AT03-76ER 70019

July 1969

MASTER

USCEE-370

Technical Report

TERMINAL-STATE DYNAMIC PROGRAMMING: QUADRATIC
COSTS, LINEAR DIFFERENTIAL EQUATIONS

David C. Collins

Sponsored by the Atomic Energy Commission
under Contract No. AT(11-1)-113, Proj. #19.

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED
MGW

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

TERMINAL-STATE DYNAMIC PROGRAMMING: QUADRATIC COSTS, LINEAR DIFFERENTIAL EQUATIONS

1. A Terminal Control Problem

A fairly general class of control problems can be posed in terms of minimizing a cost functional involving the state of the system to be controlled and the control exerted over a fixed interval of time [3, 4]. The state and control variables are related by a state equation, often with additional constraints of various forms upon the control or state. That is, we desire to find

$$(1) \quad \min_{y(t) \in Y} \left\{ p(x(T)) + \int_0^T q(x(t), y(t)) dt \right\},$$

where x and y are the N -dimensional state vector and the M -dimensional control vector, respectively, while p and q are scalar-valued functions of their arguments. The state equation is

$$(2) \quad \dot{x} = h(x, y), \quad x(0) = c,$$

with y constrained to some class of admissible controls, Y .

In this paper we wish to discuss a specialization of (1), a terminal control problem, in which the cost of state is measured only at the terminal time, T . We further assume that p and q are homogeneous quadratic functions of their arguments, that the state equation (2) is linear, and that there are no constraints on the control vector y . We thus arrive at the following terminal control problem:

$$(3) \quad \min_{y(t)} \left\{ \langle x(T), Px(T) \rangle + \int_0^T \langle y(t), Qy(t) \rangle dt \right\},$$

with x and y related by

$$(4) \quad \dot{x} = Ax + By, \quad x(0) = c,$$

where A and B are $N \times N$ and $N \times M$ dimensional matrices while Q and P are symmetric matrices of order $M \times M$ and $N \times N$ respectively; Q is positive definite and P is positive semidefinite. The notation $\langle \cdot, \cdot \rangle$ represents the usual finite-dimensional Euclidean inner product.

2. The Calculus of Variations

This problem may be approached via a straightforward application of the calculus of variations [6, 7] to the minimization of the functional

$$(1) \quad J(x, y) = \langle x(T), Px(T) \rangle + \int_0^T \left(\langle y(t), Qy(t) \rangle + \langle \lambda(t), Ax(t) + By(t) - \dot{x}(t) \rangle \right) dt,$$

where $\lambda(t)$ is the N -dimensional vector of Lagrange multipliers. This approach yields the system of equations

$$(2) \quad \dot{x} = Ax - \frac{1}{2} BQ^{-1} B' \lambda,$$

$$(3) \quad \dot{\lambda} = -A' \lambda,$$

subject to the two-point boundary conditions

$$(4) \quad x(0) = c,$$

$$(5) \quad 2Px(T) - \lambda(T) = 0.$$

The y and λ vectors are linearly related via

$$(6) \quad y = -\frac{1}{2} Q^{-1} B' \lambda.$$

One can solve the matrix system

$$(7) \quad \dot{X} = AX + D\lambda,$$

$$(8) \quad \dot{\Lambda} = -A'\Lambda,$$

and write any solution $\lambda(t)$, $x(t)$ of (2) and (3) as

$$(9) \quad x(t) = X^{(1)} c^{(1)} + X^{(2)} c^{(2)},$$

$$\lambda(t) = \Lambda^{(1)} c^{(1)} + \Lambda^{(2)} c^{(2)},$$

where $X^{(1)}$, $\Lambda^{(1)}$, $X^{(2)}$, $\Lambda^{(2)}$ are the $N \times N$ matrix solutions of (7) subject to the initial conditions $X^{(1)}(0) = I$, $\Lambda^{(1)}(0) = 0$, $X^{(2)}(0) = 0$, $\Lambda^{(2)}(0) = I$. It remains to solve for $c^{(1)}$ and $c^{(2)}$. Clearly $c^{(1)} = c$, $\Lambda^{(1)}(t) \equiv 0$, and $c^{(2)}$ can be found by solving the linear system resulting from the application of (5),

$$(10) \quad [\Lambda^{(2)}(T) - 2PX^{(2)}(T)]c^{(2)} = 2PX^{(1)}(T)c.$$

The system (7) and (8) is often unstable (we note however that (8) may be solved independently of (7) and in the direction of decreasing t). Furthermore, when errors made in the numerical solution of (7) and (8) and thus in the values of $X^{(1)}(T)$ and $X^{(2)}(T)$ are combined with the fact that the algebraic system (10) may be ill-conditioned, it is reasonable to expect significant errors in the calculation, via (10), of $c^{(2)}$. If we could determine $c^{(2)}$ accurately, however, we could use it as the missing initial condition in (2) and (3) and, subject to stability considerations, solve the system (2) and (3) for the optimal state and control trajectories. We note that this is essentially an open-loop control since once $\lambda(0)$ is determined the solution of $\lambda(t)$ and thus $y(t)$ is determined independently of $x(t)$.

We turn now to another approach that often avoids the problem of instability and the hazardous task of solving a system of linear algebraic equations.

3. Conventional Dynamic Programming

In this section we derive, formally, a matrix Riccati equation, the solution of which can be used to generate the optimal control. To this end we consider the problem formulated in (1.3) and (1.4) as one member of a family of similar processes with various initial states, $x(a)$, and various initial times, a . We define a function of these two parameters, the optimal cost function, by means of the expression

$$(1) \quad f(x(a), a) = \min_{\substack{y(t) \\ a \leq t \leq T}} \left\{ \langle x(T), P x(T) \rangle + \int_a^T \langle y(t), Q y(t) \rangle dt \right\}.$$

The optimal cost of the problem posed in (1.3) and (1.4) is $f(c, 0)$. We obtain in the usual fashion [2, 3]

$$(2) \quad -\frac{\partial f}{\partial a} = \min_{y(a)} \left\{ \langle y(a), Q y(a) \rangle + \left\langle \frac{\partial f(x(a), a)}{\partial x}, A x(a) + B y(a) \right\rangle \right\},$$

where

$$\frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{pmatrix} \bigg|_{(x(a), a)}$$

Since the control $y(a)$ is unconstrained, we may perform the minimization directly yielding

$$(3) \quad y(a)_{\min} = -\frac{1}{2} Q^{-1} B' \frac{\partial f}{\partial x}.$$

Substitution of the optimal control (3) in (2) results in a partial differential equation

$$(4) \quad \frac{\partial f}{\partial a} = \frac{1}{4} \left\langle \frac{\partial f}{\partial x}, B Q^{-1} B' \frac{\partial f}{\partial x} \right\rangle - \left\langle \frac{\partial f}{\partial x}, A x(a) \right\rangle.$$

We separate variables in (4) by using the fact that $f(x(a), a)$ is a quadratic function of $x(a)$:

$$(5) \quad f(x(a), a) = \langle x(a), R(a)x(a) \rangle,$$

where $R(a)$ is a symmetric $N \times N$ matrix.

The quadratic form (5) is suggested directly by the quadratic nature of the cost functional (1.3) and the linear dependence of the optimal control on the initial condition $x(a)$ (see (2.2), (2.3), and (2.6)). Substitution of (5) in (4) yields

$$\langle x(a), \frac{dR}{da} x(a) \rangle = \langle x(a), RBQ^{-1}B'Rx(a) \rangle - \langle x(a), RAx(a) \rangle - \langle x(a), A'Rx(a) \rangle.$$

Since this equation must be satisfied for all $x(a)$, we obtain a Riccati equation for $R(a)$,

$$(6) \quad \frac{dR(a)}{da} = R(a)BQ^{-1}B'R(a) + R(a)A - A'R(a).$$

The initial condition on (6) is from (1),

$$(7) \quad R(T) = P,$$

and the optimal control in terms of $R(a)$ and $x(a)$ is, from (3),

$$(8) \quad y(a)_{\min} = -Q^{-1}B'R(a)x(a).$$

To solve the original problem (1.3), (1.4), we must now solve system of N^2 (or $(N(N+1))/2$ if use is made of the symmetry of $R(a)$) simultaneous nonlinear ordinary differential equations, (6), subject to initial condition, (7), from $a = T$ to $a = 0$. The algorithm for acquiring the optimal state and control trajectories, $x(t)$ and $y(t)$ over $0 \leq t \leq T$,

consists of using $R(0)$ and the known initial condition on x , $x(0) = c$, in (8) to yield the initial optimal control $y(0)$. This control employed in the state equation (1.4) produces the next optimal state $x(\Delta)$. The reconstruction proceeds in this fashion from $t = 0$ to $t = T$. It can be shown that (6) is stable when solved in the direction of decreasing a . No solution of a linear system of algebraic equations is required. We have thus avoided two of the disadvantages of the calculus of variations approach.

4. Computational Aspects

In this section we describe some of the crucial steps involved in implementing the above algorithm on a digital computer. This is done both to clarify the algorithm and some of its variants as well as to provide a basis for comparison with a different, terminal state algorithm to be discussed subsequently. The algorithm divides into two parts: the backward solution of (3.6) subject to (3.7) for $R(a)$, $0 \leq a \leq T$, and the forward solution of the state equation (1.4) with the control specified in terms of $R(a)$ and $x(a)$ by (3.8) from $a = 0$ to $a = T$. The latter solution generates the optimal control and state trajectories and is termed "reconstruction."

4.1. Backward Solution. We are particularly interested in the control of large systems; that is systems where the dimension of the state vector is very large (e.g. $N = 1000$). In this situation the N^2 simultaneous non-linear differential equations cannot be solved without recourse to what we will call "slow memory" (e.g. disk or tape storage) since few current computers have rapid access storage on the order of 10^6 words. The $N \times N$ matrix R and A must then reside in slow memory and be brought into core, vector by vector, as needed. Assuming a simple Euler integration of (3.6), we compute the (i, j) -th component of $R(a)$ using the relation

$$R_{i,j}(a) = R_{i,j}(a+\Delta) - \Delta \left(\sum_{k=1}^N [R_{i,k}(a+\Delta)R_{k,j}(a+\Delta) - R_{i,k}(a+\Delta)A_{k,j} - A_{k,i}R_{k,j}] \right).$$

That is we must bring the four N-dimensional vectors (row i and column j of R, column i and column j of A) from disk to core. This procedure must be repeated for all N^2 (or $(N(N+1))/2$) components of $R(a)$. Thus for each step in a, on the order of $4N^2$ "references" must be made to slow memory. Dividing the time interval $(0, T)$ into D subintervals ($D = T/\Delta$) the total number of references to slow storage will be $4DN^2$.

4.2. Reconstruction. Having solved (3.6) from $a = T$ to $a = 0$, the matrix $R(0)$ and known initial condition $x(0)$ are used in (3.8) to compute the initial optimal control and (1.4) is integrated forward one step to yield $x(\Delta)$. At this point we need $R(\Delta)$ to compute $y(\Delta)$. Since (3.6) is generally unstable when solved in the direction of increasing a, we must either resolve (3.6) from $a = T$ to $a = \Delta$ or else acquire $R(\Delta)$ from slow memory where it would have to have been previously stored during the backward solution of (3.6). That is, we have the option of storing $R(t)$ over the interval $(0, T)$ or recomputing $R(t)$ for each forward step of (1.4).

In whatever way $R(t)$ is acquired, the forward solution of

$$(1) \quad \frac{dx(t)}{dt} = [A - BQ^{-1}B'R(t)]x(t), \quad x(0) = c,$$

will require the retrieval of rows of the $N \times N$ matrices A , $[BQ^{-1}B']$, and R from slow memory (assuming Q and B are constant matrices).

At each step $3N$ references must be made to slow memory or a total of $3ND$ references for the complete forward solution from 0 to T.

If $R(t)$ is to be recomputed for each forward step, the total number of references to slow memory increases from $4DN^2 + 3ND$ to $\frac{4D(D+1)}{2}N^2 + 3ND$; on the order of $2D^2N^2$. On the other hand if $R(t)$ is stored in slow memory during the first backward solution of (3.6), the number of references to slow memory remains on the order of $4DN^2$ but the amount of slow memory required increases from approximately $3N^2$ to $3N^2 + DN^2$.

A significant reduction of the amount of slow storage required to store $R(t)$ over $(0, T)$ is effected if the B matrix has many rows composed entirely of zeros. As a particular example, take $M = 1$ and

$$(2) \quad B = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ b_{N1} \end{pmatrix}.$$

This sort of B matrix occurs when an N -th order scalar differential equation is converted to a first order vector-matrix system. In physical terms $M = 1$ implies there is only one control function while the zero rows in B imply that only the time derivative of the N -th component of the state vector is affected directly by this control (it should be noted that the $N \times N$ matrix $[BQB']$ will not in this case require $N \times N$ storage but rather only two words for b_{N1} and q_{11}). Since in (1) the $N \times N$ matrix $R(t)$ is premultiplied by B' the resulting $M \times N$ product will contain components only from the N -th row of $R(t)$. For this example,

$$B'R = b_{N1}(r_{N1}, r_{N2}, \dots, r_{NN}).$$

Thus only the last row of $R(t)$ need be stored in order to perform the reconstruction. In general if the first $N - I$ rows of B are composed entirely of zeros, then only the last I rows of R need be stored and/or, if M is sufficiently small, only the $M \times N$ product $B'R$, whichever is less.

The following charts (see Tables 1 and 2) summarize the foregoing comments. Particular representative values for N , M , I , and D are specified and the resulting values for storage are shown in parenthesis. The formulas and numbers represent approximate totals for both the backward solution and the reconstruction.

5. Reduction of Dimensionality

In many control problems of interest the terminal cost, $\langle x(T), Px(T) \rangle$, may not involve all of the components of the N -dimensional state vector. The

Table 1

 $N = 1000, D = 1000 \quad (T = 1.0, \Delta = .001)$
 N Sufficiently Large to Require Use of Slow Storage for $N \times N$ Matrices

	<u>$R(t)$ not stored over $(0, T)$</u>		<u>$R(t)$ stored over $(0, T)$</u>	
References to slow memory	$2N^2 D(D+1)$	$(2 \cdot 10^{12})$	$4N^2 D$	$(4 \cdot 10^9)$
Slow storage required	$3N^2$	$(4 \cdot 10^6)$	$3N^2 + DN^2$	(10^9)
Core storage ($M=1000, I=1000, B$ general)	$4N$	$(4 \cdot 10^3)$	$4N$	$(4 \cdot 10^3)$
References to slow memory	$2N^2 D(D+1)$	$(2 \cdot 10^{12})$	$4N^2 D$	
Slow storage required	$2N^2$	$(2 \cdot 10^6)$	$2N^2 + DNI$ or $2N^2 + MND$	$(3 \cdot 10^6)$
Core storage required ($M=1, I=1, B$ special)	$4N$	$(4 \cdot 10^3)$	$4N$	$(4 \cdot 10^3)$

Table 2

 $N = 100, D = 1000 \quad (T = 1.0, \Delta = .001)$
 N Sufficiently Small to Permit the Storage of a Few $N \times N$ Matrices
in Rapid Access Storage

	<u>$R(t)$ not stored over $(0, T)$</u>		<u>$R(t)$ stored over $(0, T)$</u>	
Core required	$3N^2$	$(4 \cdot 10^4)$	$3N^2 + DN^2$	(10^2)
Time estimate ($M=100, I=100, B$ general)	$t_1 \frac{D(D+1)}{2}$	$(5t_1 \cdot 10^5)$	$2Dt_1$	$(2t_1 \cdot 10^3)$
Core required	$3N^2$	$(3 \cdot 10^4)$	$3N^2 + DNI$ or $3N^2 + MND$	(10^5)
Time estimate ($M=1, I=1, B$ special)	$t_1 \frac{D(D+1)}{2}$	$(5t_1 \cdot 10^5)$	$2Dt_1$	$(2t_1 \cdot 10^3)$

 $*t_1 = \text{time to solve (3.6) from } t + \Delta \text{ to } t.$

absence of a terminal cost attached to component $x_i(T)$ implies that both row i and column i of P consist entirely of zeros. Suppose that only L , $L \leq N$, components of $x(T)$ are involved in the terminal cost. Then by rearranging the state components (with appropriate changes in A and P), we can, without loss of generality, rewrite the original problem as that of finding the

$$(1) \quad \min_{\substack{y(t) \\ 0 \leq t \leq T}} \left\{ \langle \hat{x}(T), \hat{P}\hat{x}(T) \rangle + \int_0^T \langle y(t), Qy(t) \rangle dt \right\},$$

where $\hat{x}$ is an L -dimensional vector made up of the first L components of the rearranged state vector x , and $\hat{P}$ is the $L \times L$ matrix formed by deleting rows and columns of P composed entirely of zeros.

A question of some interest is the following: If $\hat{x}$ is of dimension $L \ll N$ (e.g., $L = 1$ or 2 , $N = 1000$) is it possible to reduce the dimension of the Riccati equation (3.6) from $N \times N$ to $L \times L$? Such a reduction would significantly enhance the feasibility of solving terminal control problems of high dimension.

6. A Green's Function

Since the state equation (1.4) is linear, we may express $x(T)$ as the superposition of two effects: the effect on $x(T)$ due to the system having been in state $x(a)$ at time a , and the effect on $x(T)$ due to the operation of control over the intervening period, $a \leq t \leq T$ ([1,9]). That is,

$$(1) \quad x(T) = K(T, a)x(a) + \int_a^T K(T, s)By(s)ds.$$

$K(T, a)$, the $N \times N$ fundamental matrix, can be found by fixing a and solving the linear equation

$$(2) \quad \frac{\partial K(t, a)}{\partial t} = A(t)K(t, a),$$

from $t = a$ to $t = T$, using initial condition

$$(3) \quad K(a, a) = I_N,$$

where I_N is the $N \times N$ identity matrix. Alternatively, since $K(T, t) = K^{-1}(t, T)$ we may acquire $K(T, a)$ by fixing T and solving

$$(4) \quad \frac{\partial K'(T, t)}{\partial t} = -A'(t)K'(T, t)$$

subject to

$$(5) \quad K'(T, T) = I_N,$$

from $t = T$ to $t = a$.

We note in passing that if (2) is stable for increasing t then (4) is stable when solved in the direction of decreasing t . Additionally, if A is a constant matrix, then $K(T, a) = K(T - a)$, a function of only one variable, $(T - a)$.

Since we will ultimately only be interested in the first L components of $x(T)$, the vector $\hat{x}(T)$, we note that we may write

$$(6) \quad \hat{x}(T) = \hat{K}(T, a)x(a) + \int_a^T \hat{K}(T, s)By(s)ds,$$

where $\hat{K}$ is an $L \times N$ matrix made up of the first L rows of K .

7. Terminal State Dynamic Programming

In Sec. 3 we developed a dynamic programming approach to the solution of the terminal control problem by imbedding the original problem in a family of similar problems of varying initial state, $x(a)$, and initial time, a . Rather than imbed in terms of initial state, we chose here to imbed in terms of the effect of initial state upon the terminal state, $\hat{x}(T)$. This imbedding was suggested by Bellman et al [10] and has been utilized previously to some extent by Kramer [8] and Collins [5]. Using (6.6) we define a new L -dimensional variable, $e(a)$, to represent this effect:

- (1) $e(a) = \hat{K}(T, a)x(a)$ = the effect on (contribution to) $\hat{x}(T)$ due to the system having been in state $x(a)$ at time a .

If $L = N$ then the relationship (1) between e and x is one-to-one, since $\hat{K}$ is then $N \times N$ and $K(T, a)^{-1}$ exists. However, if $L < N$ the relationship (1) is not one to one; for a particular $x(a)$ there exists a unique $e(a)$ but not conversely.

It is clear, however, that we may define a new minimum cost function, $g(e(a), a)$, in terms of this new variable,

$$(2) \quad f(x(a), a) = g(e(a), a) = \min_{\substack{y(t) \\ a \leq t \leq T}} \left\{ \langle \hat{x}(T), P\hat{x}(T) \rangle + \int_a^T \langle y(t), Qy(t) \rangle dt \right\}.$$

That is, the minimum cost function of the family of processes can equally well be considered a function of the effect of initial state and the initial time. To exhibit this dependence explicitly we use (6.6) to substitute for $\hat{x}(T)$ in (2),

$$(3) \quad f(x(a), a) = g(e(a), a) = \min_{\substack{y(t) \\ a \leq t \leq T}} \left\{ \langle e(a) + \int_a^T \hat{K}(T, s)By(s)ds, P(\cdot) \rangle + \int_a^T \langle y(t), Qy(t) \rangle dt \right\},$$

where $(\cdot)$ is meant to represent the left-hand side of the inner product repeated. The minimum cost of the problem whose solution is sought, (1.3), in this notation is

$$f(c, 0) = g(e(0), 0) = g(\hat{K}(T, 0)c, 0).$$

We may now proceed to apply the dynamic programming formalism [3] to relate the minimum costs of members of the family of processes. We break the integral $\int_a^T$ into two parts, $\int_a^{a+\Delta}$ and $\int_{a+\Delta}^T$, and apply the

mean-value theorem to the integral $\int_a^{a+\Delta}$. Thus (3) can be rewritten as

$$(4) \quad g(e(a), a) = \min_{y(a)} \left\{ \Delta \langle y(a), Qy(a) \rangle + o(\Delta^2) \right. \\ \left. + \left[\min_{\substack{y(t) \\ a+\Delta \leq t \leq T}} \left\{ \langle e(a) + \Delta \hat{K}(T, a)By(a) + o(\Delta^2) \right. \right. \right. \\ \left. \left. + \int_{a+\Delta}^T \hat{K}(T, s)By(s)ds, P(\cdot) \rangle + \int_{a+\Delta}^T \langle y(t), Qy(t) \rangle dt \right\} \right] \right\}.$$

The term in brackets in (4) is a process similar to (2) but of duration Δ less and with an altered "effect of initial state" due to the change in state caused by $y(a)$. We have then that

$$(5) \quad g(e(a), a) = \min_{y(a)} \{ \Delta \langle y(a), Qy(a) \rangle + g(e(a + \Delta), a + \Delta) + o(\Delta^2) \},$$

where

$$(6) \quad e(a + \Delta) = e(a) + \Delta \hat{K}(T, a)By(a) + o(\Delta^2).$$

One way to verify that (6) is consistent with (1) is to differentiate (1) with respect to a ,

$$(7) \quad \frac{de(a)}{da} = \frac{d\hat{K}(T, a)}{da}x(a) + \hat{K}(T, a) \frac{dx(a)}{da}.$$

Using (6.4) we have

$$\frac{d\hat{K}(T, a)}{da} = -\hat{K}(T, a)A(a),$$

and using (1.4) to substitute for $dx(a)/da$, (7) becomes

$$(8) \quad \frac{de(a)}{da} = \hat{K}(T, a)By(a),$$

which can be expressed as (6).

If we assume that g is sufficiently differentiable, we may expand $g(e(a + \Delta), a + \Delta)$ about $(e(a), a)$. After simplifying the resulting expression, dividing by Δ and letting $\Delta \rightarrow 0$, we arrive at a partial differential equation similar to (3.2),

$$(9) \quad -\frac{\partial g}{\partial a} = \min_{y(a)} \left\{ \langle y(a), Qy(a) \rangle + \left\langle \frac{\partial g}{\partial e}, \hat{K}(T, a)By(a) \right\rangle \right\},$$

where

$$\frac{\partial g}{\partial e} = \begin{pmatrix} \frac{\partial g}{\partial e_1} \\ \vdots \\ \frac{\partial g}{\partial e_L} \end{pmatrix} \bigg|_{e(a), a}$$

Since $y(a)$ is not constrained, we can minimize (9) directly,

$$(10) \quad y(a)_{\min} = -\frac{1}{2} Q^{-1} B' \hat{K}'(T, a) \frac{\partial g}{\partial e}.$$

Substituting this optimal control in (9) yields the partial differential equation

$$(11) \quad \frac{\partial g}{\partial a} = \frac{1}{4} \left\langle \frac{\partial g}{\partial e}, \hat{K}(T, a) B Q^{-1} B' \hat{K}'(T, a) \frac{\partial g}{\partial e} \right\rangle.$$

The fact that $f(x(a), a)$ is quadratic in $x(a)$, (3.5), implies that $g(e(a), a)$ is quadratic in $e(a)$ since $e(a)$ and $x(a)$ are linearly related. We assume that

$$(12) \quad g(e(a), a) = \langle e(a), G(a)e(a) \rangle,$$

where $G(a)$ is a symmetric $L \times L$ matrix. Using this form to separate variables in (11), we arrive at an ordinary differential equation for G ,

$$(13) \quad \frac{dG(a)}{da} = G(a) \hat{K}(T, a) B Q^{-1} B' \hat{K}'(T, a) G(a),$$

with initial condition

$$(14) \quad G(T) = \hat{P}.$$

The initial condition on G is deduced from the fact that since $\hat{K}(T, T) = [I_L, 0]$, (1) implies $e(T) = \hat{x}(T)$ and using (2) at $a = T$,

$$g(e(T), T) = \langle \hat{x}(T), P\hat{x}(T) \rangle = \langle e(T), \hat{P}e(T) \rangle,$$

from which (14) follows.

The optimal control can be written in terms of $G(a)$ and the L -dimensional $e(a)$ (see (10) and (12))

$$(15) \quad y(a)_{\min} = -Q^{-1}B'\hat{K}'(T, a)G(a)e(a).$$

Recalling that $e(a) = \hat{K}(T, a)x(a)$, we can also write the optimal control in terms of the N -dimensional vector $x(a)$,

$$(16) \quad y(a)_{\min} = -Q^{-1}B'\hat{K}'(T, a)G(a)\hat{K}(T, a)x(a).$$

We can now proceed as suggested in Sec. 3. Equation (13) is solved from $a = T$ to $a = 0$ for $G(0)$. The initial control can be computed using $x(0) = c$ and (16) at $a = 0$.

8. Reduction of Dimensionality Revisited

We have achieved the previously outlined objective. The $N \times N$ matrix Riccati equation (3.6) has been reduced to an $L \times L$ matrix equation (7.13) under the assumption that the $N \times N$ terminal cost matrix, P , can be reduced to an $L \times L$ matrix, $\hat{P}$, by deleting corresponding rows and columns composed entirely of zeros. If $L \ll N$, we have apparently realized a significant reduction in the number of simultaneous nonlinear equations to be solved. However, we have not, in general, reduced the storage requirements proportionately. Rather we will find that we have reduced the "dimensionality" of the problem by a factor of N , except in

those problems where A , B and Q exhibit some special, exploitable structure; in which case a reduction from N^2 to L^2 is truly achieved. In this section we will examine in some detail the computation implications of solving the terminal control problem using (7.13) rather than (3.6). We simplify and orient the following discussion by assuming that L is very small (e.g., $L = 1$ or 2) and that A , B , and Q are constant matrices. We further suppose that N is sufficiently large so that an $N \times N$ array will exceed the rapid-access storage of the computer.

8.1. Backward Solution. While the right-hand side of (7.13), taken as a whole, is an $L \times L$ matrix, it involves the $L \times N$ matrix $\hat{K}(T, a)$ which must be acquired by solving (see 6.4)

$$(1) \quad \frac{\partial \hat{K}'(T, t)}{\partial t} = -A' \hat{K}'(T, t), \quad \hat{K}'(T, T) = \begin{pmatrix} I & K \\ 0 & \end{pmatrix}.$$

The A matrix is of order $N \times N$ and thus must reside in slow storage and be brought into core row by row as (1) is solved. Likewise the $N \times N$ matrix $[BQ^{-1}B']$ will correspondingly complicate the solution of (7.13). We use simple Euler integration of (1) and (7.13) and assume that there is sufficient core storage to accommodate $\hat{K}(T, a + \Delta)$ as well as a few N -dimensional vectors. It is easily shown that N references to slow memory are required to compute $\hat{K}(T, a)$ and an additional N references to bring in the $N \times N$ matrix $[BQ^{-1}B']$ and acquire $G(a)$ from $G(a + \Delta)$. Thus if the interval $(0, T)$ is divided into D steps, we will require $2DN$ references to slow memory to solve (7.13) from $t = T$ to $t = 0$. This compares favorably with $4DN^2$ required when (3.6) is solved.

8.2. Reconstruction. There are several ways to reconstruct the optimal control and state trajectories using the solution of (7.13). We mention two: an "open loop control" in which $y(t)$ is independent of $x(t)$ for all $t > 0$ and a "closed loop control" in which $y(t)$ depends upon $x(t)$ for all t .

The open loop control can be achieved by employing the optimal control in terms of $e(a)$, (7.15), in the differential equation for $e(a)$, (7.8), yielding

$$(2) \quad \frac{de(t)}{dt} = -\hat{K}(T, t)BQ^{-1}B'\hat{K}'(T, t)G(t)e(t).$$

The initial condition is $e(0) = \hat{K}(T, 0)c$. Using the stored or recomputed $\hat{K}(T, t)$ and $G(t)$, (2) is solved from $t = 0$ to $t = T$. With $e(t)$ known, substitution of (7.15) for $y(t)$ in the state equation (1.4) yields

$$(3) \quad \frac{dx(t)}{dt} = Ax(t) - BQ^{-1}B'\hat{K}'(T, t)G(t)e(t), \quad x(0) = c,$$

which is solved for the optimal state trajectory. We note that except for the initial state, $x(0)$, the optimal control

$$(4) \quad y(t) = -Q^{-1}B'\hat{K}'(T, t)G(t)e(t)$$

does not depend on the optimal state trajectory. We will examine the computational feasibility of this algorithm subsequently.

A closed loop control, dependent upon the optimal state trajectory is found by employing (7.16), which expresses $y(t)$ in terms of $x(t)$, in the state equation (1.4),

$$(5) \quad \frac{dx(t)}{dt} = [A - BQ^{-1}B'\hat{K}'(T, t)G(t)\hat{K}(T, t)]x(t), \quad x(0) = c.$$

Solving (5) from $t = 0$ to $t = T$ produces the optimal state and control trajectories. Again we may either recompute $G(t)$ and $\hat{K}(T, t)$ as needed when solving (5) or we may store these previously computed arrays in slow memory. If we choose to recompute these matrices, we find that the forward solution of (5) will require a total of $ND(D + 1)$ (or on the order of ND^2) references to slow storage. The total slow storage required is on the order of $2N^2$ (for A and $[BQ^{-1}B']$), while the necessary core storage is on the order of $(L + 4)N$.

We find a more significant reduction of the size of the slow memory when we consider storing the previously computed $G(t)$ and $\hat{K}(T, t)$ over $(0, T)$. This storage requirement is $DL^2 + DLN + 2N^2$ which compares favorably with $DN^2 + 3N^2$ required for the conventional dynamic programming approach. With $K(T, t)$ and $G(t)$ stored the reconstruction of the optimal trajectories using (5) will require only $2N$ references to slow memory per step (for A and $[BQ^{-1}B]$) or $2ND$ for complete reconstruction. This information is summarized in Table 3.

Table 3

$D = 1000$ ($T = 1.0$, $\Delta = .001$), $L = 1$

N Sufficiently Large to Require Use of Slow Storage for $N \times N$ Matrices

	$G(t)$ and $\hat{K}(T, t)$ not stored over $(0, T)$	$G(t)$ and $\hat{K}(T, t)$ stored over $(0, T)$
References to slow memory	ND^2 (10^9)	$4ND$ (2×10^6)
Slow storage required	$2N^2$ (2×10^6)	$DL^2 + DLN + 2N^2$ (3×10^6)
Core storage required	$(L+4)N$ (5×10^3)	$(L+4)N$ (5×10^3)
$(N=1000, M=1000, I=1000, B \text{ general})$		

N Sufficiently Small to Permit Storage of a Few $N \times N$ Matrices
in Rapid Access Memory

	$G(t)$ and $\hat{K}(T, t)$ not stored over $(0, T)$	$G(t)$ and $\hat{K}(T, t)$ stored over $(0, T)$
Core required	$2N^2$ (2×10^4)	$DL^2 + DLN + 2N^2$ (10^5)
Time estimates	$\frac{t_2^* D(D+1)}{2}$ ($5 \times 10^5 t_2$)	$2Dt_2$ ($10^3 t_2$)
$(N=100, M=100, I=100, B \text{ general})$		

* t_2 = time to solve (7.13) from $t + \Delta$ to t .

Note $t_2 \approx t_1/N$ due to reduced dimension of (7.13) relative to (3.6).

8.3. Special Cases. We have seen that the terminal state imbedding results in the reduction, by a factor of N , of the number of references to slow storage and/or the amount of slow storage required and/or the time required for solution. This is a significant reduction but falls short of the reduction of order N^2 , from $N \times N$ to $L \times L$, that we hoped to achieve. Inspecting (3.6) and (7.13), however, we note that the $N \times N$ storage requirements of (3.6) are implicit in the equation for the unknown $R(a)$ while the $N \times N$ storage requirements of (7.13) enter solely via the known matrices A , B , and Q . It is clear then that if these matrices were structured so as to require less than $N \times N$ storage a further reduction in storage and/or time of solution would be possible. The following examples illustrate that such fortuitously structured matrices occur in problems of interest. Note that in these examples A and B can be stored as vectors of N (or fewer) numbers in addition to a few lines of code to specify the systematic placement of these elements.

The first example, already alluded to in Sec. 4, corresponds to the reduction of an N -th order scalar differential equation,

$$u^{(N)} + a_{N-1} u^{(N-1)} + \cdots + a_0 u = y(t),$$

to a system of first-order differential equations

$$\frac{dx}{dt} = Ax + By, \quad x_i = u^{(i-1)},$$

where

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -a_0 & -a_1 & \cdots & -a_{N-1} \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix},$$

and M , the dimension of the y vector, is one.

As a second example consider the transformation of the heat equation

$$\frac{\partial u(z, t)}{\partial t} = c_1 \frac{\partial^2 u(z, t)}{\partial z^2} + c_1 v(z, t),$$

to a system of ordinary differential equations

$$\frac{dx}{dt} = Ax + By, \quad x_i(t) = u(z_i, t), \quad y_i(t) = v(z_i, t), \quad \delta = z_{i+1} - z_i,$$

by discretizing the z variable and using a central difference approximation for the second derivative:

$$\frac{\partial^2 u(z_i, t)}{\partial z^2} \approx \frac{u(z_{i+1}, t) - 2u(z_i, t) + u(z_{i-1}, t))}{\delta^2},$$

where

$$A = \frac{c_1}{\delta^2} \begin{bmatrix} -2 & 1 & 0 & \cdots \\ 1 & -2 & 1 & \cdots \\ 0 & 1 & -2 & 1 & \cdots \\ \vdots & & & \end{bmatrix}, \quad B = c_1 I_N.$$

Consider the latter example. All information about A , B , and Q (the Q matrix for the associated cost functional will be diagonal) can comfortably be retained in core storage. If reconstruction is performed without storage of $\hat{K}$ then no references to slow storage need be made.

On the other hand if reconstruction is performed with the $L \times N$ matrix, $\hat{K}$ stored in slow storage over $(0, T)$, a total of only $2D$ references to slow storage need be made.

The same comments hold for the first example. Here, however, we note that the control matrix B has a particularly simple form which was used to advantage in Sec. 4. We note that it is the $L \times N$ matrix $\hat{K}(T, t)$ which consumes slow storage rather than the $L \times L$ matrix, $G(a)$.

Consider (7.13) and the open loop reconstruction (8.2) and (8.3). We note that $\hat{K}$ always appears post-multiplied by B , or $\hat{K}'$ pre-multiplied by

B' , which is not the case for the closed loop reconstruction (8.5). If the $N \times M$ matrix B has its first $N-I$ rows composed entirely of zeros the $L \times M$ product $\hat{K}B$ will only involve elements of the last I columns of $\hat{K}$. We will still need the entire $\hat{K}(T, 0)$ to compute the initial $e(0)$ and apparently it is necessary to solve for an entire column (or row if (6.2) is used) of $\hat{K}(T, a)$ in order to acquire any particular element. However, a large saving occurs in the amount of storage needed for the reconstruction; that is either only the last I columns of $\hat{K}$ need be stored (requiring $D \cdot L \cdot I$ words) or the $L \times M$ product $\hat{K}B$ need be stored (requiring $D \cdot L \cdot M$ words). If LM and/or I are sufficiently small the entire reconstruction can be performed using only core storage and without reiteration of (7.13).

One should, however, use the open loop reconstruction with caution. It is felt that in general the closed loop reconstruction is preferable since the inevitable numerical errors made in the backward solution of (7.13) and the forward solution of (8.5) will tend to be corrected for by the closed loop calculation of the control, whereas the open loop control (8.2) and (8.3) do not admit this possibility.

Table 4

 $D = 1000$ ($T = 1.0$, $\Delta = .001$), $L = 1$
 N Sufficiently Large to Require Use of Slow Storage for $N \times N$ Arrays

	G(t) and $\hat{K}(T, t)$ not stored over (0, T)		G(t) and $\hat{K}(T, t)$ stored over (0, T)	
	Example 1	Example 2	Example 1	Example 2
References to slow storage	0 (0)	0 (0)	0 (0)	$2 \cdot D$ (2×10^3)
Slow storage required	0 (0)	0 (0)	0 (0)	$K^2 D + D \cdot L \cdot N$ (10^6)
Core storage required	$L \cdot N$ (10^3)	$L \cdot N$ (10^3)	$D \cdot L \cdot M + D \cdot L^2$ (2×10^3) or $D \cdot L \cdot I$ (2×10^3) + $D \cdot L^2$	$L \cdot N$ (10^3)

First Example: N -th order scalar differential equation $N = 1000$, $M = 1$, $I = 1$

Second Example: Heat equation $N = 1000$, $M = 1000$, $I = 1000$

REFERENCES

1. Bellman, R., Introduction to Matrix Analysis, McGraw-Hill Book Company, Inc., New York, 1960.
2. -----, Adaptive Control Processes: A Guided Tour, Princeton University Press, Princeton, New Jersey, 1961.
3. -----, Introduction to the Mathematical Theory of Control Processes, Volume I, Academic Press Inc., New York, 1967.
4. Blum, E. K., "Calculus of Variations, Functional Analysis, and Optimal Control Problems," in Topics in Optimization (G. Leitmann, ed.), Academic Press Inc., New York, 1967.
5. Collins, D. C., Terminal State Dynamic Programming, University of Southern California, USCEE-338, March 1969.
6. Gelfand, I. M., and S. V. Tomis, Calculus of Variations, Prentice Hall, Englewood Cliffs, New Jersey, 1963.
7. Hildebrand, F. B., Methods of Applied Mathematics, 2nd edition, Prentice Hall, Englewood Cliffs, New Jersey, 1965.
8. Kramer, J. D. R. Jr., On Control of Linear Systems with Time Lags, The RAND Corporation, P-1948, 1960.
9. Schwartz, R. J., and B. Friedland, Linear Systems, McGraw-Hill Book Company, Inc., New York, 1968.
10. Bellman, R., and R. Kalaba, "Reduction of Dimensionality, Dynamic Programming, and Control Processes," J. Basic Engrg., March 1961, pp. 82-84.