WATER RESOURCES RESEARCH, VOL. 24, NO. 8, PAGES 1345–1359, AUGUST 1988

Gradient Dynamic Programming for Stochastic Optimal Control of Multidimensional Water Resources Systems

EFI FOUFOULA-GEORGIOU

Department of Civil Engineering, Iowa State University, Ames

PETER K. KITANIDIS

Department of Civil Engineering, Stanford University, Stanford, California

A new computational algorithm is presented for the solution of discrete time linearly constrained stochastic optimal control problems decomposable in stages. The algorithm, designated gradient dynamic programming, is a backward moving stagewise optimization. The main innovations over conventional discrete dynamic programming (DDP) are in the functional representation of the cost-to-go function and the solution of the single-stage problem. The cost-to-go function (assumed to be of requisite smoothness) is approximated within each element defined by the discretization scheme by the lowest-order polynomial which preserve its values and the values of its gradient with respect to the state variables at all nodes of the discretization grid. The improved accuracy of this Hermitian interpolation scheme reduces the effect of discretization error and allows the use of coarser grids which reduces the dimensionality of the problem. At each stage, the optimal control is determined on each node of the discretized state space using a constrained Newton-type optimization procedure which has quadratic rate of convergence. The set of constraints which act as equalities is determined from an active set strategy which converges under lenient convexity requirements. This method of solving the single-stage optimization is much more efficient than the conventional way based on enumeration or iterative methods with linear rate of convergence. Once the optimal control is determined, the cost-to-go function and its gradient with respect to the state variables is calculated to be used at the next stage. The proposed technique permits the efficient optimization of stochastic systems whose high dimensionality does not permit solution under the conventional DDP framework and for which successive approximation methods are not directly applicable due to stochasticity. Results for a four-reservoir example are presented.

1. INTRODUCTION

The purpose of this paper is to present a new computational algorithm for the stochastic optimization of sequential decision problems. One important and extensively studied class of such problems in the area of water resources is the discrete time optimal control of multireservoir systems under stochastic inflows. Other applications include the optimal design and operation of sewer systems [e.g., *Mays and Wenzel*, 1976; *Labadie et al.*, 1980], the optimal conjunctive utilization of surface and groundwater resources [e.g., *Buras*, 1972], and the minimum cost water quality maintenance in rivers [e.g., *Dracup and Fogarty*, 1974; *Chang and Yeh*, 1973], to mention only a few of the water resources applications and pertinent references. An extensive review of dynamic programming applications in water resources can be found in the works by *Yakowitz* [1982] and *Yeh* [1985]. Before we proceed with the description of our algorithm and its innovations and advantages over existing methodologies, a brief description of an optimal control problem is given, and the available methods of solution and their limitations are briefly discussed.

A discrete time finite operating horizon optimal control problem can be simply stated as follows. Given an initial state vector $\mathbf{x}(0)$, find a policy, i.e., a sequence of controls $\{\mathbf{u}^*(k)\}_{k=1}^{N}$ as functions of the current state vector which minimize a given objective function (or its expected value) over all other policies, and which satisfies a specified set of constraints and the equations of system dynamics. Our concern is limited to cases for which the objective function and constraints are stagewise separable so that dynamic programming is applicable.

One of the oldest and most standard algorithms to this "optimal control problem" or "explicit stochastic optimization" is discrete dynamic programming (DDP) [cf. *Bellman* [1957]; *Bellman and Dreyfus*, 1962]. Such an approach requires the discretization of the state space (and, in most applications, of the control space) and solution of the optimization problem on each of the grid points. The exponential increase of the computer memory and computation time requirements with the number of state and control variables (Bellman called it the "curse of dimensionality"), limits the applicability of DDP to oligo-dimensional systems.

Much of the recent research on dynamic programming appears to deal with methods devised to overcome the limitations of discrete dynamic programming, and several useful methods have been proposed over the years. These methods, known as "successive approximation methods" include differential DP (see, for example, *Jacobson and Mayne* [1970] for unconstrained optimal control problems and *Murray and Yakowitz* [1979] for problems with linear constraints), discrete differential DP [*Heidari et al.*, 1971], state incremental DP [*Larson*, 1968, chapter 12], nonlinear programming algorithms [*Lee and Waziruddin*, 1970; *Gagnon et al.*, 1974; *Chu and Yeh*, 1978], and a discrete maximum principle algorithm [*Papageorgiou*, 1985]. In some of these methods discretization of the state space is completely avoided.

However, such successive approximation methods are not directly applicable to stochastic optimal control problems. The main reason is that due to the stochasticity of the input, no single-state trajectory can be projected with certainty. In-

Paper number 7W4971.
0043-1397/88/007W-4971$05.00

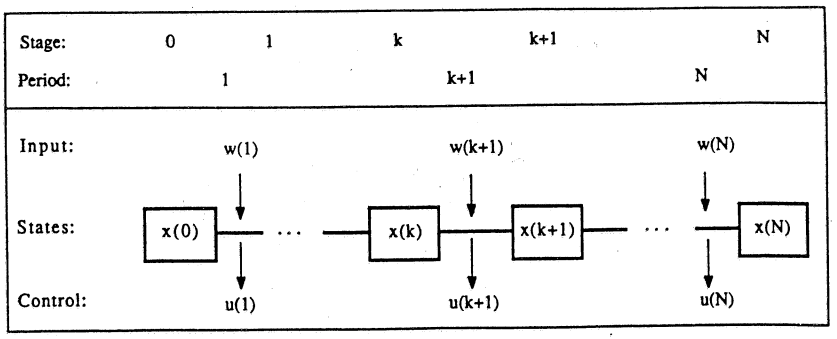

Fig. 1. Schematic representation of a general system and the variables involved.

stead, the whole optimal control policy over all states is required, so that integration over the range of states at the next stage can take place for the minimization of the expected cost. Thus apart from some approximate methods such as the small-perturbation approach of *Kitanidis* [1987], the "parameter iteration method" of *Gal* [1979], and some other methods reviewed in the work by *Yakowitz* [1982, section 5], the conventional discrete dynamic programming approach remains the only universal approach to stochastic optimal control problems (see, for example, *Larson and Casti* [1982, p. 120]). This essentially limits the dimensionality of the systems that can be solved under an explicit (and not implicit) stochastic framework. According to *Yakowitz* [1982],

> ... two reservoir systems are the largest to be reported solved by stochastic dynamic programming, whereas we have noted that deterministic reservoir systems of up to 10 reservoirs have been solved. This observation points to the motivation for making the deterministic assumption and underscores the need for research ideas for overcoming the computational burden of the stochastic case.

In this paper, we present an alternative DP technique which combines elements of conventional DDP (i.e., discrete state-space and backward stagewise optimization) with elements of constrained optimization (i.e., nonlinear programming with linear equality constraints) for the derivation of the optimal control over the continuous control space. The idea behind our method is that the cost to go and optimal control functions are approximated (within the hypercubes defined by the state discretization scheme) with piecewise Hermite interpolating polynomials. This higher order of approximation permits the use of fewer state discretization nodes (and therefore reduces the fast computer memory requirements) while still achieving high-accuracy solutions. Also, the continuity of the first derivative of the Hermitian approximation functions permits the use of efficient Newton-type schemes for the stagewise optimization.

The idea of interpolation in dynamic programming is not new. *Bellman and Dreyfus* [1962, chapter 12] used orthogonal polynomials for the approximation of the cost-to-go function. This global approximation, however, has several disadvantages as compared to local approximation. The main disadvantage is that functions hard to approximate in a particular domain of the state space will result in a poor approximation over the whole domain. Also, for fast changing functions, oscillatory approximations may be obtained unless many terms are used. *Daniel* [1976] and *Birnbaum and Lapidus* [1978] recognized the importance of using local approximations and explored the use of multidimensional B splines [e.g., *Schultz*, 1973]. Although splines provide approximations with continuous first and second derivatives, the first derivatives at the nodes are not explicitly preserved. This is important for optimal control problems where eventually only the first derivatives (and not the values of the function) are used in the computation of the optimal control. Besides, in many cases, the optimal knots of the splines must be determined (a time consuming process) or estimates of the derivatives so that a good spline approximation can be obtained. Of course, spline approximation permits the use of Newton-type methods for the stagewise optimization. This issue, although recognized by *Birnbaum and Lapidus* [1978], was not further explored in their work.

The algorithm proposed in this paper is similar in motivation but different in techniques from all previously proposed methods. It is termed gradient dynamic programming (GDP) because the gradient of the cost to go and optimal control functions with respect to all state variables are preserved at all nodes. This algorithm was briefly introduced by the authors [*Kitanidis and Foufoula-Georgiou*, 1987] in an effort to obtain methods with smaller discretization error than conventional DDP. In that work, however, only single-control optimization problems had been considered and the emphasis was on comparing GDP and DDP through an asymptotic error analysis. The encouraging theoretical and numerical results, namely, faster convergence to the "true" control policy and reduction in dimensionality in the sense that fewer nodes are needed to achieve a given degree of accuracy, motivated the extension of our efforts to the optimization of multistate, multicontrol systems. In the present paper, we present the methodology of GDP and the technical issues involved in its implementation. The application of the proposed method to the deterministic and stochastic optimal control of multireservoir systems is demonstrated in a four-reservoir example which *Yakowitz* [1982, p. 683] describes as being "probably beyond the scope of discrete dynamic programming because of the curse of dimensionality."

2. Terminology and Preliminaries

Before we embark on the description of the gradient dynamic programming (GDP) method, some terminology is in order. Let N denote the number of decision times (stages), n the dimension of the state vector $\mathbf{x}$, m the dimension of the control vector $\mathbf{u}$, and r the dimension of a random forcing function (input) $\mathbf{w}$. As illustrated in Figure 1, $\mathbf{x}(k)$ is the state vector at the beginning of period k, and $\mathbf{u}(k)$ and $\mathbf{w}(k)$ are the control and random input vectors, respectively, during period k.

For a deterministic system, $\mathbf{w}(k)$ is a known input vector, e.g., mean inflows during period k. For a stochastic system $\mathbf{w}(k)$ is a random vector with known probability density func-

tion $p(\mathbf{w}(k))$. Without loss of generality we may assume that the random vectors $\mathbf{w}(k)$, $k = 1, \cdots, N$ are independent of each other. Note that serially and cross-correlated inputs can be accounted for through state augmentation.

2.1. *System Dynamics*

Consider a system whose dynamics are described by the state transition function $\mathbf{T}_k$ such that

$$\mathbf{x}(k+1) = \mathbf{T}_k(\mathbf{x}(k), \mathbf{u}(k+1), \mathbf{w}(k+1)) \qquad (1)$$

$$k = 0, 1, \cdots, N-1$$

Note that $\mathbf{T}_k$ is an n-dimensional vector function dependent on the stage k. In the developments that follow we restrict our attention to linear dynamics. This limitation is mainly imposed from a desire to have only linear constraints at the optimization step. A commonly used formulation of (1) in reservoir systems is

$$\mathbf{x}(k+1) = \Phi(k)\mathbf{x}(k) + \Psi(k)\mathbf{u}(k+1) + \mathbf{q}(k+1) \qquad (2)$$

(see, for example, *Kitanidis* [1987]); $\Phi(k)$ and $\Psi(k)$ are known matrices and $\mathbf{q}(k+1)$ is the vector of inflows. Note that for a deterministic system, the state at stage $(k+1)$ is completely determined by the state at stage k and the transition function $\mathbf{T}_k$. For a stochastic system $\mathbf{x}(k+1)$ belongs to a set of state vectors determined by the probability density function of the random vector $\mathbf{w}(k+1)$.

It is assumed throughout this work that the functions involved possess continuous first and second derivatives with respect to the state and control vectors. Let $\partial\mathbf{T}_k/\partial\mathbf{x} := \partial\mathbf{T}_k/\partial\mathbf{x}(k)$ and $\partial\mathbf{T}_k/\partial\mathbf{u} := \partial\mathbf{T}_k/\partial\mathbf{u}(k+1)$ denote the Jacobians of $\mathbf{T}_k := \mathbf{T}_k(\mathbf{x}(k), \mathbf{u}(k+1), \mathbf{w}(k+1))$ with respect to the state and control vectors, respectively. For instance, the ijth element of $\partial\mathbf{T}_k/\partial\mathbf{x}$ is $\partial T_{k,i}/\partial x_j$, where $T_{k,i}$ denotes the ith row of $\mathbf{T}_k$. For a system with linear dynamics, $\partial\mathbf{T}_k/\partial\mathbf{x}$ and $\partial\mathbf{T}_k/\partial\mathbf{u}$ simply reduce to the matrices $\Phi(k)$ and $\Psi(k)$, respectively.

2.2. *System Constraints*

We restrict our attention to linear constraints resulting from linear transition equations. A typical set of constraints will include lower and upper bounds on the control and state variables and functional inequalities among control and state variables. For instance, a reservoir control problem will have constraints of the type

$$\mathbf{u}^{\min}(k+1) \le \mathbf{u}(k+1) \le \mathbf{u}^{\max}(k+1) \qquad (3a)$$

$$\mathbf{x}^{\min}(k+1) \le \mathbf{x}(k+1) = \mathbf{T}_k(\mathbf{x}(k), \mathbf{u}(k+1), \mathbf{w}(k+1))$$

$$\le \mathbf{x}^{\max}(k+1) \qquad k = 0, 1, \cdots, N-1 \qquad (3b)$$

where the control variables are reservoir releases and the state variables are storages. Using simple operations any such system of l linear constraints can be brought into the form

$$A\mathbf{u}(k+1) \le \mathbf{b} \qquad (4)$$

where A is an $(l \times m)$ matrix and $\mathbf{b}$ is an $(l \times 1)$ vector of known coefficients. Note that the coefficient matrices A and $\mathbf{b}$ in (4) depend on the decision time k and the initial state vector $\mathbf{x}(k)$. For a stochastic optimization problem the constraints on the random vector $\mathbf{x}(k+1)$ are introduced in a probabilistic sense:

$$Pr\{\mathbf{x}(k+1) \le \mathbf{x}^{\min}(k+1)\} \le \boldsymbol{\alpha} \qquad (5a)$$

$$Pr\{\mathbf{x}(k+1) \ge \mathbf{x}^{\max}(k+1)\} \le \boldsymbol{\beta} \qquad (5b)$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are vectors of given probabilities. Then, based on the known probability distribution function of $\mathbf{x}(k+1)$, the deterministic equivalents of the above chance constraints are used in lieu of (3*b*).

2.3. *Objective Function*

For a deterministic discrete time optimal control problem, the objective is to find the control policy $\{\mathbf{u}^*(k)\}$, $k = 1, \cdots, N$ which minimizes the performance criterion

$$J = \sum_{k=0}^{N-1} C_k(\mathbf{x}(k), \mathbf{u}(k+1)) + F_N(\mathbf{x}(N)) \qquad (6)$$

given an initial state vector $\mathbf{x}(0)$. The performance criterion (objective function) consists of the sum of the single-stage cost functions $C_k(\mathbf{x}(k), \mathbf{u}(k+1))$ over the whole operating horizon and a terminal cost $F_N(\mathbf{x}(N))$. Note that the objective function, as well as the constraints, meets the dynamic programming requirement of being decomposable in stages.

In the stochastic case, the objective function is replaced by the expected value of the expression of (6), i.e.,

$$J = \underset{\mathbf{w}(1), \cdots, \mathbf{w}(N)}{E} \sum_{k=0}^{N-1} C_k[\mathbf{x}(k), \mathbf{u}(k+1)] + F_N[\mathbf{x}(N)] \qquad (7)$$

where expectation is taken with respect to the random vectors $\mathbf{w}(1), \cdots, \mathbf{w}(N)$. In concise notation, let $C_k := C_k(\mathbf{x}(k), \mathbf{u}(k+1))$ denote the loss function at stage k and $\nabla_u C_k := \partial C_k/\partial\mathbf{u}(k+1)$ denote the gradient of C_k with respect to the m-dimensional control vector $\mathbf{u}(k+1)$, that is,

$$\nabla_u C_k = (\partial C_k/\partial u_1, \cdots, \partial C_k/\partial u_m)$$

Similarly, we define $\nabla_x C_k := \partial C_k/\partial\mathbf{x}(k)$, the gradient of C_k with respect to the state vector $\mathbf{x}(k)$. The Hessian matrix of C_k with respect to the state and control vectors is composed of the blocks $C_{k,xx}$, $C_{k,xu}$, and $C_{k,uu}$ where, for example, the ijth element of $C_{k,xu}$ is $\partial^2 C_k/\partial x_i \partial u_j$.

2.4. *Cost-To-Go Function (Optimal Cost Function)*

Let $F_k := F_k(\mathbf{x}(k))$ denote the cumulative cost associated with the state vector $\mathbf{x}(k)$ and the optimal control policy from k to the end of the operating horizon. We will refer to this function as the cost to go at stage k (or with N-k periods to go). In a deterministic backward moving dynamic programming scheme, the iterative functional equation of the system can be written as

$$F_{k-1}(\mathbf{x}(k-1)) = \min_{\mathbf{u}(k)} \{C_{k-1}(\mathbf{x}(k-1), \mathbf{u}(k)) + F_k[\mathbf{x}(k) \equiv \mathbf{T}_{k-1}(\mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k))]\} \quad k = 1, \cdots, N \qquad (8)$$

with terminal condition $F_N(\mathbf{x}(N))$, a given function of the final storage. For a stochastic system, the functional equation takes the form

$$F_{k-1}(\mathbf{x}(k-1)) = \min_{\mathbf{u}(k)} \{C_{k-1}(\mathbf{x}(k-1), \mathbf{u}(k)) + \underset{\mathbf{w}(k)}{E} F_k[\mathbf{x}(k) \equiv T_{k-1}[\mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k)]]\} \qquad (9)$$

$$k = 1, \cdots, N$$

In the above equation,

$$\underset{\mathbf{w}(k)}{E} F_k(\mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k)) = \int_{w_1(k)} \cdots \int_{w_r(k)} F_k(\mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k)) \cdot f_{1,r}^{(k)}[w_1(k), \cdots w_r(k)]\, dw_1(k) \cdots dw_r(k)$$

where $f_{1:r}^{(k)}(w_1(k), \cdots, w_r(k))$ is the joint pdf of the random variables $w_i(k)$, $i = 1, \cdots, r$ during period k.

We complete the terminology by letting $\nabla_x F_k := dF_k(\mathbf{x}(k))/d\mathbf{x}(k)$ and $\nabla_u F_k := dF_k(\mathbf{x}(k))/d\mathbf{u}(k)$ denote the gradients of F_k with respect to the state and control vectors, respectively. The Hessian of F_k is composed of blocks $F_{k,xx}$, $F_{k,xu}$, and $F_{k,uu}$ defined the same way as for the single-stage loss function. In the next section we discuss the general methodology of gradient dynamic programming.

3. General Description of the Gradient Dynamic Programming Method

Based on the principle of optimality [*Bellman*, 1957] any multistage optimization problem with objective function and constraints which are stagewise separable may be decomposed through dynamic programming into a sequence of single-stage optimization problems. This section describes the gradient dynamic programming methodology at a typical stage. For simplicity, the development of the equations is carried out for the deterministic case. The method is easily extended to stochastic optimization, as will be illustrated in the next section.

The state space is discretized and represented by a finite number of nodes (state vectors $\mathbf{x}$). Assume that at stage k, the values of the cost-to-go function $F_k(\mathbf{x}(k))$ and the values of its first derivatives $\nabla_x F_k = dF_k(\mathbf{x}(k))/d\mathbf{x}(k)$ are known for all the grid points, i.e., all discrete state vectors $\mathbf{x}(k)$. These values are known at the last operation period, $k = N$, and can be explicitly updated from stage to stage as the algorithm moves backward in time as will be shown in the sequel. Let $\mathbf{x}(k-1)$ denote a particular grid point at stage $k-1$. It is desired to (1) determine the optimal control $\mathbf{u}^*(k)$ associated with $\mathbf{x}(k-1)$; (2) compute the Jacobian of $\mathbf{u}^*(k)$ with respect to the state vector $\mathbf{x}(k-1)$; and (3) compute the values of $F_{k-1}(\mathbf{x}(k-1))$ and $\nabla_x F_{k-1} = dF_{k-1}/d\mathbf{x}(k-1)$. Once this is done for all possible state vectors $\mathbf{x}(k-1)$, the solution to the single-stage optimization problem has been completed.

3.1. *Approximation of* F_k

F_k is approximated within each n-dimensional hypercube (defined by the nodal points of the state vector $\mathbf{x}(k)$) through a Hermitian interpolation of the known values of F_k and its gradient $\nabla_x F_x$ at all the nodes defining the hypercube. The construction of this approximation polynomial is given in Appendix A. In particular, F_k is written in the form of (A1) where the basis functions ϕ_i and ψ_{ij} are defined in (A2)–(A4) in terms of the local coordinates of any point $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ within the n-dimensional hypercube.

3.2. *Determination of* $\mathbf{u}^*(k)$: *No Constraint Binding*

If no constraint is binding, $\mathbf{u}^*(k)$ is the solution to the system of equations obtained by differentiating the cost-to-go function with respect to the control variables and setting the derivatives to zero:

$$\nabla_u C_{k-1} + \nabla_u F_k = 0 \tag{10}$$

or, through application of the chain rule of differentiation,

$$\nabla_u C_{k-1} + \nabla_x F_k\left(\frac{\partial T_{k-1}}{\partial \mathbf{u}}\right) = 0 \tag{11}$$

Equation (11) represents the first-order necessary conditions for the optimum. The second-order condition for $\mathbf{u}^*(k)$ to be a unique local minimum is that the Hessian is positive definite, or symbolically,

$$H = C_{k-1,uu} + \left[\frac{\partial T_{k-1}}{\partial \mathbf{u}}\right]^T F_{k,xx}\left[\frac{\partial T_{k-1}}{\partial \mathbf{u}}\right] > 0 \tag{12}$$

Newton's method for unconstrained optimization [cf. *Luenberger*, 1984] can be used for the determination of $\mathbf{u}^*(k)$. In a Newton-type approach the basic iteration is

$$\mathbf{u}^{i+1} = \mathbf{u}^i - \rho_i R_i \mathbf{g}_i$$

where $\mathbf{u}^i$ is the vector of parameters in the ith iteration, $\mathbf{g}_i$ is the gradient (given in equation (11)) of the function to be minimized, R_i is the inverse of the Hessian matrix of (12) or an approximation thereof, and ρ_i is a scalar step size parameter which may be used to optimize the one-dimensional search in the direction $R_i\mathbf{g}_i$ in the case of nonquadratic terms. Note that in solving (11), $\nabla_x F_k$ is evaluated at the state vector $\mathbf{x}(k) = \mathbf{T}_{k-1}(\mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k))$ which may not coincide with one of the grid state vectors at stage k for which the values of F_k and $\nabla_x F_k$ are available. In that case the approximation of F_k at the state vector $\mathbf{x}(k)$ is used as computed in section 3.1. Also, $\nabla_x F_k$ and the matrix $F_{k,xx}$ of second derivatives needed for the evaluation of the Hessian in (12) are computed through differentiation. These equations are given for completeness in Appendix B.

3.3. *Determination of* $\mathbf{u}^*(k)$: *Binding Constraints*

If one or more of the constraints is binding, then constrained optimization methods may be used for the determination of $\mathbf{u}^*(k)$. They include primal, penalty and barrier, dual and cutting plane, and Lagrange methods [cf. *Gill and Murray*, 1974; *Fletcher*, 1981; *Luenberger*, 1984]. We have chosen to work with a primal method, i.e., a method which stays inside the feasible region during the search for the optimum. The many advantages of primal methods are described in *Luenberger* [1984, p. 323]. A particularly attractive feature for the problem at hand is that if the search is terminated before the solution is reached, the terminating point is guaranteed to be feasible and near the optimum. Thus it may provide a solution acceptable for all practical purposes or at least a good starting point if the procedure is reinitialized. For problems with linear constraints their convergence rates are hard to beat. A computational disadvantage associated with any primal method is the requirement of a phase 1 procedure for the determination of an initial feasible solution [see *Luenberger*, 1984]. In most practical cases, however, an initial feasible solution can be trivially determined, as for example, by setting the releases to zero or to the values of the inputs. Careful selection of the initial solutions can significantly improve the computational efficiency of the algorithm (Jery Stedinger and coworkers, Cornell University, personal communication, 1987).

We will briefly describe here an active set strategy which was found to work well with sample problems. Active set methods [cf. *Luenberger*, 1984; *Fletcher*, 1981] have unique computational advantages. The inequality constraints are partitioned into active (treated as equality constraints) and slack (essentially ignored). The working set is adjusted at each step of the iterative solution procedure. The basic components of an active set method are (1) determination of the current working set of active constraints by applying an efficient procedure for adding and dropping constraints from the previous working set and (2) a procedure for moving toward the optimum subject to the constraints prescribed by the current working set. Active set methods are much more efficient than

branch-and-bound procedures but may fail to converge ("zigzagging"). For their convergence to be guaranteed, some weak convexity requirements must be met [*Fletcher*, 1981, p. 113]. These conditions are usually met in applications and the popularity of these methods has increased significantly in the last ten years. In the reservoir operation problem they are often used in conjunction with successive approximation methods [e.g., *Murray and Yakowitz*, 1979; *Georgakakos and Marks*, 1985]. Sometimes, minor refinements based on an understanding of the problem at hand may be needed to guarantee convergence and improve efficiency. *Lenard* [1975] presents a computational study of active set strategies and suggests that highest efficiency is achieved by starting with as small a set of active constraints as possible.

Let $G\mathbf{u}(k) = \mathbf{d}$ define the working set where G is a $(p \times m)$ matrix of known coefficients of rank p (the rows of G are linearly independent) and $\mathbf{d}$ is an $(p \times 1)$ vector of known coefficients. The optimal control $\mathbf{u}^*(k)$ will be the solution to the constrained optimization problem:

minimize

$$f_k(\mathbf{u}(k)) := \{C_{k-1}(\mathbf{x}(k-1), \mathbf{u}(k)) + F_k[\mathbf{T}_{k-1}(\mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k))]\} \quad (13)$$

subject to

$$G\mathbf{u}(k) = \mathbf{d} \quad (14)$$

This problem is solved using an iterative Newton-type method for moving optimally within a working set (see Appendix D for details and also *Luenberger*, [1984, chapter 11]). If $\mathbf{u}^0(k)$ denotes an initial feasible solution vector, i.e., one which satisfies (11), the new improved solution at the next iteration will be

$$\mathbf{u}^1(k) = \mathbf{u}^0(k) + \Delta\mathbf{u}(k) \quad (15)$$

where $\Delta\mathbf{u}(k)$ is the solution to the linear system of equations

$$\left[\begin{array}{c:c} f_{k,uu} & G^T \\ \hdashline G & 0 \end{array}\right]\left[\begin{array}{c} \Delta\mathbf{u}(k) \\ \hdashline \boldsymbol{\lambda} \end{array}\right] = \left[\begin{array}{c} -(\nabla_u f_k)^T \\ \hdashline 0 \end{array}\right] \quad (16)$$

where

$$\nabla_u f_k = \nabla_u C_{k-1} + \nabla_x F_k(\partial \mathbf{T}_{k-1}/\partial \mathbf{u}) \quad (17)$$

$$f_{k,uu} = C_{k-1,uu} + \left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right)^T F_{k,xx}\left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right) \quad (18)$$

and where $\boldsymbol{\lambda}$ is a $(p \times 1)$ vector of Lagrange multipliers. The above solution is based on an approximation of the cost to go with a quadratic function of the control. Details can be found in Appendix D.

One may easily verify that since $\mathbf{u}^0$ satisfies the working set of constraints, so does $\mathbf{u}^1$. Note that if f_{uu} is a symmetric and positive definite matrix on the subspace $M = \{\mathbf{u}: G\mathbf{u} = 0\}$ and G is a $(p \times m)$ matrix of rank p then the $(m + p) \times (m + p)$ matrix

$$\begin{bmatrix} f_{uu} & G^T \\ G & 0 \end{bmatrix}$$

is nonsingular [*Luenberger*, 1984, p. 424]. As is seen from (13)–(15), at every iteration the evaluation of $\nabla_x F_k$ and $F_{k,xx}$ is required and this is accomplished through differentiation of the Hermitian interpolation function for F_k using the formulae in Appendix B.

At this point, the currently inactive constraints are checked under the new solution $\mathbf{u}^1(k)$ and any violated constraints are added to the working set. The new active set is checked to verify that the rank of the G matrix is equal to the number of its rows. If this is not the case, redundant constraints are removed. The constrained optimization is now performed under the new active set, and the procedure is repeated until a solution (within the provided stopping criteria) is reached. At this point, the Lagrange multipliers $\boldsymbol{\lambda}$ are checked and any active constraint whose corresponding λ_i is negative is dropped from the active set and the constrained optimization is repeated with the new working set. If none of the λ_i are negative, the solution is accepted. The relaxation of a constraint based on the sign of the corresponding Lagrange multipliers follows directly from the Kuhn-Tucker conditions or from the sensitivity interpretation of Lagrange multipliers [see *Luenberger*, 1984, p. 328].

According to the active set theorem [*Luenberger*, 1984, p. 329], convergence will occur after only a finite number of working sets. Within a working set a Newton-type method guarantees quadratic convergence to the optimum. In theory, the correct sign of the Lagrange multipliers, which determines which constraints are dropped from an active set, is only guaranteed at the exact global optimum, and therefore acceptance of a new optimum solution does not guarantee that the current working set will not be encountered again. In practice, however, zigzagging is rarely encountered and in most cases the active set method works very effectively.

3.4. *Computation of the Jacobian* $d\mathbf{u}_k^*/d\mathbf{x}$

The optimum $\mathbf{u}_k^*(\mathbf{x}(k-1))$, abbreviated as $\mathbf{u}_k^*$, must satisfy the Kuhn-Tucker condition at any point $\mathbf{x} \equiv \mathbf{x}(k-1)$. In this case, assuming that $G\mathbf{u} = \mathbf{d}$ represents the active constraints at the optimum, one has

$$(\nabla_u f_k)^T + G^T\boldsymbol{\lambda} = 0 \quad (19a)$$

$$G\mathbf{u}^* = \mathbf{d} \quad (19b)$$

where $\boldsymbol{\lambda} > 0$ and $\mathbf{d}$ is a function of $\mathbf{x}(k-1)$. These equations are satisfied for any values of $\mathbf{x}$, $\mathbf{u}^*$, and $\boldsymbol{\lambda}$. If $\mathbf{x}$ is replaced by $\mathbf{x} + \delta\mathbf{x}$, where $\delta\mathbf{x}$ represents an infinitesimal increment, then the control, the Lagrange multipliers, and the vector of the constraints $\mathbf{d}$ change into $\mathbf{u}^* + (d\mathbf{u}^*/d\mathbf{x})\delta\mathbf{x}$, $\boldsymbol{\lambda} + (d\boldsymbol{\lambda}/d\mathbf{x})\delta\mathbf{x}$, and $\mathbf{d} + (d\mathbf{d}/d\mathbf{x})\delta\mathbf{x}$, respectively, so that (19) is still satisfied:

$$(\nabla_u f_k)^T + \nabla_x(\nabla_u f_k)^T\left(\frac{d\mathbf{u}^*}{d\mathbf{x}}\right)\delta\mathbf{x} + \nabla_x(\nabla_u f_k)^T\delta\mathbf{x} + G^T\boldsymbol{\lambda} + G^T\left(\frac{d\boldsymbol{\lambda}}{d\mathbf{x}}\right)\delta\mathbf{x} = 0$$

$$G\mathbf{u}^* + G\left(\frac{d\mathbf{u}^*}{d\mathbf{x}}\right)\delta\mathbf{x} = \mathbf{d} + \left(\frac{d\mathbf{d}}{d\mathbf{x}}\right)\delta\mathbf{x}$$

Using (19) these equations are simplified into

$$\left[\begin{array}{c:c} f_{k,uu} & G^T \\ \hdashline G & 0 \end{array}\right]\left[\begin{array}{c} d\mathbf{u}^*/d\mathbf{x} \\ \hdashline d\boldsymbol{\lambda}/d\mathbf{x} \end{array}\right] = \left[\begin{array}{c} -\nabla_x(\nabla_u f_k)^T \\ \hdashline d\mathbf{d}/d\mathbf{x} \end{array}\right] \quad (20)$$

We remind that $d\mathbf{u}^*/d\mathbf{x}$ is an $(m \times n)$ matrix whose ijth element is du_i^*/dx_j; $d\boldsymbol{\lambda}/d\mathbf{x}$ is a $(p \times n)$ matrix whose ijth element is $d\lambda_i/dx_j$; $f_{k,uu} = \nabla_x(\nabla_u f_k)^T$ is the Hessian of f_k with respect to $\mathbf{u}$ (equation (18)); and

$$\nabla_x(\nabla_u f_k)^T = C_{k-1,ux} + \left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right)^T F_{k,xx} \quad (21)$$

3.5. *Evaluation of F_{k-1} and $\nabla_x F_{k-1}$*

Once the optimal control and its Jacobian with respect to the state vector $\mathbf{x}(k-1)$ has been determined, the cost to go and its gradient may be calculated. The optimal control $\mathbf{u}^*(k)$ is a function of $\mathbf{x}(k-1)$. Then from (7)

$$F_{k-1}(\mathbf{x}(k-1)) = C_{k-1}[\mathbf{x}(k-1), \mathbf{u}_k^*(\mathbf{x}(k-1))] + F_k\{\mathbf{x}(k) = T_{k-1}[\mathbf{x}(k-1), \mathbf{u}_k^*(\mathbf{x}(k-1)), \mathbf{w}(k-1))]\} \quad (22)$$

Substituting for the calculated optimum at $\mathbf{x}(k-1)$, F_{k-1} may be found. By differentiating (22) with respect to $\mathbf{x}(k-1)$ and using the compact notation introduced earlier, one arrives at

$$\nabla_x F_{k-1} = \nabla_x C_{k-1} + \nabla_u C_{k-1}\left(\frac{d\mathbf{u}_k^*}{dx}\right) + \nabla_x F_k\left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{x}}\right) + \nabla_x F_k\left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right)\left(\frac{d\mathbf{u}_k^*}{dx}\right) \quad (23)$$

where $d\mathbf{u}_k^*/d\mathbf{x}$ is the Jacobian of $\mathbf{u}_k^*/(\mathbf{x}(k-1))$ with respect to the state vector $\mathbf{x}(k-1)$.

Rearranging (23) gives

$$\nabla_x F_{k-1} = \nabla_x C_{k-1} + \nabla_x F_k \frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{x}} + \left[\nabla_u C_{k-1} + \nabla_x F_k \frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right]\frac{d\mathbf{u}_k^*}{dx} \quad (24)$$

Note that if no constraint is binding, the expression in the brackets is zero and the Jacobian $d\mathbf{u}_k^*/d\mathbf{x}$ does not need to be calculated. If at least one constraint is violated, $\nabla_x F_{k-1}$ is obtained by substituting in (24) the value of $(d\mathbf{u}_k^*/d\mathbf{x})$ obtained from (20).

4. Stochastic Gradient Dynamic Programming

Gradient dynamic programming is extended to stochastic optimization in a straightforward way. One has simply to replace the expressions involving the cost-to-go function and its first and second derivatives by their expectations. Hence assuming that the technical conditions for interchanging the order of differentiation and expectation are satisfied, for the determination of the optimal control $\mathbf{u}^*(k)$ the first-order necessary condition becomes

$$\nabla_u C_{k-1} + \underset{\mathbf{w}(k)}{E}\left[\nabla_x F_k\left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right)\right] = 0 \quad (25)$$

while the Hessian becomes

$$H = C_{k-1,uu} + \underset{\mathbf{w}(k)}{E}\left[\left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right)^T F_{k,xx}\left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right)\right] \quad (26)$$

The Jacobian $d\mathbf{u}^*/d\mathbf{x}$ is again computed from (20) where now $f_{k,uu}$ is given by (26) and

$$\nabla_x(\nabla_u f_k)^T = C_{k-1,ux} + \underset{\mathbf{w}(k)}{E}\left[\left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\right)^T F_{k,xx}\right] \quad (27)$$

Once $\mathbf{u}^*(k)$ and $d\mathbf{u}_k^*/d\mathbf{x}$ corresponding to the state vector nodal point $\mathbf{x}(k-1)$ have been determined, F_{k-1} and $\nabla_x F_{k-1}$ can be evaluated from

$$F_{k-1}(\mathbf{x}(k-1)) = C_{k-1}(\mathbf{x}(k-1), \mathbf{u}^*(k)) + \underset{\mathbf{w}(k)}{E}\left[F_k(\mathbf{x}(k-1), \mathbf{u}^*(k), \mathbf{w}(k))\right] \quad (28)$$

$$\nabla_x F_{k-1} = \nabla_x C_{k-1} + \nabla_u C_{k-1}\left(\frac{d\mathbf{u}_k^*}{d\mathbf{x}}\right) + \underset{\mathbf{w}(k)}{E}\left[\nabla_x F_x\left(\frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{x}} + \frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{u}}\frac{d\mathbf{u}_k^*}{d\mathbf{x}}\right)\right] \quad (29)$$

Note that in the case of linear dynamics we are considering, $(\partial \mathbf{T}_{k-1}/\partial \mathbf{u})$ and $(\partial \mathbf{T}_{k-1}/\partial \mathbf{x})$ are constant matrices and one needs only to replace F_k, $\nabla_x F_k$, and $F_{k,xx}$ in the equations of the deterministic case by their expectations with respect to $\mathbf{w}(k)$.

For the numerical evaluation of the expectation $E_{\mathbf{w}(k)}$ the distribution function of $\mathbf{w}(k)$ is discretized. Let r denote the dimension of the random vector $\mathbf{w}(k)$ during period k; D_i, $i = 1, \cdots, r$ the discretization level (i.e., number of nodes) of the ith random variable $w_i(k)$; and $p_{i,d_i}^{(k)} = \text{prob}\,\{w_i(k) = [w_i(k)]_{d_i}\}$, $d_i = 1, \cdots, D_i$, where $[w_i(k)]_{d_i}$ denotes the value of $w_i(k)$ on the node d_i of the discretized probability density function (pdf). Then, for any function $h(\mathbf{w}(k))$

$$\underset{\mathbf{w}(k)}{E}[h(\mathbf{w}(k))] = \sum_{d_1=1}^{D_1} \cdots \sum_{d_r=1}^{D_r} p_{1,d_1}^{(k)} \cdots p_{r,d_r}^{(k)} h([w_1(k)]_{d_1}, \cdots, [w_r(k)]_{d_r}) \quad (30)$$

Note that in the above equation the assumption of independence of $\mathbf{w}(k)$ has been invoked without loss of generality as discussed in section 2. In many cases, the random variables $w_i(k)$, $i = 1, \cdots, r$ will have the same probability distribution over all the operation periods $k = 1, \cdots, N$ and the terminology and equations would simplify. However, the consideration of the general case of different pdf's and different discretization levels for each random variable and each operating period does not pose any computational difficulties.

It should be emphasized that the high-speed memory requirements of the stochastic case remain the same as for the deterministic case. Only the computation time increases by a factor of $\prod_{i=1}^{r} D_i$. It is therefore advantageous to discretize the pdf of the random inputs as coarsely as possible while keeping the accuracy of integration within the desired limits. In choosing an efficient pdf discretization scheme one can take advantage of results on numerical quadrature (see, for example, *Engels* [1980] and *Abramowitz and Stegun* [1972, chapter 25]). In general, the most efficient scheme will depend on the shape of the pdf and the curvature (smoothness) of the function to be integrated. For example, for normally or lognormally distributed random inputs and for the local approximations of the cost-to-go function considered herein Hermitian integration provides an effective choice (see Appendix E).

5. Algorithmic Description of the Gradient Dynamic Programming Method

The state space is discretized and represented by a finite number of nodes (state vectors $\mathbf{x}$). At every stage and at each node the optimal control $\mathbf{u}^*$ is iteratively calculated. Optimization proceeds backwards in time, so that the first optimizations correspond to one period to go ($k = N$). Below we give an algorithmic description of GDP for one state vector $\mathbf{x}(k-1)$ during stage $(k-1)$. The known quantities at stage k are F_k, $\nabla_x F_k$, $\mathbf{u}^*(k+1)$, and $d\mathbf{u}_{k+1}^*/d\mathbf{x} = d\mathbf{u}_{k+1}^*(\mathbf{x}(k))/d\mathbf{x}(k)$ which have been stored off-line for all the nodal state vectors $\mathbf{x}(k)$ from a previous run. The procedure is repeated for all discretized state vectors $\mathbf{x}(k-1)$ and for all periods $k = N, \cdots, 1$.

5.1. *Step 0*

Approximate F_k using piecewise Hermite interpolation polynomials. These polynomials preserve the values F_k of the cost-to-go function and the values of its first derivatives $\nabla_x F_k$ at all the nodal points of the n-dimensional state vector $\mathbf{x}(k)$. The construction of these polynomials within each n-dimensional hypercube is discussed in section 3.1. and also in Appendix A.

5.2. *Step 1*

Select a state vector nodal point $\mathbf{x}(k-1)$ at which the quantities $\mathbf{u}^*(k)$, $d\mathbf{u}_k^*/d\mathbf{x}$, F_{k-1}, and $\nabla_x F_{k-1}$ are to be computed.

5.3. *Step 2*

Find an initial feasible control vector $\mathbf{u}(k)$, that is, a control vector that satisfies the constrain set $A\mathbf{u} \leq \mathbf{b}$, where $A = A(\mathbf{x}(k-1))$ and $\mathbf{b} = \mathbf{b}(\mathbf{x}(k-1))$. An initial feasible solution is usually obtained by applying the phase I procedure of linear programming. *Luenberger* [1984] describes such a general procedure, although in many specific cases an initial solution may be obtained through simpler means. If the initial feasible solution makes none of the constraints binding or active (i.e., acts as equality affecting the solution), a Newton-type unconstrained optimization method is used as described in section 3.2. At each iteration the solution is checked for feasibility. If none of the constraints becomes binding or active the algorithm proceeds with the unconstrained optimization, otherwise it goes to step 3.

5.4. *Step 3*

Form the current active constraint set (working set) corresponding to the control vector $\mathbf{u} = \mathbf{u}(k)$, i.e.,

$$G\mathbf{u} = \mathbf{d} \tag{31}$$

where $G = G(\mathbf{x}(k-1))$ is a $(p \times m)$ matrix of known coefficients of rank p and $\mathbf{d} = (\mathbf{x}(k-1))$ is a $(p \times 1)$ vector of known coefficients.

5.5. *Step 4*

Perform one iteration of the constrained optimization problem:

$$\text{minimize } f(\mathbf{u}) \tag{32}$$

subject to

$$G\mathbf{u} = \mathbf{d}$$

i.e., determine the improvement $\Delta\mathbf{u}(k)$ and the vector of Lagrange multipliers $\boldsymbol{\lambda}$ as described in section 3.3. Upon convergence, i.e., $\Delta\mathbf{u} \leq \boldsymbol{\varepsilon}$ go to step 6. Otherwise, check if the new solution $\mathbf{u}' = \mathbf{u} + \Delta\mathbf{u}$ violates any of the constraints not presently in the working set. If none of these constraints is violated, set $\mathbf{u}$ to $\mathbf{u}'$ and continue the constrained optimization within the same working set. If at least one of the previously inactive constraints is violated, go to step 5.

5.6. *Step 5*

Add constraints to the active set and determine a feasible solution under the new active set. Let $G'\mathbf{u}' = \mathbf{d}'$ denote the new (updated) active set. It is desired to obtain a feasible solution for the constrained problem which is close to the previously obtained solution $\mathbf{u}'$. For this purpose, project $\mathbf{u}'$ on the feasible domain defined by the constraints of the new active set (see Appendix C for details). The new solution is $\mathbf{u}'' = \mathbf{u}' + \delta\mathbf{u}$ where

$$\delta\mathbf{u} = G'^T(G'G'^T)^{-1}[\mathbf{d}' - G'\mathbf{u}'] \tag{33}$$

This projection provides a computationally efficient way to obtain feasible solutions every time the active set is changing or as an alternative to the phase I procedure of having to solve a linear programming problem at each iteration. At this point set $\mathbf{u}$ equal to $\mathbf{u}''$ and return to step 3.

5.7. *Step 6*

Remove constraints from the active set. When the optimum within a working set is reached ($\Delta\mathbf{u} < \boldsymbol{\varepsilon}$ at step 4), then the signs of the Lagrange multipliers $\boldsymbol{\lambda}$ obtained from (16) are checked. If all λ_i are nonnegative, then the optimum solution has been found and we proceed to step 7. If, however, one or more of the λ_i are negative, then the corresponding constraints are dropped from the active set and the algorithm returns to step 3.

In updating the active set, it is computationally advantageous to remove or add only one constraint at a time. For example, only the constraint with the largest negative Lagrange multiplier may be removed from the active set when more than one negative Lagrange multiplier is encountered. By doing so, the matrix of constraints G changes only by one row and the projection solution in step 4 may be computed from the previous one by a simple updating procedure [cf. *Luenberger*, 1984, p. 361].

5.8. *Step 7*

At the optimum $\mathbf{u}^*(k)$ compute F_{k-1}, $\nabla_x F_{k-1}$, and $d\mathbf{u}_k^*/d\mathbf{x}$. For these computations the equations in sections 3.4 and 3.5 or their obvious extensions to the stochastic case are used. These values are stored off-line for use at the next optimization period.

5.9. *Step 8*

Repeat steps 1–7 for all the nodal points of the discretized vector $\mathbf{x}(k-1)$. The number of nodal points is $\prod_{i=1}^{n} N_i$, where N_i is the number of nodes of the ith component of the state vector $\mathbf{x}(k-1)$.

5.10. *Step 9*

Repeat steps 0–8 for all stages, i.e., for $k = N, \cdots, 1$.

5.11. *Step 10*

The final step involves a forward run to determine the optimal trajectory given an initial state vector $\mathbf{x}(0)$. For the deterministic case, the optimal trajectory over the whole operating horizon can be obtained at once. Notice that due to the approximation of the cost-to-go functions, $F(\mathbf{x}(0))$ will be approximately equal to the total cost computed using (6) or (7). For the stochastic case only the total cost $F(\mathbf{x}(0))$ and the first-stage optimal control $\mathbf{u}^*(1)$ can be determined since the future optimal controls depend on the yet unknown future inputs to the system. At the end of the first stage, however, the system is usually observed and the new starting vector $\mathbf{x}(1)$ determined. At this point a new stochastic optimization problem has to be solved for $(N-1)$ operating periods and $\mathbf{u}^*(2)$ is thus determined. If the system is not observed at the end of each operating period then a "suboptimal" trajectory can be obtained by making use of the mean values of the stochastic