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

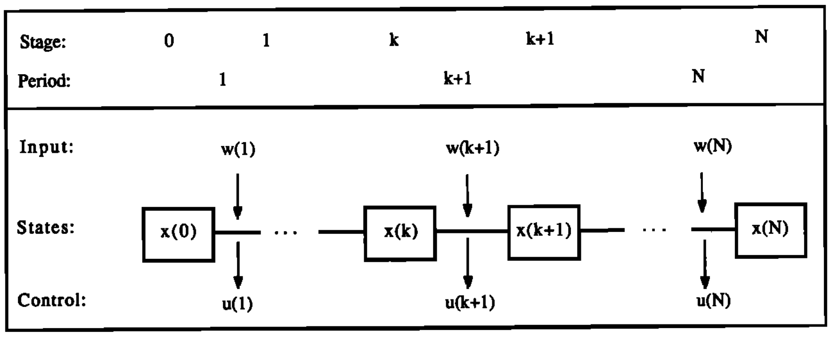

Fig. 1. Schematic representation of a general system and the variables involved.

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 function $p(\mathbf{w}(k))$. 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

$$\mathbf{x}(k+1) = \mathbf{T}_k(\mathbf{x}(k), \mathbf{u}(k+1), \mathbf{w}(k+1)) \qquad k = 0, 1, \cdots, N-1 \tag{1}$$

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

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

(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

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

$$\mathbf{x}^{\min}(k+1) \le \mathbf{x}(k+1) = \mathbf{T}_k(\mathbf{x}(k), \mathbf{u}(k+1), \mathbf{w}(k+1)) \le \mathbf{x}^{\max}(k+1) \qquad k = 0, 1, \cdots, N-1 \tag{3b}$$

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

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

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:

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

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

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

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

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.,

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

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

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

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

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

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:

$$\nabla_u C_{k-1} + \nabla_u F_k = 0 \qquad (10)$$

or, through application of the chain rule of differentiation,

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

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,

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

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

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

subject to

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

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

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

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

$$\begin{bmatrix} f_{k,uu} & G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}(k) \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -(\nabla_u f_k)^T \\ 0 \end{bmatrix} \tag{16}$$

where

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

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

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

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

$$G\mathbf{u}^* = \mathbf{d} \tag{19b}$$

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

$$\begin{bmatrix} f_{k,uu} & G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} d\mathbf{u}^*/d\mathbf{x} \\ d\boldsymbol{\lambda}/d\mathbf{x} \end{bmatrix} = \begin{bmatrix} -\nabla_x(\nabla_u f_k)^T \\ d\mathbf{d}/d\mathbf{x} \end{bmatrix} \tag{20}$$

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

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

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)

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

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

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

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

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

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

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

while the Hessian becomes

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

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

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

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

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

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

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 $h(\mathbf{w}(k))$

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

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., $\Delta\mathbf{u} \leq \varepsilon$ 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 ($\Delta\mathbf{u} < \varepsilon$ 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 inputs. Note that this trajectory will not be the same as the deterministic one because the computation of F_{k-1}, $\nabla_x F_{k-1}$, $\mathbf{u}_k^*$, and $d\mathbf{u}_k^*/d\mathbf{x}$ at each stage has taken into account the stochasticity of the inputs.

If $\mathbf{x}(0)$ or any of the subsequent state vectors $\mathbf{x}(k)$ does not fall on a nodal point then interpolation is needed for the determination of $\mathbf{u}^*(k+1)$. Our experience from the one-dimensional case [*Kitanidis and Foufoula-Georgiou*, 1987] suggests that the full advantages of gradient dynamic programming are realized when Hermitian interpolation is also used for the control function. This does not impose any computational difficulty or significant additional cost because the Jacobians $d\mathbf{u}_k^*/d\mathbf{x}$ needed for the interpolation of $\mathbf{u}_k^*$ are computed anyway for the determination of F_{k-1} and $\nabla_x F_{k-1}$ and are stored off-line. Only the computation time of the forward run is increased because now m Hermitian interpolations are needed as compared with the less accurate alternative of only one multilinear interpolation.

6. Computational Requirements of Gradient Dynamic Programming

We begin by briefly reviewing the computational requirements of the conventional DDP so that quantitative comparisons with those of gradient DP can be made. Let n denote the number of state variables, m the number of control variables, N_i the discretization level, i.e., number of nodes of the ith component of the state vector, M_i the discretization level of the ith control variable (refers to the conventional DDP), and N the number of stages. The computational requirements consist of high-speed memory requirements (HSMR), total amount of computation time (CT), and off-line storage, i.e., low-speed memory requirements (LSMR). For the conventional DDP with discretization of both state and control these requirements are well known (see, for example, *Larson and Casti* [1982, p. 230]).

$$\mathrm{HSMR} = \prod_{i=1}^{n} N_i \tag{34a}$$

$$\mathrm{CT} = \left(\prod_{i=1}^{n} N_i\right)\left(\prod_{j=1}^{m} M_j\right) N\Delta t_1 \tag{34b}$$

$$\mathrm{LSMR} = \left(\prod_{i=1}^{n} N_i\right) mN \tag{34c}$$

where Δt_1 is the computing time required for a single evaluation of the right-hand side of the functional equation (22) and a single scalar comparison. The function evaluation also includes computing time required to perform one interpolation (usually multilinear) of the state vector, unless the discretization of the control is such that only state vectors at the nodes are reached when moving from one state to the next. Note that this later case is not always feasible as for example in problems where the dimension of the state space is not equal to the dimension of the control space (noninvertible systems).

Observe that what causes the "curse of dimensionality" is the term $(\prod_{i=1}^{n} N_i)$ which determines the HSMR and is also directly involved in all other computational requirements. It is apparent that an approach to reduce the dimensionality of a problem would be to use coarser discretization schemes. As it was shown, however, in the work by *Kitanidis and Foufoula-Georgiou* [1987] one should use discretization schemes with caution with conventional DDP, since the discretization error introduced in the control is of the order of the discretization interval of the state. A natural approach to reducing the error in the control while keeping the same discretization interval in the states is to increase the accuracy of the description of the optimal cost function at each stage. This can be achieved by using a more elaborate interpolation scheme. The interpolation scheme proposed herein involves Hermitian approximation of the cost-to-go function within each of the n-dimensional hypercubes of the quantized state space. The error analysis in the work by Kitanidis and Foufoula-Georgiou suggests that under some conditions of smoothness of the cost-to-go functions, the Hermitian interpolation induces a discretization error in the control which is of the order of $(\Delta x)^3$, where Δx is the discretization interval of the state vector (assumed uniform for simplicity). This means that by halving the discretization interval of the states, the error in the control is reduced by a factor of one half in conventional DDP, whereas it is reduced by a factor of one eighth in GDP. Computational experience with one-dimensional cases and relatively smooth cost functions suggests that GDP with a small number of discretization nodes is comparable in terms of accuracy to conventional DDP with a considerably larger number of nodes.

The computational requirements of the GDP algorithm are

$$\mathrm{HSMR} = \left(\prod_{i=1}^{n} N_i'\right)(1+n) \tag{35a}$$

$$\mathrm{CT} = \left(\prod_{i=1}^{n} N_i'\right) N\Delta t_2 \tag{35b}$$

$$\mathrm{LSMR} = \left(\prod_{i=1}^{n} N_i'\right) m(1+n)N \tag{35c}$$

where N_i' is the new discretization level of the state vectors, Δt_2 is the computational time required to find $\mathbf{u}^*$ by solving an n-dimensional nonlinear optimization problem with p linear inequality constraints and compute at the optimum the values of F_k, $\nabla_x F_k$, and $d\mathbf{u}^*/d\mathbf{x}$.

Note from (35*a*) that GDP introduces a linear increase (as function of the dimensionality of the state space) into the HSMR as compared with the HSMR of DDP in (34*a*). This is because the first derivatives with respect to all state vectors are now stored on line to be used for the Hermitian interpolation. However, this linear increase is negligible compared with the exponential decrease due to the first term in (35*a*), where now N_i' need only be a fraction of N_i, i.e., $N_i' = N_i/c$, $c > 1$ and still obtain the same accuracy in the solution. Hence, the overall reduction factor is $(1+n)/c^n$ and the value of c depends on the smoothness of the cost-to-go function. It is worth noting that in most cases in practice, the dimension of the optimization problem (n) will be larger than the number of actual reservoirs involved, since augmentation of the state space is needed to account for serial correlations in the inflows. Also, note that the total computational time of GDP although it does not suffer from the exponential increase due to the discretization of the control, it might be of comparable size to that of DDP due to the larger computational time Δt_2.

For illustration purposes consider the following example: a system of 10 reservoirs and 10 controls, with discretization schemes involving 10 nodes at each state variable. The HSMR for DDP is 10^{10} positions while for GDP with four nodes is $4^{10}(1+10)$. (The assumption of four discretization nodes for GDP is based on our computational experience with the algo-

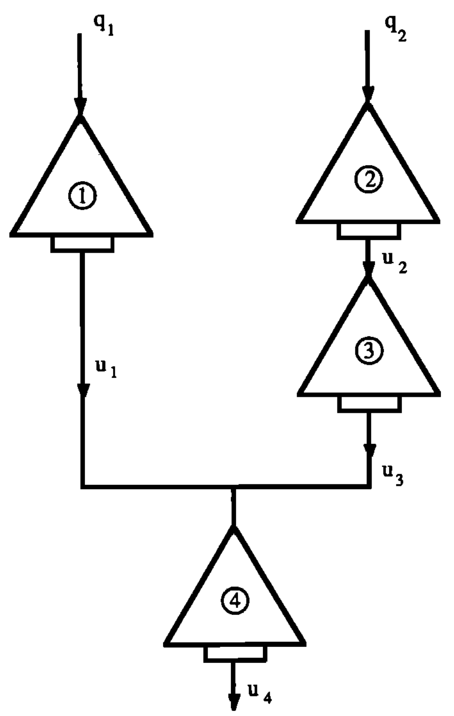

Fig. 2. Representation of the four-reservoir system used in the examples.

rithm and is supported by the example presented in the next section.) For a 32-bits/word machine and double precision arithmetic, the HSMR of DDP translates to a core of 8,000 Mbits, while for GDP to only 92 Mb (1 Mbit = 10^6 bytes), and these figures are only the core required to store the optimal cost function. While the first number is probably close to the capacity of many of today's computers the HSMR requirements of GDP are clearly not prohibiting.

The HSMR of the stochastic case is the same as that of the deterministic case. Only the total computation time is increased by a factor of $\prod_{i=1}^{r} D_i$, where D_i is the discretization level of the probability distribution function of the ith stochastic inflow and r is the number of stochastic inflows. Efficient discretization of the pdf of inflows is thus important in keeping the computational time of stochastic optimization within reasonable limits while still achieving the desired accuracy. The common approach to pdf discretization has been the use of equal probability intervals (usually five to ten) independently of the degree and smoothness of the cost-to-go function on which integration is performed (see, for example, *Weiner and Ben-Zvi* [1982]). As it is shown however in Appendix E, for a normal or lognormal pdf of inflows only two (or three) appropriately chosen nodal points can give exact integration of the pdf on functions of up to third (or sixth) degree (Hermitian integration). For an n-dimensional optimization problem, the piecewise Hermite approximation involves piecewise incomplete polynomials of degree $(n + 2)$ which reduce to third degree polynomials along each direction [see *Kitanidis*, 1986]. Thus for lognormally distributed inflows and cost-to-go functions F_k of appropriate smoothness, a three-point pdf discretization scheme will be exact for polynomials of up to degree $(2n - 1)$ (see Appendix E).

7. APPLICATION OF GDP TO MULTIRESERVOIR OPTIMIZATION

The GDP method is now applied to a multireservoir optimization problem. The purpose of this illustration is to demonstrate and test the applicability of the GDP algorithm to deterministic and stochastic optimization of multidimensional systems and provide grounds for discussion. No attempt is made to compare the proposed algorithm with other existing methods mainly because the only method that comparison would be appropriate with is the conventional DDP which would be computationally prohibiting for a large-dimensional system and a state discretization grid fine enough to assure accuracy in the solution. Such a comparative example has been presented for a one-dimensional study by *Kitanidis and Foufoula-Georgiou* [1987]. In that study, the conventional DDP and GDP algorithms were compared in terms of performance (i.e., convergence to the true solution) for both deterministic and stochastic optimization and for various discretization schemes. Under appropriate smoothness requirements for the cost-to-go function, the results of the one-dimensional case (i.e., good performance of GDP even with coarse discretization schemes as compared with DDP with much finer discretization) are expected to carry over to multidimensional systems as well.

The system chosen for illustration is the four-reservoir problem of Figure 2 which has served as an illustrative example in many studies. This example was first introduced into the literature by *Larson* [1968] for the purpose of illustrating the method of incremental dynamic programming. Subsequently, *Heidari et al.* [1971] used it as an example for discrete differential dynamic programming, *Chow et al.* [1975] for comparing computer time requirements of several algorithms and *Murray and Yakowitz* [1979] for constrained differential DP, to mention only a few studies. First, the basic system will be presented and then optimized under various cost function and constraint sets. The experiments have been designed so that discontinuities are progressively introduced into the cost-to-go function through tighter constraints and their effects on the proposed optimization algorithm studied. Note that even when the constraints on the control can be completely relaxed, the constraints on the state are needed because they define the feasible space of interpolation.

7.1. *System Description*

Let $u_i(k + 1)$, $x_i(k + 1)$ denote the release and the ending storage of reservoir i, respectively, at the time period $k + 1$. The dynamics of the system are given by the continuity equations:

$$\begin{aligned}
x_1(k+1) &= x_1(k) - u_1(k+1) + q_1(k+1) \\
x_2(k+1) &= x_2(k) - u_2(k+1) + q_2(k+1) \\
x_3(k+1) &= x_3(k) - u_3(k+1) + u_2(k+1) \\
x_4(k+1) &= x_4(k) - u_4(k+1) + u_3(k+1) + u_1(k+1)
\end{aligned} \tag{36}$$

$k = 0, 1, \cdots, N - 1$, where $q_1(k + 1)$ and $q_2(k + 1)$ are the inflows to the system considered either deterministic or stochastic and N is the length of the operating horizon. For recreation and flood control purposes constraints are imposed on the storages:

$$0 \le \mathbf{x}(k) \le \mathbf{K} \qquad k = 0, \cdots, N \tag{37}$$

where $\mathbf{K}$ is the vector of reservoir capacities, with ith element K_i the capacity of the reservoir i. Also, constraints on the releases are imposed on the basis of the capacity of the power generators (it is assumed that there is a power generation

TABLE 1*a*. Deterministic GDP Optimal Control of the System of Example 1

State Discretization N_i	Period k	Optimal Control Vector $\mathbf{u}^*(k)$	Total Cost J^*	Approximate Cost F^*
3	1	(1.50, 2.69, 1.92, 2.46)		
3	2	(1.50, 2.69, 1.93, 2.47)		
3	3	(1.51, 2.49, 1.97, 2.46)	66.95	66.01
4	1	(1.50, 2.64, 1.99, 2.50)		
4	2	(1.50, 2.64, 1.98, 2.50)		
4	3	(1.50, 2.64, 2.00, 2.50)	66.86	66.71
Exact solution for all periods		(1.50, 2.67, 2.01, 2.51)	66.85	66.85

The starting vector is $\mathbf{x}(0) = (6,6,6,6)$. For N_i=3 the nodal state vectors $\mathbf{x}(k)$ for all state variables are formed from the set of states {0,6,12}. For N_i=4 the set of states is {0,4,8,12}. Approximate cost F^* is the cost computed backwards, that is, the value of F_0 ($\mathbf{x}(0)$) evaluated at (6,6,6,6). Total cost J^* is the cost computed from a forward run using the optimal control vector $\mathbf{u}^*(k)$.

station at each reservoir outflow) and the water use downstream. These constraints are

$$\mathbf{u}^{\min}(k) \le \mathbf{u}(k) \le \mathbf{u}^{\max}(k) \qquad k = 1, \cdots, N \tag{38}$$

All quantities are expressed in the same units of storage.

The performance criterion takes into consideration the benefits from power generation and irrigation (short-term optimization) and it also includes a terminal cost to account for longer-term operating policies, e.g., a desire to reach a specific level of storage at the end of the operating horizon. The performance criterion is expressed as

$$J = \sum_{k=0}^{N-1} \sum_{i=1}^{4} A_i(u_i(k)) + \sum_{i=1}^{4} B_i(x_i(N), m_i) \tag{39}$$

where $A_i(\ )$ is the single-stage cost function, $B_i(\ )$ is the terminal cost function, and m_i is the desired level of water in reservoir i at the end of the operating horizon.

Testing of our algorithm has been performed on an optimal control problem for which deterministic and stochastic optimization can be easily obtained with other methods. It is well-known that for a system with linear dynamics, quadratic terminal and stagewise cost functions, and no constraints the cost-to-go function propagates as quadratic (see, for example, *Dreyfus and Law* [1977, chapter 6]). In such a case, the GDP method is expected to be exact even with the minimum number of discretization nodes that is two nodes for each state variable. This assertion was verified for several examples and provided a test of our computer programs. Note that conventional DDP would require an infinite number of nodes to obtain an exact solution. Of course, for this particular case the solution can be efficiently obtained through other methods such as nonlinear optimization, differential dynamic programming or linear quadratic control. Also, note that stochastic systems which satisfy the above requirements and have an additive random term in the dynamics are "certainty equivalent" (see, for example, *Dreyfus and Law* [1977, chapter 14]). This means that the optimal control of the stochastic problem is the same as that of the deterministic problem constructed from the stochastic one by replacing the random variable by its expected value. Thus for such systems testing of stochastic GDP has also been performed.

In most practical situations, however, the cost functions are not quadratic or there are constraints on the control and the state which introduce discontinuities in the cost-to-go function. The performance of the GDP algorithm for the deterministic and stochastic optimization of such systems is studied in the following examples.

7.1.1. *Example 1.* In this example the terminal cost function is quadratic and the control unconstrained. The operation horizon consists of three operating periods, the vector of the capacities of the reservoirs are $\mathbf{K} = (12, 12, 12, 12)$, the vector of the desired terminal states is $\mathbf{m} = (5, 5, 5, 7)$, and the stochastic inflows have a mean vector $\bar{\mathbf{q}} = (2, 4)$ and a vector of standard deviations $\mathbf{s}_q = (s_{q_1}, s_{q_2})$. The inflows are considered independent with a lognormal probability density function. The cost functions in (39) are

$$A_i(u_i(k)) = c_i(k)(u_i(k) - 1)^2$$

$$B_i(x_i(N), m_i) = (x_i(N) - m_i)^2$$

where

$$\mathbf{c} = [c_i(k)] = (1.1, 1.2, 1.0, 1.3) \qquad \forall k \tag{40}$$

For this system, optimization shows that the capacity constraints become active when the reservoirs are almost full at the previous stage. This means that the quadratic terminal

TABLE 1*b*. Same as Table 1a But With Starting Vector $\mathbf{x}(0)=(1,1,1,1)$

State Discretization N_i	Period k	Optimal Control Vector $\mathbf{u}^*(k)$	Total Cost J^*	Approximate Cost F^*
3	1	(0.95, 2.33, 1.20, 0.40)		
3	2	(0.95, 2.33, 1.19, 0.40)		
3	3	(0.95, 2.33, 1.19, 0.40)	10.60	10.61
Exact solution for all periods		(0.95, 2.33, 1.19, 0.40)	10.58	10.58

TABLE 1*c*. Stochastic GDP Optimal Control of the System of Example 1

Standard Deviation of Inflows s_q	Optimal Control Vector $\mathbf{u}^*(1)$	Aproximate Cost F^*
(0.5, 0.5)	(1.47, 2.62, 1.95, 2.43)	66.04
(1.5, 1.5)	(1.48, 3.17, 2.14, 2.60)	80.56

The starting vector is $\mathbf{x}(0)=(6,6,6,6)$. State discretization $N_i=3$. Approximate cost F^* as defined in Table 1*a*.

TABLE 2*b*. Stochastic GDP Optimal Control of the System of Example 2

Standard Deviation of Inflows s_q	Optimal Control Vector $\mathbf{u}^*(1)$	Approximate Cost F^*
(0.5, 0.5)	(1.81, 2.61, 2.05, 2.73)	159.26
(1.5, 1.5)	(2.67, 3.29, 1.63, 2.28)	240.10

The starting vector is $\mathbf{x}(0) = (6,6,6,6)$. State discretization $N_i=3$.

cost function does not propagate as quadratic in a backward optimization scheme. In fact, it is well-known (see, for example, *Bellman and Dreyfus* [1962, chapter 6]) that $F_k(\mathbf{x}(k))$ propagates as a piecewise quadratic function with the break points defined by the constraints. It is shown that in such cases GDP gives good approximations of the optimal control and total cost even for as coarse a discretization as three nodes per state variable.

Table 1*a* reports the results of deterministic optimization for an initial state vector $\mathbf{x}(0) = (6, 6, 6, 6)$ and for two state discretization schemes. These results are compared with the exact solution obtained through nonlinear optimization. Note that for the particular form of the cost functions considered the true optimal control vector remains constant over all operating periods. For comparison purposes Table 1*b* gives the GDP optimal control for an initial vector $\mathbf{x}(0) = (1, 1, 1, 1)$. Notice that these results are exact even for three discretization nodes, since the initial condition of almost empty reservoirs defines an optimal trajectory which does not involve in the interpolation any state vector affected by the constraints. For the stochastic optimization, probabilities $\alpha_j = \beta_j = 0.05, j = 1, \cdots 4$ were used for the deterministic equivalents of the chance constraints in (5*a*) and (5*b*), that is, the acceptance probability of violating any of the probabilistic constraints on the releases was set equal to 5%. The results for an initial vector $\mathbf{x}(0) = (6, 6, 6, 6)$ and for a three-node state discretization scheme are given in Table 1*c*. As was expected, the higher the variability of the inflows the higher the expected value of the cost and the more the stochastic optimal control $\mathbf{u}^*(1)$ deviates from the deterministic one.

7.1.2. *Example 2.* The performance of GDP for the optimal control of systems with cost functions of degree higher than quadratic is now studied. For this example, $\bar{\mathbf{q}} = (2, 3)$ and all other variables remain the same as before. The cost functions are

$$A_i(u_i(k)) = c_i(k)(u_i(k) - 1.)^4$$

$$B_i(x_i(N), m_i) = (x_i(N) - m_i)^4$$

Table 2*a* reports the results of the deterministic optimization of the system for an initial state vector $\mathbf{x}(0) = (6, 6, 6, 6)$ and for a state discretization scheme consisting of three nodes. The exact solution has again been obtained through nonlinear optimization. It is observed that even for such a coarse state discretization scheme ($N_i = 3$) the results of GDP are of reasonable accuracy whereas it is expected that DDP with only three nodes per state variable and fourth-degree cost functions would have very poor performance (see *Kitanidis and Foufoula-Georgiou* [1987] and *Goulter and Tai* [1985]). Of course, finer state discretization schemes ($N_i = 4$ or 5) which are well within the computational capabilities of most computers, would rapidly improve the performance of GDP. Such a comparative study is, however, outside the scope of the present paper.

The stochastic optimization results for lognormally distributed inputs and two vectors of standard deviations are presented in Table 2*b*. As in example 1, the probabilistic constraints on the releases have been converted to their deterministic equivalents using a probability of 5%. As was expected, the stochastic optimal control $\mathbf{u}^*(1)$ and the total approximate cost F^* deviate from their deterministic counterparts, the deviation being larger the larger the variability of the stochastic inflows.

8. CONCLUDING REMARKS

A new computational algorithm for the discrete time optimal control of systems separable in stages (i.e., sequential optimization) has been presented. The method, termed gradient dynamic programming (GDP), is believed to provide a valuable tool for the stochastic optimal control of multidimensional water resources systems. The computational burden of explicit stochastic optimization methods (namely, the conventional stochastic DDP) and the existence of many efficient methods for the deterministic optimization of large systems has many times motivated the deterministic assumption of systems which are clearly stochastic. The practical implications of such "suboptimal" operation rules have many times been emphasized in the literature (see, for example, *Weiner*

TABLE 2*a*. Deterministic GDP Optimal Control of the System of Example 2

State Discretization N_i	Period k	Optimal Control Vector $\mathbf{u}^*(k)$	Total Cost J^*	Approximate Cost F^*
3	1	(1.32, 2.51, 2.05, 2.67)		
3	2	(1.46, 2.51, 2.03, 2.71)		
3	3	(2.08, 2.49, 2.23, 2.81)	151.91	132.54
Exact solution for all periods		(1.67, 2.52, 2.15, 2.81)	154.83	154.83

The starting vector is $\mathbf{x}(0) = (6, 6, 6, 6)$. All other variables have been explained in Table 1*a*.

and Ben-Zvi [1982] and *Stedinger et al.* [1984]). One of the important aspects of stochastic optimization is that the resulting cost is much more reliable which is a highly desired property for planning purposes.

GDP is a general computational algorithm which is believed to provide a valuable tool for the stochastic optimal control of multidimensional systems which are decomposable in stages. In this paper we have presented the general methodology and have provided details of its implementation. For the constrained stagewise optimization part of the algorithm, we have studied and discussed an active set strategy with a Newton-type optimization scheme. Other optimization methods can be used at this step depending on the particular problem at hand. Also, other interpolating functions preserving the values and derivatives of the cost-to-go function at the nodes can be incorporated (see, for example, *Foufoula-Georgiou* [1987]).

One important requirement of all gradient-based optimization methods is the smoothness of the functions to be optimized. Since it is known that constraints on the state and control vectors introduce discontinuities in the derivatives of the cost-to-go functions care must be exercised in using GDP for problems which are constraint-dominated. Finer state discretization schemes may be required in those cases. Fortunately, stochastic optimization requires integration of the cost-to-go function over the range defined by the pdf of the stochastic input and this results in a "smoothing" of the function to be optimized. Our computational experience suggests that this tends to alleviate the effects of the discontinuities.

GDP has not solved the problem of stochastic optimization. It is believed, however, that it is a first step toward exploring methods which combine the advantages of the general backward moving stagewise optimization with the wealth of constrained optimization methods through the idea of local approximations of the cost-to-go functions. It should be kept in mind that no single optimization method is "best" for all problems. Further theoretical and computational work should focus on a comparison of the available methods and on identifying the classes of problems (in terms of type and smoothness of the cost-to-go function, presence of constraints, etc.) for which a particular approximation method is more attractive than others in terms of accuracy and computational efficiency.

APPENDIX A: APPROXIMATION OF F THROUGH HERMITE INTERPOLATION

Given the values of F and its derivatives $\partial F/\partial x_j$, $j = 1, \cdots n$, at all nodes, one may approximate the function F within a hypercube by

$$F(x_1, \cdots, x_n) = \sum_{i=1}^{2^n} F_i \phi_i(x_1, \cdots, x_n) + \sum_{i=1}^{2^n} \sum_{j=1}^{n} \left(\frac{\partial F}{\partial x_j}\right)_i \psi_{ij}(x_1, \cdots, x_n) \tag{A1}$$

where i is the index of nodes ($i = 1, \cdots, 2^n$) defining a hypercube, j is the index of direction or the state variable x_j ($j = 1, \cdots, n$), F_i is the value of F at node i, and $(\partial F/\partial x_j)_i$ is the value of the derivative of F with respect to x_j at node i.

The $\phi_i(x_1, \cdots, x_n)$ is a function of the state-space coordinates with the following properties: (1) it is equal to 1 at node i, (2) it vanishes at all other nodes, and (3) its first derivative in any direction is zero at all nodes.

The $\psi_{ij}(x_1, \cdots, x_n)$ has the following properties: (1) it vanishes on all nodes, (2) its derivatives in all directions except for j vanish on all nodes, and (3) its derivative in direction j is equal to 1 on node i and vanishes on all other nodes.

Let $\mathbf{x}^b = (x_1^b, \cdots, x_n^b)^T$ and $\mathbf{x}^a = (x_1^a, \cdots, x_n^a)^T$ be the vectors of coordinates just "below" and "above" the interpolation point $\mathbf{x}_i = (x_1, \cdots, x_n)^T$ in the sense that $x_i^b \le x_i \le x_i^a$ for $i = 1, \cdots, n$. Define $\Delta x_i = x_i^a - x_i^b$ and the local coordinates $\xi_i = (x_i - x_i^b)/\Delta x_i$, $i = 1, \cdots, n$.

The kth coordinate at a given node may be represented as $x_k^b + l_k \Delta x_k$, where l_k is either 0 or 1. That is, to each node i we associate an n tuple of binary zero-one numbers $(l_1, \cdots, l_n)$. Then the lowest-order polynomial ϕ_i is

$$\phi_i(\xi_1, \cdots, \xi_n) = \left(1 + \sum_{k=1}^{n} \eta_k - 2 \sum_{k=1}^{n} \eta_k^2\right) \prod_{k=1}^{n} (1 - \eta_k) \tag{A2}$$

and the lowest-order polynomial ψ_{ij} is

$$\psi_{ij}(\xi_1, \cdots, \xi_n) = \eta_j(1 - \eta_j) \prod_{k=1}^{n} (1 - \eta_k) \tag{A3}$$

$$\text{where } \eta_k = \xi_k \qquad l_k = 0$$
$$\eta_k = (1 - \xi_k) \qquad l_k = 1 \tag{A4}$$

The proof can be found in the work by *Kitanidis* [1986].

APPENDIX B: APPROXIMATION OF $\Delta_x F_k$ AND $F_{k,xx}$ BY DIFFERENTIATION OF THE POLYNOMIALS OF THE HERMITIAN INTERPOLATION

Differentiating F from (A1) with respect to the state gives

$$\frac{dF}{dx_s} = \sum_{i=1}^{2^n} F_i \left(\frac{d\phi_i}{dx_s}\right) + \sum_{i=1}^{2^n} \sum_{j=1}^{n} \left(\frac{dF}{dx_j}\right)_i \left(\frac{d\psi_{ij}}{dx_s}\right) \tag{B1}$$

We define the following quantities for notational convenience:

$$P = \prod_{k=1}^{n} (1 - \eta_k) \tag{B2}$$

$$P_{(a,b)} = \prod_{\substack{k=1 \\ k \neq a,b}}^{n} (1 - \eta_k) \tag{B3}$$

$$R = 1 + \sum_{k=1}^{n} \eta_k - 2 \sum_{k=1}^{n} \eta_k^2 \tag{B4}$$

Then in (B1)

$$\frac{d\phi_i}{dx_s} = \frac{d\eta_s}{dx_s} [(1 - 4\eta_s)P - RP_{(s)}] \tag{B5}$$

$$\frac{d\psi_{ij}}{dx_s} = \left(\frac{d\eta_s}{dx_s}\right)\left(\frac{d\eta_j}{dx_j}\right)^{-1} TP_{(s)} \tag{B6}$$

where

$$T = (1 - 3\eta_j)^2 \qquad \text{if } s = j$$
$$T = \eta_j(\eta_j - 1) \qquad \text{if } s \neq j \tag{B7}$$

Similarly,

$$\frac{d^2 F}{dx_l\, dx_s} = \sum_{i=1}^{2^n} F_i \left(\frac{d^2\phi_i}{dx_l\, dx_s}\right) + \sum_{i=1}^{2^n} \sum_{j=1}^{n} \left(\frac{dF}{dx_j}\right)_i \left(\frac{d^2\psi_{ij}}{dx_l\, dx_s}\right) \tag{B8}$$

where

$$\frac{d^2\phi_i}{dx_l\, dx_s} = \left(\frac{d\eta_l}{dx_l}\right)\left(\frac{d\eta_s}{dx_s}\right) S_{l,s} \tag{B9}$$

with

$$S_{s,s} = -4P - 2(1 - 4\eta_s)P_{(s)} \qquad \forall\, s \tag{B10}$$

$$S_{l,s} = -(1-4\eta_s)P_{(l)} - (1-4\eta_l)P_{(s)} + RP_{(l,s)} \qquad \forall\, l \neq s \tag{B11}$$

Also,

$$\frac{d^2\psi_{ij}}{dx_l\, dx_s} = \left(\frac{d\eta_l}{dx_l}\right)\left(\frac{d\eta_s}{dx_s}\right)\left(\frac{d\eta_j}{dx_j}\right)^{-1} W_{l,s,j} \tag{B12}$$

where

$$W_{s,s,j} = -3P - (1 - 3\eta_s)P_{(s)} \qquad \forall\, s = j \tag{B13a}$$

$$W_{s,s,j} = 0 \qquad \forall\, s \neq j \tag{B13b}$$

$$W_{l,s,j} = -(1 - 3\eta_s)P_{(l)} \qquad \forall\, s = j,\ l \neq s \tag{B14a}$$

$$W_{l,s,j} = -(1 - 3\eta_l)P_{(s)} \qquad \forall\, l = j,\ l \neq s \tag{B14b}$$

$$W_{l,s,j} = \eta_j(1 - \eta_j)P_{(l,s)} \qquad \forall\, s \neq j,\ l \neq s \tag{B14c}$$

Appendix C: Projection of a Solution to the Hyperplanes of the Active Constraints

Let $\mathbf{u}_0$ denote a solution vector which violates some of the constraints of the working set $G\mathbf{u} = \mathbf{d}$, where G is an $(p \times n)$ matrix of rank $p(p < m)$. The problem is to find the projection of $\mathbf{u}_0$ onto the domain defined by the active constraints, so that an initial feasible solution is obtained each time the active set changes. Let $\mathbf{u}'$ denote the projection, which can be written as

$$\mathbf{u}' = \mathbf{u}_0 + \delta\mathbf{u} \tag{C1}$$

The problem then becomes one of constrained optimization:

$$\begin{gathered} \min\ \delta\mathbf{u}^T\delta\mathbf{u} \\ \text{subject to } G(\mathbf{u}_0 + \delta\mathbf{u}) = \mathbf{d} \end{gathered} \tag{C2}$$

By forming the Lagrangian

$$\delta\mathbf{u}^T\delta\mathbf{u} + \mathbf{v}^T[G(\mathbf{u}_0 + \delta\mathbf{u}) - \mathbf{d}] \tag{C3}$$

where $\mathbf{v}$ is a $p \times 1$ vector of Lagrange multipliers, and taking derivatives with respect to $\delta\mathbf{u}$ and $\mathbf{v}^T$, we obtain

$$\begin{gathered} \delta\mathbf{u} + G^T\mathbf{v} = 0 \\ G(\mathbf{u}_0 + \delta\mathbf{u}) = \mathbf{d} \end{gathered} \tag{C4}$$

Thus $\delta\mathbf{u}$ is computed from the solution of the system of $(n + p)$ equations

$$\begin{bmatrix} I & G^T \\ G & 0 \end{bmatrix}\begin{bmatrix} \delta\mathbf{u} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{d} - G\mathbf{u}_0 \end{bmatrix} \tag{C5}$$

or

$$\delta\mathbf{u} = G^T(GG^T)^{-1}(\mathbf{d} - G\mathbf{u}_0) \tag{C6}$$

Appendix D: Solution of the Constrained Optimization Problem

Consider the problem

$$\min f(\mathbf{u}) \tag{D1}$$

$$\text{subject to } G\mathbf{u} = \mathbf{d} \tag{D2}$$

where G is a $(p \times m)$ matrix of rank p, $\mathbf{d}$ is a $p \times 1$ vector, and $\mathbf{u}$ an m-dimensional vector. The iterations start with a feasible vector $\mathbf{u}_0$. The solution at the next step is $\mathbf{u} = \mathbf{u}_0 + \Delta\mathbf{u}$, where $\Delta\mathbf{u}$ denotes an "improvement" to be determined. Since both $\mathbf{u}_0$ and $\mathbf{u}$ satisfy the equality constraints, $\Delta\mathbf{u}$ satisfies

$$G\Delta\mathbf{u} = 0 \tag{D3}$$

As in every Newton-type optimization method it is assumed that f can be approximated locally by a quadratic function about the last estimate of the optimal solution, $\mathbf{u}_0$. Through expansion into Taylor series, the following programming problem is obtained:

$$\begin{gathered} \min f(\Delta\mathbf{u}) = f + \mathbf{g}^T\Delta\mathbf{u} + \tfrac{1}{2}\Delta\mathbf{u}^TQ\Delta\mathbf{u} \\ \text{subject to } G\Delta\mathbf{u} = 0 \end{gathered} \tag{D4}$$

where f, its gradient $\mathbf{g} = (\nabla_u f)^T$, and its Hessian (matrix of second derivatives) $Q = f_{uu}$ are all calculated at $\mathbf{u}_0$.

The Lagrangian associated with the constrained optimization problem is

$$L = f(\Delta\mathbf{u}) + \boldsymbol{\lambda}^TG\Delta\mathbf{u} \tag{D5}$$

where $\boldsymbol{\lambda}^T$ is a $p \times 1$ vector of Lagrange multipliers. Taking derivatives of L with respect to $\Delta\mathbf{u}$

$$\mathbf{g} + Q\Delta\mathbf{u} + G^T\boldsymbol{\lambda} = 0 \tag{D6}$$

where combined with (D3) yields the linear system of $m + p$ equations

$$\begin{bmatrix} Q & G^T \\ G & 0 \end{bmatrix}\begin{bmatrix} \Delta\mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{g} \\ 0 \end{bmatrix} \tag{D7}$$

from which both $\Delta\mathbf{u}$ and $\boldsymbol{\lambda}$ may be obtained. The coefficient matrix is nonsingular and the quadratic optimization problem has a unique local minimum if Q is positive definite in the subspace $M = \{\Delta\mathbf{u}\colon G\Delta\mathbf{u} = 0\}$. That is, for any $\Delta\mathbf{u}$ which satisfies (D3), $\Delta\mathbf{u}^TQ\Delta\mathbf{u} > 0$. A necessary condition is that Q has at least $n - p$ positive eigenvalues. Note that this is a weaker condition than positive definite Hessian.

The step may be written in the familiar form

$$\Delta\mathbf{u} = -R \cdot \mathbf{g} \tag{D8}$$

where R may be interpreted as the inverse of the Hessian in the subspace where the decision variables are permitted to vary. That is, RQ is a projection matrix on the subspace of feasible steps $\Delta\mathbf{u}$. In the special case that Q is invertible, R may be explicitly written as

$$R = Q^{-1} - Q^{-1}G^T(GQ^{-1}G^T)^{-1}GQ^{-1} \tag{D9}$$

Appendix E: Discretization of the PDF of the Stochastic Input and Numerical Integration

For normal or lognormal pdf of the stochastic inputs efficient numerical integration schemes can be utilized, as is illustrated below. For convenience in notation we consider the numerical integration of a one-dimensional function $h(x)$. *Abramowitz and Stegun* [1972, ch. 25] gives

$$\int_{-\infty}^{+\infty} e^{-x^2}h(x)\, dx = \sum_{i=1}^{n} a_ih(x_i) + R_n \tag{E1}$$

where x_i is the ith zero of the Hermite polynomial $H_n(x)$ and a_i are weights given as

$$a_i = \frac{2^{n-1}n!\sqrt{\pi}}{n^2[H_{n-1}(x_i)]^2} \tag{E2}$$

Both a_i and x_i are given in tables in the work by *Abramowitz and Stegun* [1972, Table 25. 10]. The remainder term is

$$R_n = \frac{n!\sqrt{\pi}}{2^n(2n)!} h^{(2n)}(\xi) \qquad -\infty < \xi < \infty \tag{E3}$$

Hence for a standard normal distribution the two- and three-point Hermite integration becomes

$$\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} h(x)\,dx = \tfrac{1}{2}[h(-1) + h(1)] + R_2' \tag{E4}$$

$$\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} h(x)\,dx$$

$$= \tfrac{2}{3}h(0) + \tfrac{1}{6}[h(-1.732) + h(1.732)] + R_3' \tag{E5}$$

where

$$R_2' = \frac{1}{48} h^{(4)}(\xi) \qquad -\infty < \xi < \infty \tag{E6}$$

$$R_3' = \frac{1}{960} h^{(6)}(\xi) \qquad -\infty < \xi < \infty \tag{E7}$$

For example, one can see that the numerical integration of a normal or lognormal pdf on a smooth cost function of requisite differentiability and degree three (or five) is exact with only a two- (or three-) point discretization scheme.

NOTATION

N	number of decision periods (stages).
η	dimension of state vector.
m	dimension of control vector.
r	dimension of input vector.
k	stage index.
$\mathbf{x}(k)$	$(n \times 1)$ state vector at the beginning of stage k.
$\mathbf{u}(k)$	$(m \times 1)$ control vector during stage k.
$\mathbf{w}(k)$	$(r \times 1)$ input vector during stage k.
$\mathbf{T}_k = \mathbf{T}_k(\mathbf{x}(k), \mathbf{u}(k+1), \mathbf{w}(k+1))$	n-dimensional vector function.
$\partial\mathbf{T}_k/\partial\mathbf{x} := \partial\mathbf{T}_k/\partial\mathbf{x}(k)$	Jacobian of $\mathbf{T}_k$ with respect to the state vector.
$\partial\mathbf{T}_k/\partial\mathbf{u} := \partial\mathbf{T}_k/\partial\mathbf{u}(k+1)$	Jacobian of $\mathbf{T}_k$ with respect to the control vector.
$C_k := C_k(\mathbf{x}(k), \mathbf{u}(k+1))$	single-stage cost function.
$\nabla_u C_k := \partial C_k/\partial\mathbf{u}(k+1)$	gradient of C_k with respect to the control vector.
$\nabla_x C_k := \partial C_k/\partial\mathbf{x}(k)$	gradient of C_k with respect to the state vector.
$C_{k,uu}$	Hessian of C_k with respect to the control vector $\mathbf{u}(k+1)$.
$F_k := F_k(\mathbf{x}(k)) =$	cost to go at stage k.
$\nabla_x F_k := dF_k/d\mathbf{x}(k)$	gradient of F_k with respect to the state vector.
$\nabla_u F_k := dF_k/d\mathbf{u}(k)$	gradient of F_k with respect to the control vector.
$F_{k,xx}$	Hessian of F_k with respect to the state vector $\mathbf{x}(k)$.
$\mathbf{u}^*(k)$	optimal control associated with α state vector $\mathbf{x}(k-1)$.
$d\mathbf{u}^*/d\mathbf{x}$	the Jacobian of $\mathbf{u}_k^*(\mathbf{x}(k-1))$ with respect to the state vector $\mathbf{x}(k-1)$.

Acknowledgments. The material is based upon work supported by National Science Foundation grants CEE-8212066 and CEE-8420664. Most of the computer work was performed on a NAS-9160 computer at Iowa State University (ISU) with funds provided by the ISU Computation Center. The thoughtful comments of the referees are gratefully acknowledged.

REFERENCES

Abramowitz, M., and I. A. Stegun (Eds.), *Handbook of Mathematical Functions*, Dover, Mineola, N. Y., 1972.

Bellman, R. E., *Dynamic Programming*, Princeton University Press, Princeton, N.J., 1957.

Bellman, R. E., and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J., 1962.

Birnbaum, I., and L. Lapidus, Studies in approximation methods, I, Splines and control via discrete dynamic programming, *Chem. Eng. Sci.*, *33*, 415–426, 1978.

Buras, N., *Scientific Allocation of Water Resources*, Elsevier Science, New York, 1972.

Chang, S., and W. Yeh, Optimal allocation of artificial aeration along a polluted system using dynamic programming, *Water Resour. Bull.*, *9*(5), 985–997, 1973.

Chow, V. T., D. R. Maidment, and G. W. Tauxe, Computer time and memory requirements for DP and DDDP in water resources systems analysis, *Water Resour. Res.*, *11*(5), 621–628, 1975.

Chu, W. S., and W. W-G Yeh, A non-linear programming algorithm for real-time hourly reservoir operations, *Water Resour. Bull.*, *14*, 1048–1063, 1978.

Daniel, J. W., Splines and efficiency in dynamic programming, *J. Math. Anal. Appl.*, *54*, 402–407, 1976.

Dracup, J., and T. Fogarty, Optimal planning for a thermal discharge treatment system, *Water Resour. Res.*, *10*(1), 69–71, 1974.

Engels, H., *Numerical Quadrature and Cubature*, Academic, Orlando, Fla., 1980.

Fletcher, R., *Practical Methods of Optimization*, vol. 2, *Constrained Optimization*, John Wiley, New York, 1981.

Foufoula-Georgiou, E., Convex interpolation for gradient dynamic programming, technical report, Dep. of Civ. Eng., Iowa State Univ., Ames, 1987.

Gagnon, C. R., R. H. Hicks, S. L. S. Jacoby, and J. S. Kowalik, A non-linear programming approach to a very large hydroelectric system optimization, *Math. Progr.*, *6*, 28–41, 1974.

Gal, S., Optimal management of a multireservoir water supply system, *Water Resour. Res.*, *15*(4), 737–749, 1979.

Georgakakos, A. P., and D. H. Marks, Real-time control of reservoir systems, *Tech. Rep. 301*, Ralph M. Parsons Lab. for Hydrol. and Water Resour., Mass. Inst. of Technol., Cambridge, May 1985.

Gill, P. E., and W. Murray, *Numerical Methods for Constrained Optimization*, Academic, Orlando, Fla., 1974.

Goulter, J. C., and F.-K. Tai, Practical implications in the use of stochastic dynamic programming for reservoir operation, *Water Resour. Bull.*, *21*(1), 65–74, 1985.

Heidari, M., V. T. Chow, P. V. Kokotovic, and D. D. Meredith, Discrete differential dynamic programming approach to water resources systems optimization, *Water Resour. Res.*, *7*(2), 273–282, 1971.

Jacobson, D., and D. Mayne, *Differential Dynamic Programming*, Academic, Orlando, Fla., 1970.

Kitanidis, P. K., Hermite interpolation on an n-dimensional rectangular grid, water resources technical note, St. Anthony Falls Hydraul. Lab., Univ. of Minn., Minneapolis, July 1986.

Kitanidis, P. K., A first-order approximation to stochastic optimal control of reservoirs, *Stoc. Hydrol. Hydraul.*, *1*(3), 169–184, 1987.

Kitanidis, P. K., and E. Foufoula-Georgiou, Error analysis of conventional discrete and gradient dynamic programming, *Water Resour. Res.*, *23*(5), 845–856, 1987.

Labadie, J. W., D. M. Morrow, and Y. H. Chen, Optimal control of unsteady combined sewer flow, *J. Water Resour. Manage. Plann. Am. Soc. Civ. Eng.*, *106*(WR1), 205–223, 1980.

Larson, R., *State Increment Dynamic Programming*, Elsevier Science, New York, 1968.

Larson, R. E., and J. L. Casti, *Principles of Dynamic Programming, Advanced Theory and Applications*, vol. 7, *Control and Systems Theory*, Dekker, New York, 1982.

Lee, E. S., and S. Waziruddin, Applying gradient projection and conjugate gradient to the optimum operation of reservoirs, *Water Resour. Bull.*, *6*(5), 713–724, 1970.

Lenard, M., A computational study of active set strategies in nonlinear programming with linear constraints, *Tech. Rep. 1564*, Math Res. Cent., Univ. of Wis., Madison, 1975.

Luenberger, *Introduction to Linear and Nonlinear Programming*, 2nd ed., Addison Wesley, Reading, Mass., 1984.

Mays, L., and H. Wenzel, Optimal design of multilevel branching sewer systems, *Water Resour. Res.*, *12*(5), 913–917, 1976.

Murray, D., and S. Yakowitz, Constrained differential dynamic programming and its application to multireservoir control, *Water Resour. Res.*, *15*(4), 1017–1027, 1979.

Papageorgiou, M., Optimal multireservoir network control by the discrete maximum principle, *Water Resour. Res.*, *21*(12), 1824–1830, 1985.

Schultz, M. H., *Spline Analysis*, 156 pp., Prentice-Hall, Englewood Cliffs, N.J., 1973.

Stedinger, J. R., B. F. Sula, and D. P. Loucks, Stochastic dynamic programming models for reservoir operation optimization, *Water Resour. Res.*, *20*(11), 1499–1505, 1984.

Weiner, D., and A. Ben-Zvi, A stochastic dynamic programming model for the operation of the Mediterranean-Dead Sea project, *Water Resour. Res.*, *18*(4), 729–734, 1982.

Yakowitz, S., Dynamic programming applications in water resources, *Water Resour. Res.*, *18*(4), 673–696, 1982.

Yeh, W. W-G., Reservoir management and operations models: A state-of-the-art review, *Water Resour. Res.*, *21*(2), 1797–1818, 1985.

E. Foufoula-Georgiou, Department of Civil Engineering, Iowa State University, Ames, IA 50011.

P. K. Kitanidis, Department of Civil Engineering, Stanford University, Stanford, CA 94305.

(Received June 1, 1987;
revised April 25, 1988;
accepted April 25, 1988.)