Survey Papers

A Survey of Dynamic Programming Computational Procedures

ROBERT E. LARSON, MEMBER, IEEE

Abstract—**Although dynamic programming has long provided a powerful approach to optimization problems, its applicability has been somewhat limited because of the large computational requirements of the standard computational algorithm. In recent years a number of new procedures with greatly reduced computational requirements have been developed. The purpose of this paper is to survey a number of the more promising of those techniques. A review of the theory of dynamic programming and the standard computational algorithm is included. Several applications of the new techniques are discussed.**

I. Introduction

One of the most powerful techniques developed for the solution of optimization problems is Bellman's dynamic programming.[1]–[3] This technique solves, at least in principle, many important problems from fields such as electrical engineering, aerospace engineering, chemical engineering, economics, and operations research. However, because of the high computational requirements of the standard dynamic programming computational algorithm, only relatively simple problems have been solved on existing computers.

In recent years a number of new computational methods based on dynamic programming have been developed. Many of these methods retain the power and generality of dynamic programming and yet have substantially reduced computational requirements over those of the standard algorithm. This paper surveys a number of the more promising of these techniques and mentions some recent applications of them.

The fundamental concepts of dynamic programming are reviewed in Section II. The general deterministic variational control problem is formulated. The iterative functional equation based on Bellman's principle of optimality[1]–[3] is then derived. The standard dynamic programming computational algorithm is next described and discussed. Finally, the extension of these results to stochastic control and other problems involving uncertainty is indicated.

The new procedures are presented in Section III. They are grouped into four categories. In Section III-A procedures for obtaining a complete feedback control solution are discussed. In all cases, these procedures have greatly reduced high-speed memory requirements. However, because optimal control still must be computed for every admissible state, these procedures (except where mentioned explicitly) have computing time requirements which are about the same as for the standard algorithm. In Section III-B procedures for finding the optimal control sequence from a single initial state are examined. Because control does not need to be computed over much of the admissible state space, great savings in both memory and time are achieved. However, it is interesting to note that with these procedures a true feedback solution is computed in a region about the optimal trajectory. In Section III-C procedures for infinite-stage problems are given. These procedures solve a multistage problem with a very large number of stages by making single-stage calculation iteratively. Finally, in Section III-D procedures that do not fit any of these categories are discussed.

Some applications of the new techniques are discussed in Section IV. Because these techniques are not yet in widespread use, the author has chosen to present work being performed by himself and his associates. The examples come from many different fields, and they indicate the broad range of applicability of the ideas discussed in the paper. The examples include minimum-time-to-intercept trajectories for a ground-based interceptor missile, minimum-fuel trajectories for the SST, an airline scheduling problem, optimum operation of natural gas pipeline networks, optimum operations and planning for multipurpose water resource systems, optimal control of reliable and maintainable systems, and the optimum control of a robot in a partially unknown environment. Many other applications of dynamic programming, including some in which the new techniques are used, can be found elsewhere.[1]–[3],[7],[18],[26],[27]

Manuscript received May 19, 1967; revised July 26, 1967. This paper was presented at the IEEE International Convention, New York, N. Y., March 20–23, 1967.

The author is with the Information and Control Laboratory, Stanford Research Institute, Menlo Park, Calif.

II. Fundamentals of Dynamic Programming

A. Problem Formulation for the Deterministic Case

Most of the problems for which dynamic programming has been used to obtain numerical solutions can be formulated as deterministic discrete-time variational control problems.[1]–[3] The general case of this problem is formulated as follows.

Given

1) A system described by the nonlinear difference equation

$$\boldsymbol{x}(k+1) = \boldsymbol{\Phi}[\boldsymbol{x}(k), \boldsymbol{u}(k), k] \tag{1}$$

where $\boldsymbol{x}$ is an n-dimensional state vector, $\boldsymbol{u}$ is an m-dimensional control vector, k is an index for the stage variable, and $\boldsymbol{\Phi}$ is an n-dimensional vector functional.

2) A variational performance criterion

$$J = \sum_{k=0}^{K} L[\boldsymbol{x}(k), \boldsymbol{u}(k), k] \tag{2}$$

where J is the total cost and L the cost for a single stage.

3) Constraints

$$\boldsymbol{x} \in X(k) \tag{3}$$

$$\boldsymbol{u} \in U(\boldsymbol{x}, k) \tag{4}$$

where $X(k)$ is a set of admissible states at stage k, and $U(\boldsymbol{x}, k)$ a set of admissible controls at state $\boldsymbol{x}$, stage k.

4) An initial state

$$\boldsymbol{x}(0) = \boldsymbol{c}. \tag{5}$$

Find

The control sequence $\boldsymbol{u}(0), \cdots, \boldsymbol{u}(k)$ such that J in (2) is minimized subject to the system equation (1), the constraint equations (3) and (4), and the initial condition (5).

Continuous-time variational control problems can be treated by assuming that the control is piecewise constant in time and making appropriate transformations to the discrete-time case.[1]–[3] Extensions to problems involving uncertainty can be made as in Part D of this section.

B. Derivation of the Basic Iterative Functional Equation

The dynamic programming solution to the above problem is obtained by using an iterative functional equation that determines the optimal control for any admissible state at any stage. This equation follows immediately from Bellman's principle of optimality.[1]–[3] However, in the interest of clarity, a direct derivation will be given, and the principle of optimality will then be interpreted in terms of this equation.

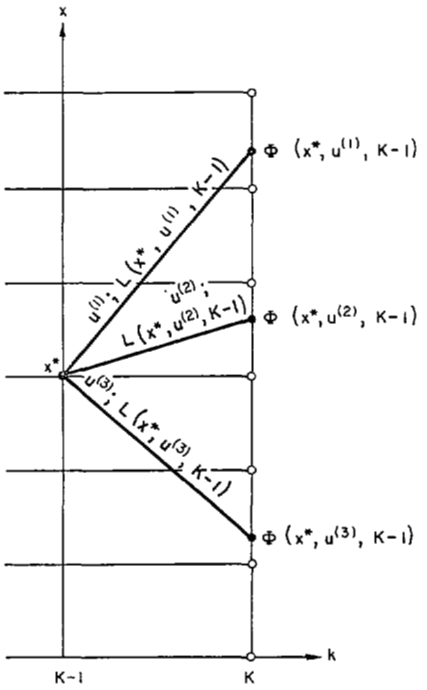

Fig. 1. Dynamic programming calculation in one-dimensional example at state x^*, stage $K-1$. ○—quantized states at stage K where $I(x, k)$ is known.

The first step in the derivation is to define the minimum cost function for all $\mathbf{x}\in X$ and all k, $k=0, 1, \cdots, K$, as

$$I(\mathbf{x}, k) = \min_{\substack{\mathbf{u}(j) \\ =k,k+1,\cdots,K}} \left\{ \sum_{j=k}^{K} L[\mathbf{x}(j), \mathbf{u}(j), j] \right\} \tag{6}$$

where

$$\mathbf{x}(k) = \mathbf{x}.$$

The summation is then split into two parts, the term evaluated for $j=k$ and the summation over $j=k+1$ to $j=K$. The minimization is similarly split into two parts. The result is

$$I(\mathbf{x}, k) = \min_{\mathbf{u}(k)} \min_{\substack{\mathbf{u}(j) \\ j=k+1,\cdots,K}} \left\{ L(\mathbf{x}, \mathbf{u}(k), k) + \sum_{j=k+1}^{K} L[\mathbf{x}(j), \mathbf{u}(j) j] \right\}. \tag{7}$$

The first term in brackets in (7) is not affected by the second minimization. Thus, (7) becomes

$$I(\mathbf{x}, k) = \min_{\mathbf{u}(k)} \left\{ L[\mathbf{x}, \mathbf{u}, (k), k] + \min_{\substack{\mathbf{u}(j) \\ j=k+1,\cdots,K}} \left[\sum_{j=k+1}^{K} L[\mathbf{x}(j), \mathbf{u}(j), j] \right] \right\}. \tag{8}$$

The second term in brackets in (8) is exactly analogous to the definition in (6), where the argument of I is $\Phi[\mathbf{x}, \mathbf{u}(k), k]$, $k+1$. Abbreviating $\mathbf{u}(k)$ as $\mathbf{u}$, the iterative functional equation becomes

$$I(\mathbf{x}, k) = \min_{\mathbf{u}} \{ L(\mathbf{x}, \mathbf{u}, k) + I[\Phi(\mathbf{x}, \mathbf{u}, k), k+1] \}. \tag{9}$$

This equation is a mathematical statement of Bellman's principle of optimality. It states that the minimum cost for state $\mathbf{x}$ at stage k is found by choosing the control that minimizes the sum of the cost to be paid at the present stage and the minimum cost in going to the end from the state at stage $k+1$ which results from applying this control. The optimal control at state $\mathbf{x}$ and stage k, denoted by $\hat{\mathbf{u}}(\mathbf{x}, k)$, is directly obtained as the value of $\mathbf{u}$ for which the minimum in (9) is attained.

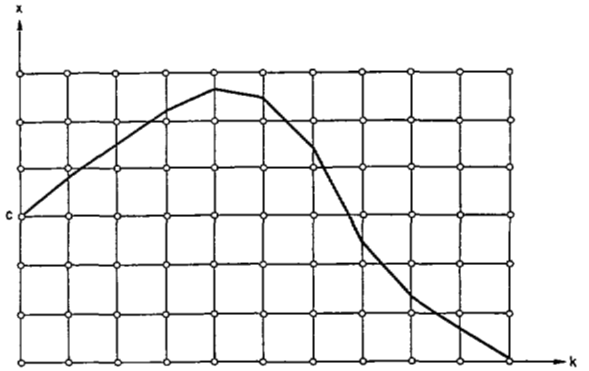

Fig. 2. Recovery of the optimal trajectory from initial state c. ○—quantized states at which $\hat{u}(x, k)$ and $I(x, L)$ are computed.

Since (9) determines $I(\mathbf{x}, k)$ and $\hat{\mathbf{u}}(\mathbf{x}, k)$ in terms of $I(\mathbf{x}, k+1)$, it must be solved backward in k. As a terminal boundary condition

$$I(x, K) = \min_{u} \{ L(x, u, K) \}. \tag{10}$$

The optimization over a sequence of controls is thus reduced to a sequence of optimizations over a single control vector.

C. The Standard Computational Algorithm

In the standard method for solving (9), each state variable x_i, $i=1, 2, \cdots, n$, is quantized to N_i levels, and each control variable $u_j, j=1, 2, \cdots, m$, is quantized to M_j levels.

Initially, $I(\mathbf{x}, K)$ is found for all quantized states $\mathbf{x}\in X$ by evaluating $L(\mathbf{x}, \mathbf{u}, K)$ for each quantized control $\mathbf{u}\in U$ and choosing the minimum value by a direct comparison. The optimal control $\hat{\mathbf{u}}(\mathbf{x}, K)$ is the value of $\mathbf{u}$ that minimizes $L(\mathbf{x}, \mathbf{u}, K)$. In many problems no control is applied at the final stage K; in this case $I(\mathbf{x}, K)$ is evaluated directly as $L(\mathbf{x}, K)$.

Next, at $k=K-1$, for each quantized state $\mathbf{x}\in X$, each quantized control $\mathbf{u}\in U$ is applied, and the next state $\Phi(\mathbf{x}, \mathbf{u}, K-1)$ is computed. The minimum cost at the next state, $I[\Phi(\mathbf{x}, \mathbf{u}, K-1), K]$, is found by interpolation using the values of $I(\mathbf{x}, K)$ at quantized states. The quantity $L(\mathbf{x}, \mathbf{u}, K-1)$ is computed directly. The sums of these quantities for each quantized control are then compared, and the minimum value is stored as $I(\mathbf{x}, K-1)$. The optimal control $\hat{\mathbf{u}}(\mathbf{x}, K-1)$, is stored as the value of $\mathbf{u}$ for which the minimum is attained. This procedure is illustrated for a one-dimensional problem in Fig. 1.

The procedure continues in this manner, with $I(\mathbf{x}, k)$ and $\hat{\mathbf{u}}(\mathbf{x}, k)$ being computed in terms of $I(\mathbf{x}, k+1)$, until $k=0$ is reached. The quantized states at which these quantities are computed in a one-dimensional example are shown in Fig. 2.

The optimal control sequence from the given initial state $\mathbf{c}$ is obtained from the values of $\hat{\mathbf{u}}(\mathbf{x}, k)$ at quantized states. The first control is read directly as $\hat{\mathbf{u}}(\mathbf{c}, 0)$. The next state is then computed as $\Phi[\mathbf{c}, \hat{\mathbf{u}}(\mathbf{c}, 0), 1]$, and the corresponding control is evaluated using interpolation on values of $\hat{\mathbf{u}}(\mathbf{x}, 1)$ at quantized $\mathbf{x}$. This procedure continues until the entire control sequence is obtained. This is shown in Fig. 2 for a one-dimensional example.

The same procedure can be used to compute the optimal control sequence for any initial state starting at any stage. If the initial state is not a quantized state, an additional interpolation may be necessary. Thus, the solution to many optimization problems is obtained in the same calculation as for the original problem. Bellman calls this invariant imbedding.[1]–[3]

This computational procedure is very appealing for a number of reasons. In the first place, thorny questions of existence and uniqueness are avoided; as long as there is at least one feasible control sequence, then the direct-search procedure guarantees that the absolute minimum cost is obtained. Furthermore, extremely general types of system equations, performance criteria, and constraints can be handled. Constraints actually reduce the computational burden by

decreasing the admissible sets X and U. Finally, the optimal control is obtained as a true feedback solution in which the optimal control for any admissible state and stage is determined.

In practice, however, the computational requirements of this technique become excessive when it is applied to large problems[1]-[3] (Bellman's curse of dimensionality). The most severe restriction arises because of the number of high-speed storage locations required to store $I(\mathbf{x}, k+1)$ during the computation of $I(\mathbf{x}, k)$ and $\hat{u}(\mathbf{x}, k)$. This number is

$$\text{HSMR} = \prod_{i=1}^{n} N_i. \tag{11}$$

Another consideration, which is generally not as restrictive as the high-speed memory requirement (HSMR), is the total amount of computing time (CT) required. This is

$$\text{CT} = \left(\prod_{i=1}^{n} N_i\right)\left(\prod_{j=1}^{m} M_j\right) K \Delta t_c \tag{12}$$

where Δt_c is the computing time required for a single evaluation of the quantity in brackets in (9) and a single scalar comparison.

The final factor that must be taken into account is the amount of off-line storage required to store the complete solution. This number of storage locations is

$$\text{LSMR} = \left(\prod_{i=1}^{n} N_i\right) Km. \tag{13}$$

D. Extension to Problems Containing Uncertainty

A number of computational methods based on dynamic programming have been developed for problems containing uncertainty. One case that has received much attention is Bellman's optimum stochastic control problem.[2] In this problem the state of the system is perfectly measurable, but a random forcing function $\mathbf{w}(k)$, which has a known probability density function, enters the system equation. The problem formulation is exactly as in Part A, except that

1) the system equation is affected by the random forcing function vector, so that (1) becomes

$$\mathbf{x}(k+1) = \mathbf{\Phi}[\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k), k]; \tag{14}$$

2) the expected value of the performance criterion is to be minimized, where the expectation is over the sequence of random forcing functions $\mathbf{w}(0), \mathbf{w}(1), \cdots, \mathbf{w}(K)$.

An iterative functional equation for this problem can be developed exactly as in Part B. The minimum cost function is defined as

$$I(\mathbf{x}, k) = \min_{\mathbf{u}(k),\cdots,\mathbf{u}(K)} \cdot \left\{ \underset{\mathbf{w}(k),\cdots,\mathbf{w}(K)}{E} \left[\sum_{j=k}^{K} L[\mathbf{x}(j), \mathbf{u}(j), \mathbf{w}(j), j] \right] \right\} \tag{15}$$

where

$$\mathbf{x}(k) = \mathbf{x}.$$

Assuming that samples of $\mathbf{w}(k)$ at different stages are uncorrelated,[1] the desired equation is obtained as

$$I(\mathbf{x}, k) = \min_{\mathbf{u}} \{ \underset{\mathbf{w}}{E}[L(\mathbf{x}, \mathbf{u}, \mathbf{w}, k) + I[\mathbf{\Phi}(\mathbf{x}, \mathbf{u}, \mathbf{w}, k), k+1]] \}. \tag{16}$$

As before, the optimal control $\hat{\mathbf{u}}(\mathbf{x}, k)$ is the control for which the minimum is attained. As a terminal boundary condition,

$$I(\mathbf{x}, K) = \min_{\mathbf{u}} \{ \underset{\mathbf{w}}{E}[L(\mathbf{x}, \mathbf{u}, \mathbf{w}, K)] \}. \tag{17}$$

A computational procedure analogous to that of Part C can be applied; the random forcing function $\mathbf{w}$ is quantized, the corresponding probability density function is converted to a discrete probability distribution, and the expectation is taken by summation.

A more general stochastic control problem, called the combined optimum control and estimation problem,[4] is represented in Fig. 3.

[1] This assumption can always be relaxed at the expense of defining additional state variables to account for the correlation.

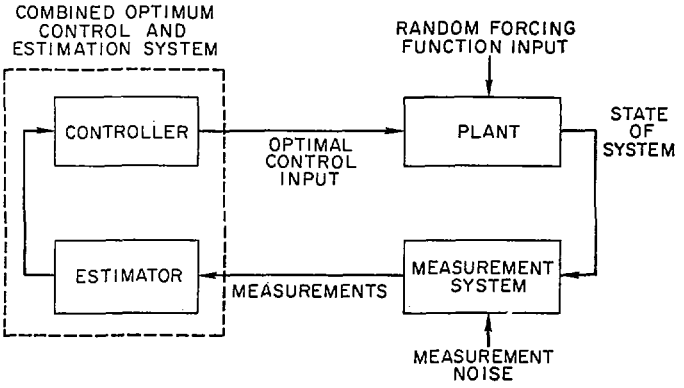

Fig. 3. Combined optimum control and estimation problem.

Here the state of the system is not known exactly, but instead is observed through a noisy measurement system. An iterative functional equation can be derived for this problem[4]-[7] in which the argument of the minimum cost function at a given stage is either the entire sequence of past controls and measurements or else a sufficient statistic for the a posteriori probability density function of the present state. This argument is generally called the *information state*. In either case it is necessary to calculate the latter probability density function by recursive application of Bayes' rule.[4]-[7] Other approaches to this problem are discussed in Kushner[38] and Wonham.[39]

Dynamic programming has also been applied to a number of other problem formulations involving uncertainty, notably optimum estimation and identification,[2],[7]-[10] optimum adaptive control,[2],[4],[7] and differential games.[11],[12]

III. New Computational Procedures

A. Procedures for Obtaining a Complete Feedback Control Solution

1) Closed-Form Solution. The most computationally efficient method for solving (9) is to find a closed-form expression for $I(\mathbf{x}, k)$. However, the number of cases where this can be done is small. The best known case is that in which the system equations are linear

$$\mathbf{x}(k+1) = F(k)\mathbf{x}(k) + D(k)\mathbf{u}(k); \tag{18}$$

the performance criterion is quadratic

$$J = \sum_{k=0}^{K} [\mathbf{x}^T(k) A(k) \mathbf{x}(k) + \mathbf{u}^T(k) B(k) \mathbf{u}(k)]; \tag{19}$$

and there are no constraints. In this case it can be shown that $I(\mathbf{x}, k)$ takes the form[13]

$$I(\mathbf{x}, k) = \mathbf{x}^T P(k) \mathbf{x} \tag{20}$$

where $P(k)$ satisfies the well-known Riccati equation

$$\begin{aligned} P(k) = {} & A(k) + F^T(k) P(k+1) F(k) \\ & - F^T(k) P(k+1) D(k) [B(k) + D^T(k) P(k+1) D(k)]^{-1} \\ & \times D^T(k) P(k+1) F(k). \end{aligned} \tag{21}$$

The corresponding optimal control $\hat{\mathbf{u}}(\mathbf{x}, k)$ takes the form[13]

$$\hat{\mathbf{u}}(\mathbf{x}, k) = -W(k)\mathbf{x} \tag{22}$$

where

$$W(k) = [B(k) + D^T(k) P(k+1) D(k)]^{-1} D^T(k) P(k+1) F(k). \tag{23}$$

The computations for an n-dimensional problem thus reduce to the iteration of an $n \times n$ symmetric matrix difference equation. A great saving in both storage and time is obtained.

Analogous results can be obtained for the optimum estimation problem in which the system equations and measurement equations are linear and all random variables are Gaussian,[7]-[10],[14] for the stochastic control problem in which the system equations are linear and the performance criterion is quadratic,[15] and for the combined optimum control and estimation problem in which the system equa-

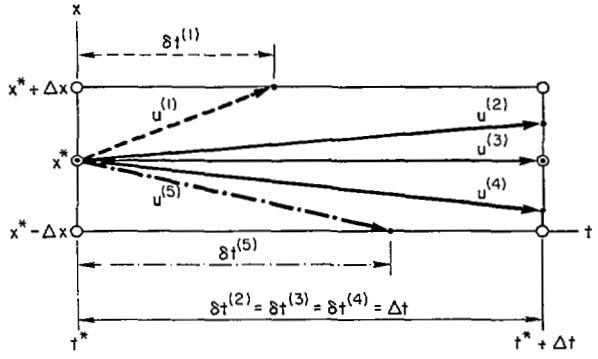

Fig. 4. Values of δt in state increment dynamic programming for controls $u^{(i)}$, $i = 1, 2, \cdots, 5$, applied at state x^*, time t^*.

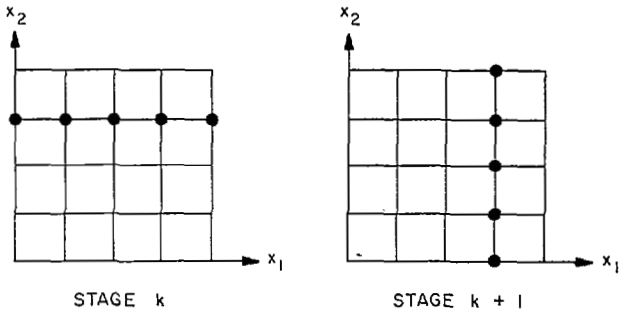

Fig. 5. States at stage $k+1$ for which $I(x, k+1)$ must be available in order to compute optimal control for the indicated states at stage k. ●—corresponding states.

tions and measurements equation are linear, all random variables are Gaussian, and the performance criterion is quadratic.[15]-[17]

2) Polynomial Approximation of the Minimum Cost Function. In certain well-behaved problems it is possible to obtain an accurate approximation to $I(\mathbf{x}, k+1)$ over all $\mathbf{x} \in X$ by expressing I as a low-order polynomial in $\mathbf{x}$.[1],[3] In such cases a saving in high-speed memory requirement can be made by storing only the coefficients of the polynomial, rather than values of I for all quantized $\mathbf{x}$.

3) Search Procedures other than Quantizing Control. If the interpolation formulas for $I(\mathbf{x}, k+1)$ are of the proper form, it is sometimes possible to use efficient search procedures in the minimization in (9), rather than quantization of control and direct comparison. Search procedures that have been used in the past include Fibonacci search,[1]-[3],[7] the simplex method of linear programming,[3],[7] and certain steepest-descent algorithms.[3],[7],[16]

4) Use of Analytical Results and Necessary Conditions. Analysis of a particular problem may sometimes reveal useful information about the solution. For example, in the bang-bang control problem it can be shown that the optimal control is always one of the two extreme admissible values.[19] Conditions such as this can reduce computational requirements by decreasing the size of the admissible control set $U(\mathbf{x}, k)$.

5) State Increment Dynamic Programming. In discrete-time optimal control problems that have been transformed from continuous-time problems, a substantial saving in high-speed memory requirements can be obtained by using state increment dynamic programming.[7],[20]-[22] If the system equations for the continuous-time problem are

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{24}$$

where t is a continuous stage variable (usually time), $\mathbf{f}$ an n-dimensional vector functional, and the dot denotes d/dt, and if the performance criterion is

$$J = \int_{t_0}^{t_f} l(\mathbf{x}, \mathbf{u}, t)dt + \psi[\mathbf{x}(t_f), t_f] \tag{25}$$

where J is the total cost, t_0 the initial time, t_f the final time, and l and ψ are scalar functionals representing cost per unit time and terminal cost, respectively, and if the approximations are made that

$$\mathbf{x}(t + \delta t) \cong \mathbf{x}(t) + \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]\delta t \tag{26}$$

and that

$$\int_t^{t+\delta t} l[\mathbf{x}(\sigma), \mathbf{u}(\sigma), \sigma]d\sigma \cong l[\mathbf{x}(t), \mathbf{u}(t), t]\delta t, \tag{27}$$

then the iterative functional equation becomes

$$I(x, t) = \min_{\mathbf{u}} \{l(x, u, t)\delta t + I[x + f(x, u, t)\delta t, t + \delta t]\}. \tag{28}$$

In the state increment dynamic programming computational procedure, the state $\mathbf{x}$ and control $\mathbf{u}$ are quantized as before. The stage variable t is also quantized with an increment size Δt, and minimum cost and optimal control are computed only at quantized values of x and t. However, δt, the time increment over which a given piecewise constant control is applied, is determined as

$$\delta t = \min_{i=1,2,\cdots,n} \left\{ \left| \frac{\Delta x_i}{f_i(\mathbf{x}, \mathbf{u}, t)} \right|, \Delta t \right\} \tag{29}$$

where Δx_i is the quantization increment in the ith state variable and $f_i(\mathbf{x}, \mathbf{u}, t)$ the ith component of $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$. This equation ensures that the change in any state variable x_i over the increment δt is at most Δx_i and that δt is at most Δt. Representative values of δt for a one-dimensional example are shown in Fig. 4.

The major consequence of this value of δt is that $I[x + f(x, u, t)\delta t, t + \delta t]$ can be determined using only values of the minimum cost function at neighboring quantized states for two or three time increments. Interpolation is in $n-1$ state variables and t if $\delta t < \Delta t$ and in n state variables if $\delta t = \Delta t$.

This result can be exploited by processing data in units called blocks,[7],[20]-[22] which cover few increments along each state variable axis but several increments in t. A great saving in high-speed memory requirement can thus be achieved at a small cost in computing time. In one example the saving in high-speed memory requirement was from 10^6 storage locations to 100 locations.[7],[20] A general program for implementing this technique in programs having four or less state variables has recently been written.[7]

6) Reduction of the Dimensionality of High-Speed Memory Requirement from n to m. If $m < n$, i.e., if there are fewer control variables than state variables, then a substantial saving in high-speed memory requirement can be achieved by transforming the state space so that the control vector changes only m of the state variables.[7],[23] As a result, $n-m$ of the state variables in the next state can be fixed, and values of minimum cost at the next state need be stored only as a function of m of the state variables.

The conditions under which the transformation can be made[23] are closely related to Kalman's controllability.[24] For the scalar control case ($m = 1$), the transformation consists of rewriting (1) in the form

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= x_3(k) \\ &\vdots \\ x_{n-1}(k+1) &= x_n(k) \\ x_n(k+1) &= \Phi_n[x_1(k), \cdots, x_n(k), u(k)]. \end{aligned} \tag{30}$$

For fixed values of $x_2(k), x_3(k), \cdots, x_n(k)$, the state variables $x_1(k+1), k_2(k+1), \cdots, x_{n-1}(k+1)$ are also fixed. Thus at these fixed values, optimal control at all values of $x_1(k)$ can be computed by storing the minimum costs as a function of $x_n(k+1)$ only. The point at which optimal control is computed at stage k and the corresponding points at stage $k+1$ for which the minimum cost must be stored in high-speed memory are shown for an example where $m = 1$, and $n = 2$ in Fig. 5. The generalization for $m > 1$ is discussed in Wong.[23]

7) *Forward Dynamic Programming.* If the minimum cost function is redefined to be the minimum cost to reach a given state and stage from the initial state, an iterative functional equation analogous to (9) can be derived. In this case the calculations proceed forward in k rather than backward.

For many applications this solution is more desirable than the backward dynamic programming solution. This is particularly true when the initial state is fixed and the terminal state and/or stage is free. The optimum final state can be selected by searching over all admissible final states and, if desired, adding a terminal cost function. The terminal cost function can be quite flexible, and the effect of using different functions can easily be seen without repeating the dynamic programming calculation. One class of problems where this procedure has considerable appeal is real-time dispatching in which the initial state is fixed and the human operator is allowed some freedom in determining the final state.

A computational procedure analogous to that of Section II-B can be used. However, this necessitates inversion of the system equations, i.e., finding $\boldsymbol{x}(k)$ such that

$$\boldsymbol{\Phi}[\boldsymbol{x}(k), \boldsymbol{u}(k), k] = \boldsymbol{x}(k+1)$$

for a given $\boldsymbol{x}(k+1)$ and $\boldsymbol{u}(k)$.[7] An alternative procedure is to use the system equations forward in time, establish a tentative minimum cost for each new next state computed, compare this cost whenever another trajectory arrives at the same next state, and store the tentative minimum costs as actual minimum costs when all controls have been applied for all quantized present states.[7]

B. Procedures for Obtaining the Optimal Trajectory from a Single Initial State

1) *Direct Iteration.* In order to begin this approach, all available information about the system is utilized to obtain a nominal trajectory. The set of admissible states $X(k)$ is then adjusted to cover a region about this trajectory, and a normal dynamic programming computation is made. This calculation is usually done at a great saving in both storage and time requirements over the case where $X(k)$ covers the entire admissible state space.

If it is found that the optimal trajectory from the initial state is unsatisfactory, e.g., if it leaves the set $X(k)$ at some stage, then the region is readjusted and a new computation is made. Iterations are performed until a satisfactory trajectory is obtained.

The successful application of a direct iterative procedure depends to a large extent on how close to the true optimal trajectory is the nominal trajectory. In many applications a good nominal trajectory can readily be found, while in other cases this is a very difficult task.

A direct iterative procedure of particular interest when a good nominal cannot be found is to choose a large initial region for $X(k)$ and to use a coarse quantization in both state and control variables. Then in successive iterations the size of the region is decreased and the quantizations are made finer.[3],[7],[18]

In a direct approach based on state increment dynamic programming, the region is specified by choosing only certain blocks to be processed. The variation of δt according to (29) is useful for insuring that the constraints on $X(k)$ are not violated when the region becomes small.[7],[9]

Still other direct iterative techniques are currently being studied.[26] In all these cases a useful by-product of the calculations is the optimum feedback control in a region about the optimal trajectory.

2) *Quasilinearization.* Another computational procedure based on the existence of a nominal trajectory is quasilinearization.[1]–[3],[25] The basic idea behind this technique is to make a linear expansion of system equations and a quadratic expansion of the performance criterion about this nominal trajectory and then to use the closed-form solution discussed in Part A-1. A number of variations of this method have been studied, and computer programs for implementing many of them are available.[25]

3) *Successive Approximations.* In this approach a nominal trajectory is again assumed. For the case $m=n$, i.e., where there are as many control variables as state variables, the sequence of state variables is held fixed for all but one. This state variable is then used in a one-dimensional dynamic programming problem, where the control vector is n dimensional, except that holding the sequences of the other $n-1$ state variables fixed imposes $n-1$ equality constraints on the control variables. The performance criterion and constraints remain the same. After this problem has been solved, the optimal sequence of states for this state variable is found, and the procedure is repeated with a different state variable. Iterations continue until convergence is obtained. In this manner the solution of an n-dimensional dynamic programming problem is reduced to solving a sequence of one-dimensional problems.[3],[7],[18]

The computational savings in both time and storage are very impressive. There are several variations on this basic technique, and extensions can be made to problems in which $m \neq n$. In all cases the result is the same: the solution of a high-dimensional problem is reduced to solving a sequence of lower-dimensional ones.[3],[7]

It is easy to show that convergence is monotonic, but the conditions under which an absolute opinium is obtained are not precisely known. For a number of problems involving four or less state variables, the method was found to converge to the true optimum.[7] In still other examples, most with a larger number of state variables, the solution appears reasonable; however, the true optimum cannot be computed as a check.[7]

Convergence to the true optimal trajectory is more likely to occur if the nominal trajectory is close to the true optimal trajectory. A method for choosing a good nominal trajectory in problems where successive approximations can be applied is discussed in Larson.[7] Other methods for improving the likelihood of convergence are discussed in Bellman and Dreyfus.[3]

4) *Use of a Lagrange Multiplier.* In problems with certain types of constraints, it is possible to eliminate state variables by defining Lagrange multipliers.[3],[18] This reduction in dimensionality makes possible substantial savings in both storage and time.

C. Procedures for Obtaining a Steady-State Solution

1) *Approximation in Function Space.* Under certain conditions, e.g., if Φ and L do not explicitly depend on k and if $K \to \infty$, then $I(x, k)$ does not depend on k. The functional equation then becomes

$$I(\boldsymbol{x}) = \min_{\boldsymbol{u}} \{L(\boldsymbol{x}, \boldsymbol{u}) + I[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u})]\}. \tag{31}$$

The problem is thus effectively single stage; however, the solution is complicated by the fact that the minimum cost function appears both inside and outside the minimization.

One method of obtaining a solution to this problem is to make an appropriate guess of $I(\boldsymbol{x})$, say $I^{(0)}(\boldsymbol{x})$, and solve for a sequence of minimum cost functions according to the relation

$$I^{(j+1)}(\boldsymbol{x}) = \min_{\boldsymbol{u}} \{L(\boldsymbol{x}, \boldsymbol{u}) + I^{(j)}[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u})]\}. \tag{32}$$

If $I^{(0)}(\boldsymbol{x})$ is a close approximation to the true $I(\boldsymbol{x})$, this procedure may be expected to converge to the proper function. However, the conditions under which convergence can be guaranteed are quite restrictive.

2) *Approximation in Policy Space.* An alternative method, which has better convergence properties, is to guess an optimal policy $\hat{\boldsymbol{u}}^{(0)}(\boldsymbol{x})$. The corresponding minimum cost function $I^{(0)}(\boldsymbol{x})$ is then computed by a direct iteration according to the relation

$$I^{(0,i+1)}(\boldsymbol{x}) = L[\boldsymbol{x}, \hat{\boldsymbol{u}}^{(0)}(\boldsymbol{x})] + I^{(0,i)}\{\boldsymbol{\Phi}[\boldsymbol{x}, \hat{\boldsymbol{u}}^{(0)}(\boldsymbol{x})]\}. \tag{33}$$

The initial guess $I^{(0,0)}(\boldsymbol{x})$ is usually

$$I^{(0,0)}(\boldsymbol{x}) = 0. \tag{34}$$

When $I^{(0)}(x)$ has been found from iteration of (33), a new policy $\hat{u}^{(1)}(x)$ is found by solving

$$I^*(x) = \min_u \{L(x, u) + I^{(0)}[\Phi(x, u)]\}. \quad (35)$$

The policy $\hat{u}^{(1)}(x)$ for a given value of x is determined as the value of u for which the minimum is attained in (35). However, $I^*(x)$ is *not* $I^{(1)}(x)$, the minimum cost function corresponding to policy $\hat{u}^{(1)}(x)$, because $I^{(0)}[\Phi(x, u)]$ appears inside the brackets in (35). Another direct iteration as in (33) is thus necessary.

In general, the minimum cost function $I^{(j)}(x)$ corresponding to the policy $\hat{u}^{(j)}(x)$ is found by iterations of

$$I^{(j,i+1)}(x) = L[x, \hat{u}^{(j)}(x)] + I^{(j,i)}[\Phi(x, \hat{u}^{(j)}(x)] \quad (36)$$

with initial guess

$$I^{(j,0)}(x) = 0. \quad (37)$$

A new policy $\hat{u}^{(j+1)}(x)$ is then formed from

$$I^*(x) = \min_u \{L(x, u) + I^{(j)}[I(x, u)]\}. \quad (38)$$

Convergence to the true optimum can be proved for many important cases.

Howard[27] has generalized this procedure to the stochastic control case. He shows that in the limit of $K \to \infty$, the minimum cost function takes the form

$$I(x) = V(x) + gK \quad (39)$$

where $V(x)$ is a transient cost and g is a steady-state "gain." By following a procedure somewhat analogous to (36), which he calls the value determination operation (VDO), he finds the $V(x)$ and g corresponding to a particular policy. To obtain a new policy, he minimizes g; this process, which he calls the policy improvement routine (PIR), is similar to employing (38). Again, convergence can be proved for a quite general set of cases.

3) State Increment Dynamic Programming. In some problems the minimum cost function and optimal control can be found in one step by using state increment dynamic programming. Both steady-state-problems of the type described in Part C-1 and certain minimum time-to-origin problems have been solved in this manner.[7]·[20] Applications have been made to both deterministic and stochastic cases.[7]

D. Other Procedures

1) Adaptive Data Processing. By varying increment sizes, nominal trajectories, and other program inputs, it is possible to cut down substantially the number of computations for a given problem. These variations can either be preprogrammed or made in real time by the programmer/operator.[3]·[7]

2) New Concepts in Computers. Improvements in computer hardware and software are constantly increasing the scope of dynamic programming. Some recent ideas for computer reorganization, especially in the areas of parallel processing and cellular logic,[28] could lead to very large reductions in the computer time and storage required for solving large problems.

3) Clever Problem Formulation. In the final analysis, careful choice of state, control, and stage variables, selection of an appropriate performance criterion, and use of judicious approximations are the most important factors in the successful application of dynamic programming. Unless the mathematical problem formulation is both a concise and a meaningful representation of the actual situation, the application of sophisticated computational procedures will yield little benefit.

IV. Applications

A. Minimum-Time-to-Intercept Trajectories for a Ground-Based Interceptor Missile

In attempting to intercept an enemy missile or aircraft with an interceptor missile, the initial launch strategy is extremely important. Although later guidance commands can compensate for some errors, the outcome of an engagement is largely determined by the initially chosen trajectory.

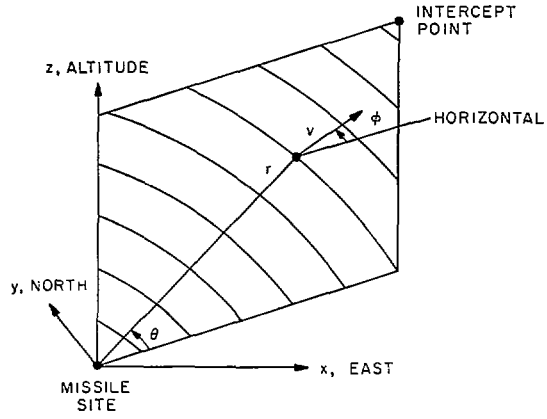

Fig. 6. Coordinates r, θ, v, and ϕ in minimum-time-to-intercept problem.

In order to aid in the selection of an initial trajectory, a dynamic programming program has recently been written for finding the minimum-time trajectory from the missile site to any point in three-dimensional space.[29] The program uses forward dynamic programming. Although the problem appeared at first to have six state variables, namely, missile position and velocity along each coordinate axis, the number was quickly reduced to three. First, it was observed that minimum-time trajectories always stayed in the plane determined by the intercept point and the local vertical at the missile site; this eliminated two state variables. Next, the radial distance r from the missile site was selected as the stage variable; this reduced the number of state variables to three. These variables were chosen to be angle of missile position vector with respect to horizontal θ, magnitude of missile velocity v, and angle of missile velocity vector with respect to horizontal ϕ. The final choice of coordinates is shown in Fig. 6.

The program can be modified to handle any missile dynamics and any constraints. Alternative performance criteria, such as minimum loss of velocity, can also be considered. A terminal performance measure, such as the angle between missile velocity vector and target velocity vector can be imposed.

The program was run with data for a hypothetical missile. The program required 40 minutes on the IBM 7090 to generate optimal trajectories to any point that the missile is capable of reaching. These results have been stored on tape, and a program has been written for recovering the trajectory to any specified point or points; this latter program takes less than 40 seconds to recover the optimum trajectory to any 100 selected points. The accuracy of the program was tested by using a quasilinearization technique to generate the minimum-time trajectory to some representative points; the dynamic programming solution was used as the initial nominal trajectory. In all cases the improvement in minimum time was by less than two percent.

B. Minimum-Fuel Trajectories for the SST

A computer program based on state increment dynamic programming has been written for computing minimum-fuel trajectories for the SST.[7]·[20]–[22]·[30] The program is capable of handling a wide variety of aerodynamic equations and constraints. The program can be run either in an evaluation mode in which it generates optimum trajectories from all feasible initial states, or in a real-time control mode in which it generates optimum trajectories only in a region about a preselected nominal trajectory.

C. Airline Scheduling

In the airline scheduling problem it is desired to assign a fleet of aircraft to a set of scheduled flights in an optimum manner. The

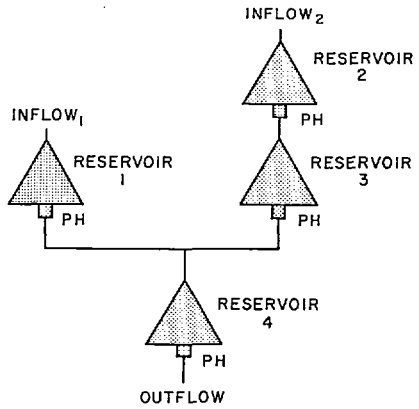

Fig. 7. Configuration of four-dam problem.

multistage decision aspects of this problem enter because not only is an aircraft committed for a specified time interval, but also its position in space at the end of that time is specified. The problem becomes especially difficult when one or more aircraft become unavailable because of unscheduled maintenance or when one or more airports become inaccessible because of weather. In addition, there are a number of operating constraints, such as the maximum number of hours an aircraft can fly over the day.

A dynamic programming program for optimally scheduling aircraft was discussed in Larson.[31] Although this program was primarily designed for operations over a 24-hour period, it can easily be adapted both to short-term scheduling when the originally planned schedule is interrupted by weather or aircraft failure and to longer-term scheduling. It is very flexible in the constraints it can consider and in the factors used in the performance criterion. However, because of computational requirements its applicability is limited to about six aircraft.

A new program, based on successive approximations, has recently been written.[7] This program is capable of handling a few hundred aircraft on presently available computers. On a medium-speed computer[2] the solution to a problem with 70 aircraft, 10 airports, and over 300 flights was obtained in 3 minutes.

D. Optimum Operations and Planning of Water Resource Systems

A number of problems related to the optimum operations and planning of multipurpose water reservoir systems have been solved by dynamic programming. Several of these are discussed in Larson and Keckler[32] One example is the optimum operation over a 24-hour period of a four-dam system in a series-parallel configuration (Fig. 7) in which the performance criterion takes into account use of the system for power generation, irrigation, flood control, and recreation. This problem was solved by successive approximations; convergence to the true optimum was obtained in 30 seconds on a medium-speed computer.[2] Another example is the optimum operation of a single reservoir over an entire year in which the stochastic nature of stream flows must be taken into account; a solution for this case was obtained by iteration in policy space. Still another example is the long-term planning of system additions; this problem was solved by forward dynamic programming.

E. Optimum Operation of a Natural Gas Pipeline Network

Dynamic programming has been used to find the short-term (on the order of 24 hours) operating policy for compressor stations, such that all demanded gas is delivered at contract pressure or greater; the pressure limits of the pipes are not exceeded, and compressor operating costs are minimized.[33] Dynamic equations of the pipeline were obtained by making lumped approximations to the partial differential equations describing the flow of gas.

[2]Burroughs B-5500.

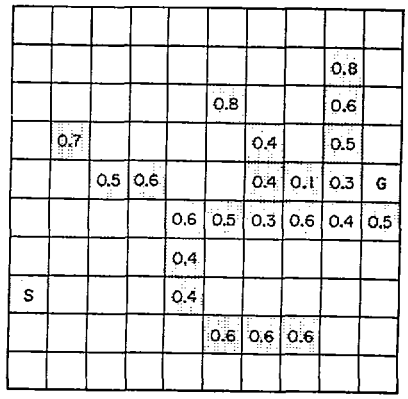

Fig. 8. Optimal control of a robot in a partially unknown environment. Probability of barrier being on the square equals the number in the square if one is given, and zero otherwise.

In this application a dynamic programming program was not implemented in real time, but instead its results were used to suggest and evaluate a suboptimal algorithm that is currently being implemented in an actual network.

F. Optimum Design of Reliable and Maintainable Systems

Dynamic programming has recently been used to solve the problem of optimally designing a system when some of the subsystems have a nonzero probability of failing and when there is limited repair capability.[34] The basic approach is to augment the state vector with variables that indicate whether or not the subsystems have failed and to what extent the repair capability has already been used; in terms of the combined optimal control and estimation theory,[4]–[7] this is the "information state" for the system. The decision of when to repair is made part of the control vector. The resulting problem formulation can be handled by the dynamic programming procedure for the stochastic control case. Some examples are worked out in Ratner.[34]

G. Optimal Control of a Goal-Seeking Robot in a Partially Unknown Environment

A problem that contains elements of dual control theory,[35] "clairvoyance,"[36] and learning systems theory is the following. A robot is attempting to find its way to a particular goal. The location of the goal is not known exactly, but an a priori probability distribution is available. There are barriers that the robot is not allowed to cross, but their location is also specified only by an a priori probability distribution. The robot is allowed to "see" only in a small area about its present location; however, it is allowed to remember where it has been before. Also, it is allowed to see a large area by paying a certain penalty. The problem is to find the sequence of "moves" and "looks" that will enable it to minimize, in an expected value sense, the sum of the penalty for looks and the distances covered by moves.

A dynamic programming formulation of this problem has recently been developed.[37] As in the previous section, it is necessary to augment the basic system state and control vectors. A number of sample problems have been solved. In Fig. 8 one such example is depicted in which the goal location is known, the possible barrier locations are known, and the a priori probabilities that the barriers are present are given. A move for the robot consists of moving $\pm$ one unit in either the x or y direction. The robot is allowed to "see" one move ahead at all times, and for a cost of 0.2 moves, it can "see" two moves ahead. The performance criterion is the total expected number of moves to reach the goal, including the costs for looks. As a matter of interest, it was found that a group of engineers, when given the same set of rules for the problem, did not achieve as good a performance as the computer-generated solution.

V. Conclusions

In this paper a number of new computational procedures based on dynamic programming have been presented. These methods have computational requirements that are far less than those of the standard computational algorithm. Many nonlinear problems involving four state variables have been solved by these methods, and several applications involving a larger number of state variables have been made.

The reduced computational requirements of these procedures should lead to increased use by control engineers of dynamic programming as a computational tool. Athans' comprehensive survey paper on optimal control theory and applications[40] showed little activity in applications of dynamic programming, but considerable utilization of computational algorithms based on the calculus of variations and the maximum principle. However, as control engineers attack a broader range of problems, it is felt that there will be more demand for computational algorithms that have the unique advantages of dynamic programming. In particular, these methods will find application in problems where one or more of the following conditions is present: a feedback control solution is particularly desirable; it is difficult to guarantee a true optimum solution by using indirect methods; the system variables are naturally discrete; stochastic effects must be considered; the system equations are nonlinear; the performance criterion is a complicated function; or constraints of a general nature are present. It is hoped that this paper, by pointing out efficient computational procedures that retain these advantages, will simulate further work by control engineers, both in applications of the present algorithms and in the development of even more improved methods.

References

[1] R. Bellman, *Dynamic Programming*. Princeton, N. J.: Princeton University Press, 1957.

[2] R. Bellman, *Adaptive Control Processes*. Princeton, N. J.: Princeton University Press, 1961.

[3] R. Bellman and S. Dreyfus, *Applied Dynamic Programming*. Princeton, N. J.: Princeton University Press, 1962.

[4] L. Meier, III, "Combined optimum control and estimation," *Proc. 3rd Ann. Allerton Conf. on Circuit and System Theory*, (Urbana, Ill.; October 1965), pp. 109–120.

[5] R. Sussman, "Optimal control of systems with stochastic disturbances," Electronics Research Lab., University of California, Berkeley, ser. 63, no. 20, November 1963.

[6] M. Aoki, "Optimal Bayesian and min–max controls of a class of stochastic and adaptive dynamic systems," Preprints, IFAC Tokyo Symp. on Systems Engrg. for Control System Design (1965), pp. II-21–II-31.

[7] R. E. Larson, *State Increment Dynamic Programming*. New York: American Elsevier, 1967.

[8] R. E. Larson and J. Peschon, "A dynamic programming approach to trajectory estimation," *IEEE Trans. Automatic Control (Short Papers)*, vol. AC-11, pp. 537–540, July 1966.

[9] R. E. Bellman *et al.*, "Invariant imbedding and nonlinear filtering theory," RAND Corp., Santa Monica, Calif., Research Memo. RM-4374, December 1964.

[10] H. Cox, "On the estimation of state variables and parameters for noisy dynamic systems," *IEEE Trans. Automatic Control*, vol. AC-9, pp. 5–12, January 1964.

[11] R. Isaacs, *Differential Games*. New York: Wiley, 1965.

[12] P. A. Meschler, "On a goal-keeping differential game," *IEEE Trans. Automatic Control*, vol. AC-12, pp. 15–21, February 1967.

[13] R. E. Kalman and R. W. Koepke, "Optimal synthesis of linear sampling control systems using generalized performance indexes," *Trans. ASME*, vol. 80, pp. 1820–1838, November 1958.

[14] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Trans. ASME, J. Basic Engrg.*, ser. D, vol. 83, pp. 95–108, March 1961.

[15] T. L. Gunkel, II, "Optimum design of sampled-data systems with random parameters," Stanford Electronics Labs., Stanford, Calif., Tech. Rept. 2102-2, Contracts Nonr-225(24) and NR 373-360, April 24, 1961.

[16] T. L. Gunkel, II, and G. F. Franklin, "A general solution for linear sampled-data control," *Trans. ASME, J. Basic Engrg.*, ser. D, vol. 85, pp. 197–201, March 1963.

[17] P. D. Joseph and J. T. Tou, "On Linear control theory," *Trans. AIEE (Applications and Industry)*, vol. 80, pp. 193–196, September 1961.

[18] G. L. Nemhauser, *Introduction to Dynamic Programming*. New York: Wiley, 1966.

[19] R. Bellman, I. Glicksberg, and O. Gross, "On the 'bang-bang' control problem," *Q. Appl. Math.*, vol. 14, no. 1, pp. 11–18, 1956.

[20] R. E. Larson, "Dynamic programming with continuous independent variable," Stanford Electronics Labs., Stanford, Calif., Tech. Rept. 6302-6, Contracts Nonr-225(24) and NR 373 366, SEL-64-019, April 1964.

[21] R. E. Larson, "Dynamic programming with reduced computational requirements," *IEEE Trans. Automatic Control*, vol. AC-10, pp. 135–143, April 1965.

[22] R. E. Larson, "An approach to reducing the high-speed memory requirement of dynamic programming," *J. Math. Anal. Appl.*, vol. 11, pp. 519–537, July 1965.

[23] P. J. Wong, "Dynamic programming using shift vectors," Ph.D. dissertation, Dept. of Elec. Engrg., Stanford University, Stanford, Calif. (in preparation).

[24] R. E. Kalman, "On the general theory of control systems," *Proc. 1st Internat'l Fed. Automatic Control Cong.* (Moscow, 1960). London: Butterworths, 1960, pp. 481–492.

[25] R. Bellman and R. E. Kalaba, *Quasilinearization and Nonlinear Boundary-Value Problems*. New York: American Elsevier, 1965.

[26] W. G. Keckler, "Optimization about a nominal trajectory via dynamic programming," engineer's thesis, Dept. of Elec. Engrg., Stanford University, Stanford, Calif. (in preparation).

[27] R. A. Howard, *Dynamic Programming and Markov Processes*. New York: Wiley, 1960.

[28] R. Minnick, Montana State University, Bozeman, private communication.

[29] R. E. Larson, "Optimum guidance laws for an AMM," Stanford Research Institute, Menlo Park, Calif., Preliminary Rept., Contract DA—01-021—AMC-90006(Y), SRI Project 5188, January 1966.

[30] D. W. Richardson, E. P. Hechtman, and R. E. Larson, "Control data investigation for optimization of fuel in supersonic transport vehicles," Hughes Aircraft Co., Culver City, Calif., Phase II Rept., Contract AF 33(657)-8822, Project 9056, June 1963.

[31] R. E. Larson, "A dynamic programming approach to airline scheduling," presented at the 5th AGIFORS Conf. (Chicago, Ill., October 1965).

[32] R. E. Larson and W. G. Keckler, "Applications of dynamic programming to the control of water resource systems," presented at IFAC Haifa Symp. on Computer Control of Natural Resources and Public Utilities, Haifa, Israel, September 1967.

[33] R. E. Larson, T. L. Humphrey, and P. J. Wong, "Short-term optimization of a single pipeline," Stanford Research Institute, Menlo Park, Calif., Interim Rept., SRI Project 5973, February 1967.

[34] R. S. Ratner, "Optimum design of reliable and maintainable systems" (to be published).

[35] A. A. Feldbaum, "Dual control theory—I," *Avtom. i Telemekh.*, vol. 21, pp. 1240–1249, September 1960.

[36] R. A. Howard, "Information value theory," *IEEE Trans. Systems Science Cybernetics*, vol. SSC-2, pp. 22–26, August 1966.

[37] R. E. Larson and W. G. Keckler, "Optimum control of a robot in a partially unknown environment," *IEEE Trans. Automatic Control* (to be published).

[38] H. J. Kushner, "On the status of optimal control and stability for stochastic systems," *1966 IEEE International Conv. Rec.*, vol. 14, pt. 6, pp. 143–151.

[39] W. M. Wonham, "Stochastic problems in optimal control," *1963 IEEE Internat'l Conv. Rec.*, vol. 11, pt. 2, pp. 114–124.

[40] M. Athans, "The status of optimal control theory and applications for deterministic systems," *IEEE Trans. Automatic Control (Survey Papers)*, vol. AC-11, pp. 580–596, July 1966.