0 MEANS THAT THE FUNDAMENTAL MATRIX WILL BE COMPUTED
C     FOR EACH STEP,ITER=1 MEANS THAT THE FUNDAMENTAL MATRIX IS COMPU-
C     TED ONLY FOR THE FIRST STEP AND THEN USED IN THE OTHER STEPS.
C
C     SUBROUTINE REQUIRED
C            RICCE
C            MEXP7T
C            GJRV
C            NORM
C
      DIMENSION A(10,10),B(10,10),Q0(10,10),Q1(10,10),Q2(10,10)
      DIMENSION S(10,10),JL(10,10),C(10,10)
C
      READ 1000,N,NU,ITIME,ITER,TIMEDIF
 1000 FORMAT(4I3,F10.5)
      READ 1001,((A(I,J),J=1,N),I=1,N)
      READ 1001,((B(I,J),J=1,NU),I=1,N)
      READ 1001,((Q0(I,J),J=1,N),I=1,N)
      READ 1001,((Q1(I,J),J=1,N),I=1,N)
      READ 1001,((Q2(I,J),J=1,NU),I=1,NU)
 1001 FORMAT(4E20.10)
      PRINT 1023
 1023 FORMAT(31H1PRINTOUTS FROM PROGRAM LIOPCON,/)
      PRINT 1002
 1002 FORMAT(14H THE SYSTEM IS,/)
      PRINT 1003
 1003 FORMAT(9H MATRIX A,/)
      DO 2 K=1,N
    2 PRINT 1004,(A(K,J),J=1,N)
 1004 FORMAT(6E20.10)
      PRINT 1005
 1005 FORMAT(/,9H MATRIX B,/)
      DO 4 K=1,N
    4 PRINT 1004,(B(K,J),J=1,NU)
      PRINT 1006
 1006 FORMAT(/,10H MATRIX Q0,/)
      DO 6 K=1,N
    6 PRINT 1004,(Q0(K,J),J=1,N)
      PRINT 1007
 1007 FORMAT(/,10H MATRIX Q1,/)
      DO 8 K=1,N
    8 PRINT 1004,(Q1(K,J),J=1,N)
      PRINT 1008
 1008 FORMAT(/,10H MATRIX Q2,/)
      DO 10 K=1,NU
   10 PRINT 1004,(Q2(K,J),J=1,NU)
C
      PRINT 1009,ITIME
 1009 FORMAT(/,30H NUMBER OF EQUIDISTANT POINTS=,I3)
      PRINT 1010
 1010 FORMAT(30H (EXCLUDING THE FINAL TIME T1),/)
```

APPENDIX A
(continued)

```
      PRINT 1011,TIMEDIF
 1011 FORMAT(36H TIME DIFFERENCE BETWEEN THE POINTS=,F10.5,/)
      IF(ITER) 14,12,14
   12 PRINT 1012
 1012 FORMAT(48H THE FUNDAMENTAL MATRIX IS COMPUTED AT EACH STEP,/)
      GO TO 16
   14 PRINT 1013
 1013 FORMAT(54H THE FUNDAMENTAL MATRIX IS COMPUTED ONLY AT FIRST STEP)
   16 CONTINUE
C
      PRINT 1014
 1014 FORMAT(1H1,5X,6HTD=0.0,/)
      PRINT 1015
 1015 FORMAT(13H S-INITIAL=Q0,/)
      DO 18 K=1,N
   18 PRINT 1004,(Q0(K,J),J=1,N)
      PRINT 1016
 1016 FORMAT(/,16H L-INITIAL(U=-L*X),/)
C
      DO 20 I=1,NU
      DO 20 J=1,NU
   20 UL(I,J)=Q2(I,J)
      CALL GJRV(UL,NU,1.0E-008,IERR,10)
      IF(IERR+1) 24,22,24
   22 PRINT 1017
 1017 FORMAT(//,39H THE MATRIX Q2 IS NOT POSITIVE DEFINITE)
      GO TO 100
   24 DO 28 I=1,NU
      DO 28 J=1,N
      R=0.
      DO 26 K=1,NU
   26 R=R+UL(I,K)*B(J,K)
   28 C(I,J)=R
      DO 32 I=1,NU
      DO 32 J=1,N
      R=0.
      DO 30 K=1,N
   30 R=R+C(I,K)*Q0(K,J)
   32 UL(I,J)=R
C
      DO 34 K=1,NU
   34 PRINT 1004,(UL(K,J),J=1,N)
C
C     START THE LOOP
C
      DO 80 ICOUNT=1,ITIME
      TD=TIMEDIF*FLOAT(ICOUNT)
      PRINT 1018,TD
 1018 FORMAT(///,5X,3HTD=,F10.5,/)
      IF(ITER-1) 36,40,36
   36 CONTINUE
C
      CALL RICCE(A,B,Q0,Q1,Q2,S,N,NU,10,10,TD,IERR)
C
      IF(IERR+1) 60,38,60
   38 PRINT 1019
 1019 FORMAT(33H AN INVERSION HAS FAILED IN RICCE)
      GO TO 80
C
C     ITERATION
C
   40 IF(ICOUNT-1) 46,42,46
```

APPENDIX A
(continued)

```
   42 CONTINUE
C
      CALL RICCE(A,B,QU,Q1,Q2,S,N,NU,10,10,TD,IERR)
C
      IF(IERR+1) 60,44,60
   44 PRINT 1019
      PRINT 1020
 1020 FORMAT(50H THE PROBLEM IS IMPOSSIBLE TO SOLVE WITH ITERATION)
      GO TO 100
   46 DO 46 I=1,N
      DO 46 J=1,N
   48 Q0(I,J)=S(I,J)
C
      CALL ITERATE(A,B,Q0,Q1,Q2,S,N,NU,10,10,TD,IERR)
C
      IF(IERR+1) 60,50,60
   50 PRINT 1018
      PRINT 1019
      GO TO 100
   60 PRINT 1021
 1021 FORMAT(18H COMPUTED S-MATRIX,/)
      DO 62 K=1,N
   62 PRINT 1004,(S(K,J),J=1,N)
      PRINT 1022
 1022 FORMAT(/,26H COMPUTED L-MATRIX(U=-L*X),/)
      DO 66 I=1,NU
      DO 66 J=1,N
      R=0.
      DO 64 K=1,N
   64 R=R+C(I,K)*S(K,J)
   66 UL(I,J)=R
      DO 68 K=1,NU
   68 PRINT 1004,(UL(K,J),J=1,N)
C
   60 CONTINUE
C
  100 CONTINUE
      CALL EXIT
      END
```

APPENDIX A
(continued)

```
      SUBROUTINE RICCE(A,B,Q0,Q1,Q2,S,N,NU,IA,IB,TD,IERR)
C
C     THE SUBROUTINE COMPUTES THE SOLUTION TO THE RICATTIEQUATION
C     DS/DT=(AT)*S+S*A-S*B*(Q2-1)*(BT)*S+Q1 WITH S(T1)=Q0,
C     BY USING THE EXPONENTIAL SERIES FOR THE CANONICAL EQUATION.
C
C     A,Q0,Q1,S=NXN-MATRICES,S(T) IS THE SOLUTION.
C     B=NXNU-MATRIX.
C     Q2=NUXNU-MATRIX.
C     IA AND IB ARE THE DIMENSION PARAMETERS.
C     TD IS THE DIFFERENCE T1-T.
C     IERR IS RETURNED=-1 IF ANY INVERSION HAS FAILED.
C     MAXIMUM ORDER OF THE SYSTEM=10.
C     THE ROUTINE HAS AN ENTRY POINT CALLED ITERATE.
C     WHEN THE ROUTINE IS CALLED WITH ITERATE,WHICH REQUIRES THAT
C     A PREVIOUS CALL TO RICCE HAS BEEN MADE,USE IS MADE OF THE IN THE
C     FIRST CALL COMPUTED FUNDAMENTALMATRIX.Q0 IS THEN SET EQUAL TO
C     THE PREVIOUSLY COMPUTED S OUTSIDE THE ROUTINE BEFORE CALLING.
C
C     SUBROUTINE REQUIRED
C               MEXP7T
C               NORM
C               GJRV
C
      DIMENSION A(IA,IA),B(IA,IB),Q0(IA,IA),Q1(IA,IA),Q2(IB,IB),S(IA,IA)
      DIMENSION C(10,10),EA(20,20),EB(20,20)
C
C     COMPUTATION OF EULERMATRIX
C
      DO 10 I=1,N
      DO 10 J=1,N
      EA(I,J)=-A(I,J)*TD
      C(I,J)=Q2(I,J)
      NPI=N+I
      NPJ=N+J
      EA(NPI,J)=Q1(I,J)*TD
   10 EA(NPI,NPJ)=A(J,I)*TD
C
      CALL GJRV(C,NU,1.0E-008,IERR,10)
      IF(IERR+1) 15,50,15
   15 DO 20 I=1,N
      DO 20 J=1,N
      R=0.0
      DO 21 L=1,NU
      DO 21 M=1,NU
   21 R=R+B(I,L)*C(L,M)*B(J,M)
      NPJ=N+J
   20 EA(I,NPJ)=R*TD
C
C     COMPUTATION OF EB=EXP(EA)
C
      M=N+N
      III=0
      CALL MEXP7T(EA,EB,M,20,III)
C
      GO TO 29
C
      ENTRY ITERATE
C
   29 DO 30 I=1,N
      DO 30 J=1,N
      NPI=N+I
      R=0.0
      DO 31 K=1,N
      NPK=N+K
   31 R=R+EB(NPI,NPK)*DD(K,J)
   30 C(I,J)=EB(NPI,J)+R
C
      DO 40 I=1,N
      DO 40 J=1,N
      R=0.0
      DO 41 K=1,N
      NPK=N+K
   41 R=R+EB(I,NPK)*DD(K,J)
   40 EA(I,J)=EB(I,J)+R
C
      CALL GJRV(EA,N,1.0E-008,IERR,20)
      IF(IERR+1) 45,60,45
   45 DO 50 I=1,N
      DO 50 J=1,N
      R=0.0
      DO 51 K=1,N
   51 R=R+C(I,K)*EA(K,J)
   50 S(I,J)=R
C
   60 RETURN
      END
```

APPENDIX A
(continued)

```
      SUBROUTINE MEXP7T(A,B,N,IA,NOTRACE)
C
C     COMPUTES B=EXP(A) BY ORIGIN SHIFT AND SERIES EXPANSION USING 7
C     TERMS.
C
C     A-NXN-MATRIX.
C     B-NXN-MATRIX.
C     IA-DIMENSION PARAMETER.
C     NOTRACE=0 MEANS THAT NO TRACE
C     COMPUTATION WILL BE PERFORMED.
C     MAXIMUM ORDER OF A AND B=20.
C     THE MATRIX A IS DESTROYED.
C
C     SUBROUTINE REQUIRED
C             NORM
C
      DIMENSION  A(IA,IA),B(IA,IA),C(7,20,20)
      IF(NOTRACE) 1,5,1
    1 TRAA=0.
      DO 2 I=1,N
    2 TRAA=TRAA+A(I,I)
      IF(TRAA) 3,5,3
    3 TRAA=TRAA/N
      DO 4 I=1,N
    4 A(I,I)=A(I,I)-TRAA
    5 KDIV=0
      DO 6 I=1,N
      DO 6 J=1,N
    6 C(1,I,J)=A(I,J)
      DO 10 LOP=2,7
      DO 10 I=1,N
      DO 10 J=1,N
      R=0.
      DO 8 K=1,N
    8 R=R+C(LOP-1,I,K)*A(K,J)
   10 C(LOP,I,J)=R/LOP
   12 DO 14 I=1,N
      DO 14 J=1,N
   14 B(I,J)=C(7,I,J)
      CALL NORM(B,N,IA,P)
      IF(P-1.0E-010) 20,20,16
   16 REST=P*1.0E+010
   15 KDIV=KDIV+1
      RO=2.0**(KDIV*7)
      IF(RO-REST) 15,15,17
   17 DO 18 LOP=1,7
      PKVAD=2.0**(KDIV*LOP)
      DO 18 I=1,N
      DO 18 J=1,N
   18 C(LOP,I,J)=C(LOP,I,J)/PKVAD
   20 DO 22 I=1,N
      DO 22 J=1,N
   22 B(I,J)=0.0
      DO 26 I=1,N
   26 B(I,I)=1.0
      DO 28 LOP=1,7
      DO 28 I=1,N
      DO 28 J=1,N
   28 B(I,J)=B(I,J)+C(LOP,I,J)
      IF(KDIV) 46,46,36
   36 DO 44 IPK=1,KDIV
```

APPENDIX A
(continued)

```
      DO 40 I=1,N
      DO 40 J=1,N
      R=0.
      DO 38 K=1,N
   38 R=R+B(I,K)*B(K,J)
   40 C(1,I,J)=R
      DO 42 I=1,N
      DO 42 J=1,N
   42 B(I,J)=C(1,I,J)
   44 CONTINUE
   46 IF(NOTRACE) 47,50,47
   47 IF(TRAA) 49,50,49
   49 CC=EXPF(TRAA)
      DO 48 I=1,N
      DO 48 J=1,N
   48 B(I,J)=CC*B(I,J)
   50 RETURN
      END
```

APPENDIX A
(continued)

```
      SUBROUTINE GJRV(A,N,EPS,IERR,IA)
C
C     INVERTS ASYMMETRIC MATRICES, HAS EMERGENCY EXIT,
C     REQUIRES N**2+4*N WORDS OF ARRAY STORAGE
C
C     A IS THE NAME OF THE MATRIX TO BE INVERTED
C     N IS THE ORDER OF A
C     EPS IS A VALUE TO BE USED AS A TOLERANCE FOR
C     ACCEPTANCE OF THE SINGULARITY OF A GIVEN MATRIX
C     IERR IS AN INTEGER VARIABLE WHICH WILL CONTAIN ZERO
C     UPON RETURN IF INVERSION IS COMPLETED OR -1 IF SOME
C     PIVOT ELEMENT HAS AN ABSOLUTE VALUE LESS THAN EPS
C     IA IS THE DIMENSION PARAMETER
C     MAXIMUM ORDER OF A=40
C     THE ORIGINAL MATRIX IS DESTROYED
C     IF IERR IS RETURNED =-1 THEN THE INVERSION HAS FAILED
C     OTHERWISE THE RESULTING INVERSE IS PLACED IN A
C
C     SUBROUTINE REQUIRED
C              NONE
C
      DIMENSION A(IA,IA),B(40),C(40),IP(40),IQ(40)
      IERR=0
      DO 140 K=1,N
      PIVOT=0.0
      DO 120 I=K,N
      DO 2 J=K,N
      IF(ABSF(A(I,J))-ABSF(PIVOT)) 2,2,1
    1 PIVOT=A(I,J)
      IP(K)=I
      IQ(K)=J
    2 CONTINUE
  120 CONTINUE
      IF(ABSF(PIVOT)-EPS) 100,100,3
    3 IF(IP(K)-K) 4,6,4
    4 DO 5 J=1,N
      IPX=IP(K)
      Z=A(IPX,J)
      A(IPX,J)=A(K,J)
    5 A(K,J)=Z
    6 IF(IQ(K)-K) 7,9,7
    7 DO 8 I=1,N
      IPX=IQ(K)
      Z=A(I,IPX)
      A(I,IPX)=A(I,K)
    8 A(I,K)=Z
    9 DO 13 J=1,N
      IF(J-K) 11,10,11
   10 B(J)=1.0/PIVOT
      C(J)=1.0
      GO TO 12
   11 B(J)=-A(K,J)/PIVOT
      C(J)=A(J,K)
   12 A(K,J)=0.0
      A(J,K)=0.0
   13 CONTINUE
      DO 14 I=1,N
      DO 14 J=1,N
   14 A(I,J)=A(I,J)+C(I)*B(J)
  140 CONTINUE
      DO 20 KP=1,N
```

APPENDIX A
(continued)

```
      K=N+1-KP
      IF(IP(K)-K) 15,17,15
   15 DO 16 I=1,N
      IPX=IP(K)
      Z=A(I,IPX)
      A(I,IPX)=A(I,K)
   16 A(I,K)=Z
   17 IF(IQ(K)-K) 18,20,18
   18 DO 19 J=1,N
      IPX=IQ(K)
      Z=A(IPX,J)
      A(IPX,J)=A(K,J)
   19 A(K,J)=Z
   20 CONTINUE
      GO TO 21
  100 IERR=-1
   21 RETURN
      END
```

APPENDIX A
(continued)

```
      SUBROUTINE NORM(A,N,IA,S)
C
C     THE SUBROUTINE COMPUTES THE MINIMAXNORM OF A WHERE
C     A=NXN-MATRIX
C     S IS THE RESULTING NORM
C     IA IS THE DIMENSION PARAMETER
C
C     SUBROUTINE REQUIRED
C            NONE
C
      DIMENSION A(IA,IA)
      S=S1=0.0
      DO 20 J=1,N
      R=0.0
      DO 10 I=1,N
      R=R+ABSF(A(I,J))
   10 CONTINUE
      IF(R.GT.S1) 15,20
   15 S1=R
   20 CONTINUE
C     S1=MAX OVER THE COLUMNS
      DO 40 I=1,N
      R=0.0
      DO 30 J=1,N
      R=R+ABSF(A(I,J))
   30 CONTINUE
      IF(R.GT.S) 35,40
   35 S=R
   40 CONTINUE
C     S=MAX OVER THE ROWS
C
      IF(S.GT.S1) 50,60
   50 S=S1
   60 RETURN
      END
```

APPENDIX B

```
      PROGRAM RKRICCE
C
C     COMPUTES THE OPTIMAL CONTROL LAW OF CONTINUOUS LINEAR
C     DYNAMIC SYSTEMS WITH QUADRATIC LOSS,BY SOLVING THE RICCATTI-
C     EQUATION WITH RUNGE-KUTTA METHOD.
C
C     N-NUMBER OF STATES(MAX 10).
C     NU-NUMBER OF INSIGNALS(MAX 10).
C     ITIME-NUMBER OF EQUIDISTANT POINTS IN WHICH S AND L ARE COM-
C     PUTED.THE FINAL TIME T1 IS NOT INCLUDED.
C     TIMEDIF-TIME DIFFERENCE BETWEEN THE POINTS(INTEGRATION STEP).
C     NUMBDIST- DISTANCE BETWEEN PRINTOUTS.IF NUMBDIST IS SET=N,THEN
C     THE COMPUTED MATRICES WILL BE PRINTED FOR T=T1,T=T1-N*TIMEDIF,
C     T=T1-2N*TIMEDIF ETC.
C
C     SUBROUTINE REQUIRED
C             GJRV
C             RK1STMAT
C             FUNC
C
      DIMENSION A(10,10),B(10,10),Q0(10,10),Q1(10,10),Q2(10,10)
      DIMENSION S(10,10),UL(10,10),Q2INVBT(10,10),SE(10,10)
      COMMON/SYSTEM/A,Q1,Q2
      READ 1000,N,NU,ITIME,NUMBDIST,TIMEDIF
 1000 FORMAT(2I2,2I5,F10.5)
      READ 1001,((A(I,J),J=1,N),I=1,N)
      READ 1001,((B(I,J),J=1,NU),I=1,N)
      READ 1001,((Q0(I,J),J=1,N),I=1,N)
      READ 1001,((Q1(I,J),J=1,N),I=1,N)
      READ 1001,((Q2(I,J),J=1,NU),I=1,NU)
 1001 FORMAT(4E20.10)
      PRINT 1002
 1002 FORMAT(31H1PRINTOUTS FROM PROGRAM RKRICCE ,/)
      PRINT 1003
 1003 FORMAT(14H THE SYSTEM IS,/)
      PRINT 1004
 1004 FORMAT(9H MATRIX A,/)
      DO 2 K=1,N
    2 PRINT 1005,(A(K,J),J=1,N)
 1005 FORMAT(6E20.10)
      PRINT 1006
 1006 FORMAT(/,9H MATRIX B,/)
      DO 4 K=1,N
    4 PRINT 1005,(B(K,J),J=1,NU)
      PRINT 1007
 1007 FORMAT(/,10H MATRIX Q0,/)
      DO 6 K=1,N
    6 PRINT 1005,(Q0(K,J),J=1,N)
      PRINT 1008
 1008 FORMAT(/,10H MATRIX Q1,/)
      DO 8 K=1,N
    8 PRINT 1005,(Q1(K,J),J=1,N)
      PRINT 1009
 1009 FORMAT(/,10H MATRIX Q2,/)
      DO 10 K=1,NU
   10 PRINT 1005,(Q2(K,J),J=1,NU)
C
      PRINT 1010,ITIME
 1010 FORMAT(/,30H NUMBER OF EQUIDISTANT POINTS=,I5)
      PRINT 1011
 1011 FORMAT(30H (EXCLUDING THE FINAL TIME T1),/)
```

APPENDIX B
(continued)

```
      PRINT 1012,TIMEDIF
 1012 FORMAT(36H TIME DIFFERENCE BETWEEN THE POINTS=,F10.5,/)
      PRINT 1021,NUMBDIST
 1021 FORMAT(6H EVERY,I5,19HTH POINT IS PRINTED,/)
C
      CALL GJRV(Q2,NU,1.0E-006,IERR,10)
      IF(IERR+1) 14,12,14
   12 PRINT 1013
 1013 FORMAT(/,39H THE MATRIX Q2 IS NOT POSITIVE DEFINITE)
      GO TO 80
   14 DO 18 I=1,NU
      DO 18 J=1,N
      R=0.
      DO 16 K=1,NU
   16 R=R+Q2(I,K)*B(J,K)
   18 Q2INVBT(I,J)=R
      DO 22 I=1,N
      DO 22 J=1,N
      R=0.
      DO 20 K=1,NU
   20 R=R+B(I,K)*Q2INVBT(K,J)
   22 Q2(I,J)=R
      DO 24 I=1,N
      DO 24 J=1,N
   24 S(I,J)=Q0(I,J)
      TD=0.
      PRINT 1014
 1014 FORMAT(1H1,5X,6HTD=0.0,/)
      PRINT 1015
 1015 FORMAT(13H S-INITIAL=Q0,/)
      DO 26 K=1,N
   26 PRINT 1005,(Q0(K,J),J=1,N)
      PRINT 1016
 1016 FORMAT(/,18H L-INITIAL(U=-L*X),/)
      DO 30 I=1,NU
      DO 30 J=1,N
      R=0.
      DO 28 K=1,N
   28 R=R+Q2INVBT(I,K)*S(K,J)
   30 UL(I,J)=R
      DO 32 K=1,NU
   32 PRINT 1005,(UL(K,J),J=1,N)
C
C     START THE LOOP
C
      IPRINT=0
      DO 74 ICOUNT=1,ITIME
      IPRINT=IPRINT+1
      TD=TIMEDIF*FLOAT(ICOUNT)
      DELTA=-TIMEDIF
      XX=0.
      CALL RK1STMAT(XX,S,DELTA,SE,N,10)
      IF(IPRINT-NUMBDIST) 70,40,40
   40 IPRINT=0
      DO 44 I=1,NU
      DO 44 J=1,N
      R=0.
      DO 42 K=1,N
   42 R=R+Q2INVBT(I,K)*SE(K,J)
   44 UL(I,J)=R
      PRINT 1018,TD
 1018 FORMAT(///,5X,3HTD=,F10.5,/)
```

APPENDIX B
(continued)

```
      PRINT 1019
 1019 FORMAT(18H COMPUTED S-MATRIX,/)
      DO 66 K=1,N
   66 PRINT 1005,(SE(K,J),J=1,N)
      PRINT 1020
 1020 FORMAT(/,26H COMPUTED L-MATRIX(U=-L*X),/)
      DO 68 K=1,NU
   68 PRINT 1005,(UL(K,J),J=1,N)
C
   70 DO 72 I=1,N
      DO 72 J=1,N
   72 S(I,J)=SE(I,J)
   74 CONTINUE
C
   80 CALL EXIT
      END
```

APPENDIX B
(continued)

```
      SUBROUTINE GJRV(A,N,EPS,IERR,IA)
C
C     INVERTS ASYMMETRIC MATRICES, HAS EMERGENCY EXIT,
C     REQUIRES N**2+4*N WORDS OF ARRAY STORAGE
C
C     A IS THE NAME OF THE MATRIX TO BE INVERTED
C     N IS THE ORDER OF A
C     EPS IS A VALUE TO BE ØSED AS A TOLERANCE FOR
C     ACCEPTANCE OF THE SINGULARITY OF A GIVEN MATRIX
C     IERR IS AN INTEGER VARIABLE WHICH WILL CONTAIN ZERO
C     UPON RETURN IF INVERSION IS COMPLETED OR -1 IF SOME
C     PIVOT ELEMENT HAS AN ABSOLUTE VALUE LESS THAN EPS
C     IA IS THE DIMENSION PARAMETER
C     MAXIMUM ORDER OF A=40
C     THE ORIGINAL MATRIX IS DESTROYED
C     IF IERR IS RETURNED =-1 THEN THE INVERSION HAS FAILED
C     OTHERWISE THE RESULTING INVERSE IS PLACED IN A
C
C     SUBROUTINE REQUIRED
C             NONE
C
      DIMENSION A(IA,IA),B(40),C(40),IP(40),IQ(40)
      IERR=0
      DO 140 K=1,N
      PIVOT=0.0
      DO 120 I=K,N
      DO 2 J=K,N
      IF(ABSF(A(I,J))-ABSF(PIVOT)) 2,2,1
    1 PIVOT=A(I,J)
      IP(K)=I
      IQ(K)=J
    2 CONTINUE
  120 CONTINUE
      IF(ABSF(PIVOT)-EPS) 100,100,3
    3 IF(IP(K)-K) 4,6,4
    4 DO 5 J=1,N
      IPX=IP(K)
      Z=A(IPX,J)
      A(IPX,J)=A(K,J)
    5 A(K,J)=Z
    6 IF(IQ(K)-K) 7,9,7
    7 DO 8 I=1,N
      IPX=IQ(K)
      Z=A(I,IPX)
      A(I,IPX)=A(I,K)
    8 A(I,K)=Z
    9 DO 13 J=1,N
      IF(J-K) 11,10,11
   10 B(J)=1.0/PIVOT
      C(J)=1.0
      GO TO 12
   11 B(J)=-A(K,J)/PIVOT
      C(J)=A(J,K)
   12 A(K,J)=0.0
      A(J,K)=0.0
   13 CONTINUE
      DO 14 I=1,N
      DO 14 J=1,N
   14 A(I,J)=A(I,J)+C(I)*B(J)
  140 CONTINUE
      DO 20 KP=1,N
```

APPENDIX B
(continued)

```
      K=N+1-KP
      IF(IP(K)-K) 15,17,15
   15 DO 16 I=1,N
      IPX=IP(K)
      Z=A(I,IPX)
      A(I,IPX)=A(I,K)
   16 A(I,K)=Z
   17 IF(IQ(K)-K) 18,20,18
   18 DO 19 J=1,N
      IPX=IQ(K)
      Z=A(IPX,J)
      A(IPX,J)=A(K,J)
   19 A(K,J)=Z
   20 CONTINUE
      GO TO 21
  100 IERR=-1
   21 RETURN
      END
```

APPENDIX B
(continued)

```
      SUBROUTINE RK1STMAT(T,YIN,H,YE,N,IA)
C
C     SOLVES THE MATRIX DIFFERENTIAL EQUATION DA/DT=F(A,T) BY
C     RUNGE-KUTTA METHOD.
C
C     T-ACTUAL TIME.
C     YIN-ACTUAL MATRIX.
C     H-INTEGRATION STEP LENGTH.
C     YE-COMPUTED MATRIX AT T+H.
C     N-ORDER OF THE MATRICES.
C     IA-DIMENSION PARAMETER.
C
C     SUBROUTINE REQUIRED
C              FUNC
C
      DIMENSION YIN(IA,IA),YE(IA,IA),W(10,10),Z(10,10),A(5)
      COMMON/FUNCTION/ TE,W,Z
      A(1)=A(2)=A(5)=H/2.
      A(3)=A(4)=H
      TE=T
      DO 10 I=1,N
      DO 10 J=1,N
   10 YE(I,J)=W(I,J)=YIN(I,J)
      DO 20 K=1,4
      CALL FUNC(N)
      TE=T+A(K)
      DO 20 I=1,N
      DO 20 J=1,N
      W(I,J)=YIN(I,J)+A(K)*Z(I,J)
   20 YE(I,J)=YE(I,J)+A(K+1)*Z(I,J)/3.
      RETURN
      END
```

APPENDIX B
(continued)

```
      SUBROUTINE FUNC(N)
C
C     THE MATRIX FUNCTION IS
C     DS/DT=-(AT)*S-S*A+S*B*(Q2-I)*(BT)*S-Q1
C
      DIMENSION A(10,10),Q1(10,10),Q2(10,10),T(10,10)
      DIMENSION S(10,10),DSDT(10,10)
      COMMON/SYSTEM/A,Q1,Q2
      COMMON/FUNCTION/TE,S,DSDT
      DO 4 I=1,N
      DO 4 J=1,N
      R=0.
      DO 2 K=1,N
    2 R=R+S(I,K)*A(K,J)
    4 DSDT(I,J)=-R
      DO 6 I=1,N
      DO 6 J=1,N
    6 T(I,J)=DSDT(J,I)
      DO 8 I=1,N
      DO 8 J=1,N
    8 DSDT(I,J)=DSDT(I,J)+T(I,J)
      DO 12 I=1,N
      DO 12 J=1,N
      R=0.
      DO 10 K=1,N
   10 R=R+Q2(I,K)*S(K,J)
   12 T(I,J)=R
      DO 16 I=1,N
      DO 16 J=1,N
      R=0.
      DO 14 K=1,N
   14 R=R+S(I,K)*T(K,J)
   16 DSDT(I,J)=DSDT(I,J)+R-Q1(I,J)
      RETURN
      END
```

APPENDIX C

```
    TD=0.0

S-INITIAL=00

     1.0000000000+000     -0.0000000000+000
    -0.0000000000+000     -0.0000000000+000


    TD=    1.00000

EXACT S-MATRIX

     5.9999999999-001      5.9999999999-001
     5.9999999999-001      5.9999999999-001

COMPUTED S-MATRIX

     6.0000000000-001      6.0000000000-001
     6.0000000000-001      6.0000000000-001


    TD=    2.00000

EXACT S-MATRIX

     1.5789473684-001      3.1578947368-001
     3.1578947368-001      6.3157894736-001

COMPUTED S-MATRIX

     1.5789473685-001      3.1578947369-001
     3.1578947369-001      6.3157894738-001


    TD=    3.00000

EXACT S-MATRIX

     5.2631578947-002      1.5789473684-001
     1.5789473684-001      4.7368421052-001

COMPUTED S-MATRIX

     5.2631578950-002      1.5789473685-001
     1.5789473685-001      4.7368421054-001


    TD=    4.00000

EXACT S-MATRIX

     2.2900763358-002      9.1603053433-002
     9.1603053433-002      3.6641221374-001

COMPUTED S-MATRIX

     2.2900763359-002      9.1603053436-002
     9.1603053436-002      3.6641221374-001
     TD=    5.00000

EXACT S-MATRIX

   1.1857707510-002    5.9288537550-002
   5.9288537550-002    2.9644268775-001

COMPUTED S-MATRIX

   1.1857707509-002    5.9288537548-002
   5.9288537547-002    2.9644268774-001



     TD=    6.00000

EXACT S-MATRIX

   6.8965517241-003    4.1379310344-002
   4.1379310344-002    2.4827586206-001

COMPUTED S-MATRIX

   6.8965517238-003    4.1379310346-002
   4.1379310343-002    2.4827586208-001



     TD=    7.00000

EXACT S-MATRIX

   4.3541364295-003    3.0478955007-002
   3.0478955007-002    2.1335268504-001

COMPUTED S-MATRIX

   4.3541364293-003    3.0478955007-002
   3.0478955006-002    2.1335268505-001



     TD=    8.00000

EXACT S-MATRIX

   2.9211295033-003    2.3369036027-002
   2.3369036027-002    1.8695228822-001

COMPUTED S-MATRIX

   2.9211295034-003    2.3369036027-002
   2.3369036027-002    1.8695228822-001



     TD=    9.00000

EXACT S-MATRIX
```

```
     2.0533880903-003     1.8480492813-002
     1.8480492813-002     1.6632443532-001

COMPUTED S-MATRIX

     2.0533880899-003     1.8480492813-002
     1.8480492809-002     1.6632443531-001


       TD=   10.00000

EXACT S-MATRIX

     1.4977533699-003     1.4977533700-002
     1.4977533700-002     1.4977533699-001

COMPUTED S-MATRIX

     1.4977533699-003     1.4977533700-002
     1.4977533699-002     1.4977533700-001
```

APPENDIX D

```
PRINTOUTS FROM PROGRAM LIOPCON

THE SYSTEM IS

MATRIX A

  -0.0000000000+000    1.0000000000+000
  -1.0000000000+000   -0.0000000000+000

MATRIX B

  -0.0000000000+000
   1.0000000000+000

MATRIX Q0

   1.0000000000+000   -0.0000000000+000
  -0.0000000000+000   -0.0000000000+000

MATRIX Q1

  -0.0000000000+000   -0.0000000000+000
  -0.0000000000+000   -0.0000000000+000

MATRIX Q2

   5.0000000000-001

NUMBER OF EQUIDISTANT POINTS= 10
(EXCLUDING THE FINAL TIME T1)

TIME DIFFERENCE BETWEEN THE POINTS=   1.00000

THE FUNDAMENTAL MATRIX IS COMPUTED ONLY AT FIRST STEP
```

```
     TD=0.0

S-INITIAL=00

   1.0000000000+000    -0.0000000000+000
  -0.0000000000+000    -0.0000000000+000

L-INITIAL(U=-L*X)

   0.0000000000+000     0.0000000000+000


     TD=    1.00000

COMPUTED S-MATRIX

   1.8890629221-001     2.9420411873-001
   2.9420411874-001     4.5819576716-001

COMPUTED L-MATRIX(U=-L*X)

   5.8840823750-001     9.1639153431-001


     TD=    2.00000

COMPUTED S-MATRIX

   5.1260397126-002    -1.1200601111-001
  -1.1200601111-001     2.4473759918-001

COMPUTED L-MATRIX(U=-L*X)

  -2.2401202223-001     4.8947519836-001


     TD=    3.00000

COMPUTED S-MATRIX

   2.3675225470-001    -3.3748215454-002
  -3.3748215456-002     4.8106914455-003

COMPUTED L-MATRIX(U=-L*X)

  -6.7496430915-002     9.6213828910-003


     TD=    4.00000

COMPUTED S-MATRIX

   9.4832309335-002     1.0979886602-001
   1.0979886602-001     1.2712746388-001

COMPUTED L-MATRIX(U=-L*X)

   2.1959773204-001     2.5425492777-001
```

```
     TD=    5.00000

COMPUTED S-MATRIX

   1.2829097593-002    -4.3368956915-002
  -4.3368956911-002     1.4660940959-001

COMPUTED L-MATRIX(U=-L*X)

  -8.6737913822-002     2.9321881919-001


     TD=    6.00000

COMPUTED S-MATRIX

   1.2684241116-001    -3.6911926962-002
  -3.6911926957-002     1.0741599275-002

COMPUTED L-MATRIX(U=-L*X)

  -7.3823853916-002     2.1483198551-002


     TD=    7.00000

COMPUTED S-MATRIX

   7.5735057698-002     6.5999163273-002
   6.5999163275-002     5.7514837711-002

COMPUTED L-MATRIX(U=-L*X)

   1.3199832655-001     1.1502967542-001


     TD=    8.00000

COMPUTED S-MATRIX

   2.3152200131-003    -1.5742828034-002
  -1.5742828033-002     1.0704668798-001

COMPUTED L-MATRIX(U=-L*X)

  -3.1485656066-002     2.1409337597-001


     TD=    9.00000

COMPUTED S-MATRIX

   8.0011456267-002    -3.6190434585-002
  -3.6190434585-002     1.6369500275-002
```

```
COMPUTED L-MATRIX(U=-L*X)

  -7.2380869173-002    3.2739000552-002


    TD=  10.00000

COMPUTED S-MATRIX

   6.6774714544-002    4.3294109199-002
   4.3294109196-002    2.8070204476-002

COMPUTED L-MATRIX(U=-L*X)

   8.6588218394-002    5.6140408951-002
```