Techniques in Computational Stochastic Dynamic Programming[1,2]

Floyd B. Hanson[3]

University of Illinois at Chicago
Chicago, Illinois 60607-7045

Excerpted Preprint Section

A. DIFFERENTIAL DYNAMIC PROGRAMMING

Differential dynamic programming (DDP) is a variant of dynamic programming in which a quadratic approximation of the cost about a nominal state and control plays an essential role. The method uses successive approximations and expansions in differentials or increments to obtain a solution of optimal control problems. The DDP method is due to Mayne [11, 8]. DDP is primarily used in deterministic problems in discrete time, although there are many variations. Mayne [11] in his original paper did give a straight-forward extension to continuous time problems, while Jacobson and Mayne [8] present several stochastic variations. The mathematical basis for DDP is given by Mayne in [12], along the relations between dynamic programming and the Hamiltonian formulation of the maximum principle. A concise, computationally oriented survey of DDP developments is given by Yakowitz [16] in an earlier volume of this series and the outline for deterministic control problems in discrete time here is roughly based on that chapter. Earlier, Yakowitz [15] surveys the use of dynamic programming in water resources applications, nicely placing DDP in the larger perspective of other dynamic programming variants. Also, Jones, Willis and Yeh [9], and Yakowitz and Rutherford [17] present brief helpful summaries with particular emphasis on the computational aspects of DDP.

1. Dynamic Programming in Discrete Time

Let κ be the discrete forward time corresponding to $t_\kappa = \kappa \cdot \Delta t$ with initial time $t_0 = 0$ or $\kappa = 0$ and final time $t_K = t_f = K \cdot \Delta t$ or $\kappa = K$ so the stages go from $\kappa = 0$ to K in steps of 1 (in opposite direction to the backward time $T_k = k \cdot \Delta T$ of SDP). Let $\mathbf{x}_\kappa = [\mathbf{x}_{i,\kappa}]_{n\times 1}$ be the n-dimensional state vector and $\mathbf{u}_\kappa = [\mathbf{u}_{i,\kappa}]_{m\times 1}$ be the m-dimensional control vector. The discrete time dynamics along the state trajectory is given recursively by

$$\mathbf{x}_{\kappa+1} = \mathbf{F}_\kappa(\mathbf{x}_\kappa, \mathbf{u}_\kappa), \text{ for } 0 \le \kappa \le K-1, \tag{1}$$

where discrete plant function $\mathbf{F}_\kappa$ is at least twice continuously differentiable in both state and control vectors.

The total cost of the trajectory from $\kappa = 0$ to K is

$$v[\mathbf{x}, \mathbf{u}; 0, K] = \sum_{\kappa=0}^{K-1} G_\kappa(\mathbf{x}_\kappa, \mathbf{u}_\kappa) + Z_K(\mathbf{x}_K), \tag{2}$$

[1] This is a preprint from a chapter that appeared in F. B. Hanson, "Techniques in Computational Stochastic Dynamic Programming" in *Stochastic Digital Control System Techniques,* within series *Control and Dynamic Systems: Advances in Theory and Applications,* vol. 76, (C. T. Leondes, Editor), Academic Press, New York, NY, pp. 103-162, April 1996.

[2] This section on Differential Dynamic Programing appeared in the chapter: F. B. Hanson, "Techniques in Computational Stochastic Dynamic Programming" in *Stochastic Digital Control System Techniques,* within series *Control and Dynamic Systems: Advances in Theory and Applications,* vol. 76, (C. T. Leondes, Editor), Academic Press, New York, NY, pp. 103-162, April 1996.

[3] Supported in part by National Science Foundation Grant DMS 93-0117, National Center for Supercomputing Applications, Pittsburgh Supercomputing Center, and Los Alamos National Laboratory's Advanced Computing Laboratory; written while on sabbatical in Division of Applied Mathematics at Brown University, Providence, RI 02912

where $G_\kappa(\mathbf{x},\mathbf{u})$ is the discrete-time cost function, assumed to be a least twice continuously differentiable in the state and control vectors. Implicit in (2) is that the cost is separable with regard to the stages κ. The final or salvage cost is denoted by $Z_K(\mathbf{x}_K)$, assumed at least twice continuously differentiable.

The ultimate goal is to seek the minimum total cost

$$v_0^*(\mathbf{x}_\kappa) = \min_{\{\mathbf{u}_0,\cdots,\mathbf{u}_{K-1}\}} [v[\mathbf{x},\mathbf{u};0,K]]\,, \tag{3}$$

subject to the dynamic rule (1). However, for enabling dynamic programming analysis, the *variable* time-to-go cost

$$v[\mathbf{x},\mathbf{u};\kappa,K] = \sum_{i=\kappa}^{K-1} G_i(\mathbf{x}_i,\mathbf{u}_i) + Z_K(\mathbf{x}_K), \tag{4}$$

and its minimization,

$$v_\kappa^*(\mathbf{x}_\kappa) = \min_{\{\mathbf{u}_\kappa,\cdots,\mathbf{u}_{K-1}\}} [v[\mathbf{x},\mathbf{u};\kappa,K]]\,, \tag{5}$$

is considered, subject to the final cost condition

$$v_K^*(\mathbf{x}) = Z_K(\mathbf{x}), \tag{6}$$

consistent with the definition that $\sum_{i=K}^{K-1} a_i \equiv 0$ and with salvage costs assumed in (2).

Applying to the *dynamic programming principle of optimality*,

$$v_\kappa^*(\mathbf{x}_\kappa) = \min_{\mathbf{u}_\kappa}\left[G_\kappa(\mathbf{x}_\kappa,\mathbf{u}_\kappa) + \min_{\{\mathbf{u}_{\kappa+1},\cdots,\mathbf{u}_{K-1}\}} v[\mathbf{x},\mathbf{u};\kappa+1,K]\right],$$

decomposing the optimization into that of the current step plus that for the rest of the *cost-to-go*. Using the definition of time-to-go optimal cost (5) and substituting for $\mathbf{x}_{\kappa+1}$ from the recursive dynamic rule (1),

$$v_\kappa^*(\mathbf{x}_\kappa,\kappa) = \min_{\mathbf{u}_\kappa}\left[G_\kappa(\mathbf{x}_\kappa,\mathbf{u}_\kappa) + v_{\kappa+1}^*(\mathbf{F}_\kappa(\mathbf{x}_\kappa,\mathbf{u}_\kappa))\right]. \tag{7}$$

The calculation of this minimum simultaneously produces the optimal control,

$$u_\kappa^*(\mathbf{x}_\kappa) = \arg\min_{\mathbf{u}_\kappa}\left[G_\kappa(\mathbf{x}_\kappa,\mathbf{u}_\kappa) + v_{\kappa+1}^*(\mathbf{F}_\kappa(\mathbf{x}_\kappa,\mathbf{u}_\kappa))\right], \tag{8}$$

as the argument of the minimization, for $\kappa = K-1$ to 0 in backward steps (i.e., (-1) steps). The equations (7,8) comprise the *DP recursive backward sweep*. Here, the term *sweep* is used to indicate an iteration over all time steps, reserving the word *step* for either time or state steps. The use of the term sweep is not to be confused with the use in the related *Successive Sweep Method* as in [5].

The *DP recursive forward sweep* uses the optimal control u_κ^* found in the backward sweep in a forward recursion of the dynamical equation (1),

$$\mathbf{x}_{\kappa+1}^* = \mathbf{F}_\kappa(\mathbf{x}_\kappa^*,\mathbf{u}_\kappa^*(\mathbf{x}_\kappa^*)), \tag{9}$$

starting from the initial state $\mathbf{x}_0^* = \mathbf{x}_0$ and calculation future optimal states for $\kappa = 0$ to $K-1$ in forward steps of 1 up to the final state $\mathbf{x}_K^*$.

While the backward and forward recursions of dynamic programming seem to give a method for computing a solution to the discrete time control problem, they do not give any actual computational method for calculation the minimum or the optimal trajectory motion that would be needed for computational implementation. The implementation is especially unclear if the problem is nonlinear (this is true also of the continuous time case). In order to make the actual computation well-posed, a quadratic approximation cost at each DDP stage k is applied.

2. Final Time DDP Backward Sweep

Each DDP iterate starts out with a current, approximate iterate c for the state-control set $\{\mathbf{x}_\kappa^c, \mathbf{u}_\kappa^c\}$ of near final time K pairs of state and control vectors, satisfying the discrete dynamics

$$\mathbf{x}_{\kappa+1}^c = \mathbf{F}_\kappa(\mathbf{x}_\kappa^c, \mathbf{u}_\kappa^c), \tag{10}$$

for $\kappa = 0$ to $K-1$. Since dynamic programming takes backward steps from the final time, the starting iterate is really the *final* time iterate. It is assumed that these current, nominal iterates are somewhat close to the target optimal trajectory to justify Taylor approximations. The iterations proceed until the trajectories of successive iterates are sufficiently close.

For the final time-to-go cost (the starting step for each backward DDP sweep), a Taylor approximation about the current iterate state-control set $\{\mathbf{x}_K^c, \mathbf{u}_K^c\}$ up to quadratic terms leads to the approximate formula

$$\begin{aligned} v[\mathbf{x},\mathbf{u};K,K] &= G_K(\mathbf{x}_K,\mathbf{u}_K) \equiv Z_K(\mathbf{x}_K) \\ \simeq \widetilde{v}_K(\mathbf{x}_K,\mathbf{u}_K) &\equiv G_K^c + \nabla_x^T[G_K]\cdot\delta\mathbf{x}_K + \nabla_u^T[G_K]\cdot\delta\mathbf{u}_K \\ &+ \frac{1}{2}\delta\mathbf{x}_K^T\cdot\nabla_x\nabla_x^T[G_K]\cdot\delta\mathbf{x}_K + \delta\mathbf{u}_K^T\cdot\nabla_u\nabla_x^T[G_K]\cdot\delta\mathbf{x}_K \\ &+ \frac{1}{2}\delta\mathbf{u}_K^T\cdot\nabla_u\nabla_u^T[G_K]\cdot\delta\mathbf{u}_K, \end{aligned} \tag{11}$$

with increments $\delta\mathbf{x}_K = \mathbf{x}_K - \mathbf{x}_K^c$ and $\delta\mathbf{u}_K = \mathbf{u}_K - \mathbf{u}_K^c$. The following array Taylor coefficients are redefined for computational storage as

$$\begin{aligned} G_K^c &\equiv G_K = G_K(\mathbf{x}_K^c,\mathbf{u}_K^c) \\ (\mathbf{E}_K^c)^T &\equiv \nabla_x^T[G_K] = \nabla_x^T[G_K](\mathbf{x}_K^c,\mathbf{u}_K^c) \\ (\mathbf{D}_K^c)^T &\equiv \nabla_u^T[G_K] = \nabla_u^T[G_K](\mathbf{x}_K^c,\mathbf{u}_K^c) \\ A_K^c &\equiv \frac{1}{2}\nabla_x\nabla_x^T[G_K] = \frac{1}{2}\nabla_x\nabla_x^T[G_K](\mathbf{x}_K^c,\mathbf{u}_K^c) \\ B_K^c &\equiv \nabla_u\nabla_x^T[G_K] = \nabla_u\nabla_x^T[G_K](\mathbf{x}_K^c,\mathbf{u}_K^c) \\ C_K^c &\equiv \frac{1}{2}\nabla_u\nabla_u^T[G_K] = \frac{1}{2}\nabla_u\nabla_u^T[G_K](\mathbf{x}_K^c,\mathbf{u}_K^c), \end{aligned} \tag{12}$$

where the notation A^T denotes the transpose of array A, ∇_x is the gradient in the state and ∇_u is the gradient in the control. In this section, gradients are evaluated at the current state-control set for the time considered. It is assumed that the set $\{\mathbf{x}_K, \mathbf{u}_K\}$ are variable points on a trajectory nearby to the current set $\{\mathbf{x}_K^c, \mathbf{u}_K^c\}$. However, we will not need the $\mathbf{x}_K$ state values themselves except here for the purposes of analysis, but only the coefficients defining the functional form of the quadratic approximation (note the analogous properties to that of the classical LQ control problem).

In absence of constraints, the improved approximation to the optimal control is then the current approximation to the optimal control is obtained from the critical points of $\widetilde{v}$ at $\kappa = K$,

$$\begin{aligned} \nabla_u[\widetilde{v}_K](\mathbf{x}_K,\widetilde{\mathbf{u}}_K^*) &= \mathbf{E}_K^c + B_K^c\cdot\delta\mathbf{x}_K + 2\cdot C_K^c\cdot\delta\widetilde{\mathbf{u}}_K^* \\ &= \nabla_u^T[G_K] + \nabla_u\nabla_x^T[G_K]\cdot\delta\mathbf{x}_K + \nabla_u\nabla_u^T[G_K]\cdot\delta\widetilde{\mathbf{u}}_K^* \\ &= 0, \end{aligned} \tag{13}$$

so that

$$\begin{aligned} &\delta\widetilde{\mathbf{u}}_K^* \\ &= \widetilde{\mathbf{u}}_K^*(\mathbf{x}_K) - \mathbf{u}_K^c \\ &= -\frac{1}{2}C_K^{-c}\cdot(\mathbf{E}_K^c + B_K^c\cdot\delta\mathbf{x}_K) \\ &= -(\nabla_u\nabla_u^T[G_K])^{-1}\cdot\left(\nabla_u[G_K] + \nabla_u\nabla_x^T[G_K]\cdot\delta\mathbf{x}_K\right) \\ &\equiv \boldsymbol{\alpha}_K^c + \beta_K^c\cdot\delta\mathbf{x}_K. \end{aligned} \tag{14}$$

This a linear control law in the state perturbation $\delta\mathbf{x}_K$ with fixed coefficient $\boldsymbol{\alpha}_K^c$ and gain coefficient β_K^c, provided that the inverse $C_K^{-c} = (C_K^c)^{-1}$ is well defined (e.g., $C_K^c = 2\nabla_u\nabla_u^T[G_K]$ is positive definite and hence nonsingular; however, positive definiteness is only needed near the optimal trajectory). Since the expansion in Taylor approximations

with only a few terms is good only if both $\delta\mathbf{x}_K$ and $\delta\mathbf{u}_K$ are small, a problem in consistency arises if the fixed part of $\delta\mathbf{u}_K$, $\boldsymbol{\alpha}_K^c$, is not small in (14). However, nearby the optimal trajectory, $\boldsymbol{\alpha}_K^c = -(\nabla_u \nabla_u^T[G_K])^{-1} \cdot \nabla_u[G_K]$ should be small since $\nabla_u[G_K]^* = 0$ on the optimal trajectory at the final time. In the case that $\boldsymbol{\alpha}_K^c$, as when the starting guess in not good, Jacobson and Mayne [8] suggest multiplying $\boldsymbol{\alpha}_K^c$ by a sufficiently small parameter ε until the iterations are closer to the optimal trajectory. These comments apply on the time horizon, $0 \le \kappa \le K$, not just for the time $\kappa = K$.

In the presence of constraints, the linear control law (14) yields only the regular control, which must be modified for the constraints, as in the previous section for SDP. However, for inequality constraints, the inconsistency problem of non-small $\boldsymbol{\alpha}_K^c$ can arise, so penalty functions must be used to move the constraints to the objective functional.

The corresponding approximation to the optimal cost value function follows from substituting the current approximation of the optimal control:

$$\begin{aligned} \widetilde{v}_K^*(\mathbf{x}_K) &= \widetilde{v}_K(\mathbf{x}_K, \widetilde{\mathbf{u}}_K^*(\mathbf{x}_K)) \\ &= R_K^c + (\mathbf{Q}_K^c)^T \cdot \delta\mathbf{x}_K + \delta\mathbf{x}_K^T \cdot P_K^c \cdot \delta\mathbf{x}_K, \end{aligned} \tag{15}$$

reduced to a quadratic function in $\delta\mathbf{x}_K$ only. Upon calculation, the *current coefficients* are given by

$$\begin{aligned} P_K^c &= A_K^c - \frac{1}{4}(B_K^c)^T C_K^{-c} B_K^c \\ (\mathbf{Q}_K^c)^T &= (\mathbf{E}_K^c)^T - \frac{1}{2}(\mathbf{D}_K^c)^T C_K^{-c} B_K^c \\ R_K^c &= G_K^c - \frac{1}{4}(\mathbf{D}_K^c)^T C_K^{-c} \mathbf{D}_K^c. \end{aligned} \tag{16}$$

The last, currently fixed coefficient R_K^c is not essential for the iteration, unless the value optimal cost needed as a result.

3. General DDP Backward Sweep in Time

The coefficients at final time K define the beginning of the DDP backward sweep for the current iteration c, while the coefficients for the remaining iteration times are found by marching backward from $\kappa = K-1$ to $\kappa = 0$ assuming the quadratic value induction hypothesis,

$$\widetilde{v}_{\kappa+1}^*(\mathbf{x}_{\kappa+1}) = R_{\kappa+1}^c + (\mathbf{Q}_{\kappa+1}^c)^T \cdot \delta\mathbf{x}_{\kappa+1} + \delta\mathbf{x}_{\kappa+1}^T \cdot P_{\kappa+1}^c \cdot \delta\mathbf{x}_{\kappa+1}, \tag{17}$$

based on the formula (15) for $\kappa = K$, where $\delta\mathbf{x}_{\kappa+1} \equiv \mathbf{x}_{\kappa+1} - \mathbf{x}_{\kappa+1}^c$. Assuming both $\mathbf{x}_{\kappa+1}$ and $\mathbf{x}_{\kappa+1}^c$ obey the dynamics of (1) (if not, the calculation is more complex and not very suitable for implementation), then the increment in the vector dynamics is expanded as the quadratic,

$$\begin{aligned} \delta\mathbf{x}_{\kappa+1} &= \mathbf{F}_\kappa(\mathbf{x}_\kappa, \mathbf{u}_\kappa) - \mathbf{F}_\kappa(\mathbf{x}_\kappa^c, \mathbf{u}_\kappa^c) \\ &= \mathbf{F}_\kappa(\mathbf{x}_\kappa^c + \delta\mathbf{x}_\kappa, \mathbf{u}_\kappa^c + \delta\mathbf{u}_\kappa) - \mathbf{F}_\kappa(\mathbf{x}_\kappa^c, \mathbf{u}_\kappa^c) \\ &\simeq (\nabla_x \mathbf{F}_\kappa^T)^T \cdot \delta\mathbf{x}_\kappa + (\nabla_u \mathbf{F}_\kappa^T)^T \cdot \delta\mathbf{u}_\kappa \\ &+ \left[\frac{1}{2}\delta\mathbf{x}_\kappa^T \cdot (\nabla_x \nabla_x^T F_{i,\kappa}) \cdot \delta\mathbf{x}_\kappa + \delta\mathbf{u}_\kappa^T \cdot (\nabla_u \nabla_x^T F_{i,\kappa}) \cdot \delta\mathbf{x}_\kappa \right. \\ &\qquad \left. + \frac{1}{2}\delta\mathbf{u}_\kappa^T \cdot (\nabla_u \nabla_u^T F_{i,\kappa}) \cdot \delta\mathbf{u}_\kappa \right]_{n\times 1}, \end{aligned} \tag{18}$$

which is similar to the expansion of the scalar cost G_K in (11).

Next the argument of the minimum of the Principle of Optimality in (7) is expanded by substituting the quadratic expansions for G_κ in (11) and $\mathbf{F}_\kappa$ in (18), while retaining only terms up to quadratic order (being careful to expand about the later current state $\mathbf{x}_{\kappa+1}^c$ when using $\widetilde{v}_{\kappa+1}^*(\mathbf{x}_{\kappa+1})$ as required by the induction hypothesis (17)), the argument becomes

$$\begin{aligned} \widetilde{v}[\mathbf{x}, \mathbf{u}; \kappa, K] &\equiv G_\kappa(\mathbf{x}_\kappa, \mathbf{u}_\kappa) + \widetilde{v}_{\kappa+1}^*(\mathbf{F}_\kappa(\mathbf{x}_\kappa, \mathbf{u}_\kappa)) \\ \simeq \widetilde{v}_\kappa(\mathbf{x}_\kappa, \mathbf{u}_\kappa) &\equiv V_\kappa^c + (\mathbf{E}_\kappa^c)^T \cdot \delta\mathbf{x}_\kappa + (\mathbf{D}_\kappa^c)^T \cdot \delta\mathbf{u}_\kappa \\ &+ \delta\mathbf{x}_\kappa^T \cdot A_\kappa^c \cdot \delta\mathbf{x}_\kappa + \delta\mathbf{u}_\kappa^T \cdot B_\kappa^c \cdot \delta\mathbf{x}_\kappa + \delta\mathbf{u}_\kappa^T \cdot C_\kappa^c \cdot \delta\mathbf{u}_\kappa. \end{aligned} \tag{19}$$

This is the quadratic approximation to the minimization argument in (7). The corresponding coefficients are

$$\begin{aligned} V_\kappa^c &= G_\kappa^c + R_{\kappa+1}^c, \\ (\mathbf{E}_\kappa^c)^T &= \nabla_x^T[G_\kappa](\mathbf{x}_\kappa^c, \mathbf{u}_\kappa^c) + (\nabla_x \mathbf{F}_\kappa^T)^T(\mathbf{x}_\kappa^c, \mathbf{u}_\kappa^c)\mathbf{Q}_{\kappa+1}^c, \\ (\mathbf{D}_\kappa^c)^T &= \nabla_u^T[G_\kappa](\mathbf{x}_\kappa^c, \mathbf{u}_\kappa^c) + (\nabla_u \mathbf{F}_\kappa^T)^T(\mathbf{x}_\kappa^c, \mathbf{u}_\kappa^c)\mathbf{Q}_{\kappa+1}^c, \\ A_\kappa^c &= \frac{1}{2}\nabla_x \nabla_x^T[G_\kappa] + \frac{1}{2}(\nabla_x \nabla_x^T \mathbf{F}_\kappa^T)\mathbf{Q}_{\kappa+1}^c + (\nabla_x \mathbf{F}_\kappa^T)P_{\kappa+1}^c \cdot (\nabla_x \mathbf{F}_\kappa^T)^T \\ B_\kappa^c &= \nabla_u \nabla_x^T[G_\kappa] + \frac{1}{2}(\nabla_u \nabla_x^T \mathbf{F}_\kappa^T)\mathbf{Q}_{\kappa+1}^c + 2(\nabla_u \mathbf{F}_\kappa^T)P_{\kappa+1}^c \cdot (\nabla_x \mathbf{F}_\kappa^T)^T \\ C_\kappa^c &= \frac{1}{2}\nabla_u \nabla_u^T[G_\kappa] + \frac{1}{2}(\nabla_u \nabla_u^T \mathbf{F}_\kappa^T)\mathbf{Q}_{\kappa+1}^c + (\nabla_u \mathbf{F}_\kappa^T)P_{\kappa+1}^c \cdot (\nabla_u \mathbf{F}_\kappa^T)^T \end{aligned} \tag{20}$$

where arguments of the gradients at the current time in the last three coefficients have been omitted.

Minimizing the quadratic approximation in (19),

$$0 = (\nabla_u[\widetilde{v}_\kappa])^* = \mathbf{D}_\kappa^c + B_\kappa^c \cdot \delta\mathbf{x}_\kappa + 2 \cdot C_\kappa^c \cdot \delta\widetilde{\mathbf{u}}_\kappa^*,$$

yields the current approximation to the optimal control at κ,

$$\delta\widetilde{\mathbf{u}}_\kappa^* = \boldsymbol{\alpha}_\kappa^c + \beta_\kappa^c \cdot \delta\mathbf{x}_\kappa, \tag{21}$$

where

$$\boldsymbol{\alpha}_\kappa^c \equiv -\frac{1}{2}\overline{C_\kappa^{-c}} \cdot \mathbf{D}_\kappa^c \text{ and } \beta_\kappa^c \equiv -\frac{1}{2}\overline{C_\kappa^{-c}} \cdot B_\kappa^c. \tag{22}$$

These latter two coefficients need only be saved for the current approximate optimal control since they define its linear function in the state increment. The quadratic approximation is critical for the straight forward determination of the control, in this case unconstrained. Recall that in the case where $\boldsymbol{\alpha}_\kappa^c$ is not small, then a small coefficient can used where $\boldsymbol{\alpha}_\kappa^c$ is applied [11] (i.e., $\boldsymbol{\alpha}_\kappa^c$ is replaced by $\varepsilon\boldsymbol{\alpha}_\kappa^c$ with sufficiently small ε until convergence of the successive approximations to v_κ^* is assured). The difficulty arises when the approximations are too far from the optimal trajectory, but close to the optimal trajectory $\mathbf{D}_\kappa^c = \nabla_u[H_\kappa^c]$ should be small if H_κ^C is some pseudo-Hamiltonian to which the minimum principle applies.

Upon substituting the linear control law (21) into the bivariate quadratic approximation (19), the formula for induction hypothesis (17) is obtained for time κ given the result for $\kappa + 1$,

$$\widetilde{v}_\kappa^*(\mathbf{x}_\kappa) = R_\kappa^c + (\mathbf{Q}_\kappa^c)^T \cdot \delta\mathbf{x}_\kappa + \delta\mathbf{x}_\kappa^T \cdot P_\kappa^c \cdot \delta\mathbf{x}_\kappa, \tag{23}$$

proving the induction part of the current backward sweep. Further, the state dependent quadratic coefficients are given in the procedure as

$$\begin{aligned} P_\kappa^c &= A_\kappa^c + \frac{1}{2}(B_\kappa^c)^T\beta_\kappa^c \\ (\mathbf{Q}_\kappa^c)^T &= (\mathbf{E}_\kappa^c)^T - \frac{1}{2}(\mathbf{D}_\kappa^c)^T\beta_\kappa^c \\ R_\kappa^c &= V_\kappa^c - \frac{1}{4}(\mathbf{D}_\kappa^c)^T\boldsymbol{\alpha}_\kappa^c, \end{aligned} \tag{24}$$

for $\kappa = K$ to 0 in backward steps, including the final condition in (16) as well. This concludes the backward sweep for iterate c.

A primary property of DDP is that it is exact, in infinite precision, for the LQP (Linear Quadratic Problem), since the Taylor approximations would be exact in the LQP case [11].

4. DDP Forward Sweep

In the forward sweep for iterate c, the cth backward sweep approximation for the optimal control,

$$\widetilde{\mathbf{u}}_\kappa^* = \mathbf{u}_\kappa^c + \delta\widetilde{\mathbf{u}}_\kappa^* = \mathbf{u}_\kappa^c + \boldsymbol{\alpha}_\kappa^c + \beta_\kappa^c \cdot \delta\mathbf{x}_\kappa, \tag{25}$$

given in increment form by (21) for all the times $\kappa = 0$ to K, is used to calculate improvements in the optimal state trajectory. If $\boldsymbol{\alpha}_\kappa^c$ is not small, then (25) is replaced by

$$\widetilde{\mathbf{u}}_\kappa^* = \mathbf{u}_\kappa^c + \varepsilon \cdot \boldsymbol{\alpha}_\kappa^c + \beta_\kappa^c \cdot \delta \mathbf{x}_\kappa,$$

with $0 < \varepsilon \leq 1$ selected to foster convergence until $\boldsymbol{\alpha}_\kappa^c$ is small again [11, 17]. The discrete dynamic law (1),

$$\widetilde{\mathbf{x}}_{\kappa+1}^* = \mathbf{F}_\kappa(\widetilde{\mathbf{x}}_\kappa^*, \widetilde{\mathbf{u}}_\kappa^*), \tag{26}$$

can be solved recursively for $\kappa = 0$ to $K-1$ in forward steps since the linear control law approximation is given. Once done, the successor iterate is defined as

$$\mathbf{x}_\kappa^{c+1} = \widetilde{\mathbf{x}}_\kappa^*, \tag{27}$$

for $\kappa = 0$ to K. However, for computational storage considerations, the new iterate just replaces the old, $\mathbf{x}_\kappa^{c+1} \longrightarrow \mathbf{x}_\kappa^c$, saving the old iterate only for the next check of the iteration stopping criterion.

The combined backward and forward sweep iterations end when an appropriate stopping criterion is reached, such as

$$\max_\kappa ||\widetilde{v}_\kappa^*(\mathbf{x}_\kappa^{c+1}) - \widetilde{v}_\kappa^*(\mathbf{x}_\kappa^c)|| \leq \text{tol}_v,$$

in some suitable norm. This value stopping criterion may also be supplemented using successive states,

$$\max_\kappa ||\mathbf{x}_\kappa^{c+1} - \mathbf{x}_\kappa^c|| \leq \text{tol}_x,$$

or using successive controls,

$$\max_\kappa ||\mathbf{u}_\kappa^{c+1} - \mathbf{u}_\kappa^c|| \leq \text{tol}_u,$$

where the tolerances tol_γ are prescribed.

5. Similarities and Differences between DDP and SDP

The obvious difference between DDP and SDP, as presented here, is that DDP is primarily applied to deterministic problems, rather than stochastic as is SDP. However, Jacobson and Mayne [8] and others do give stochastic variants, including formulation for the LQGP (Linear, Quadratic and Gaussian Problem). These authors also present the continuous time case, as well as give estimates for the errors committed in DDP and step-size adjustments.

Another difference is that the DDP iterations are over the entire time horizon for each iterate, where as SDP iterations in the backward sweep range over only a single time step followed by a backward march in time for the next set of iterations. That is, an DDP backward sweep iteration ranges over both state and temporal spaces, whereas the SDP backward sweep ranges over only the state space and stop when successive values for that time step satisfy corrector stopping criterion. In contrast, an approximation for all times is found in DDP, before proceeding to the next iteration. In term of the time domain, roughly speaking, DDP is somewhat analogous to the point Jacobi method, while SDP is more like Gauss-Seidel.

Also, a mesh selection ratio recipe is available in SDP for selecting the ratio of the time step to an approximate measure of the space step. However, DDP does not discretize either the state $\mathbf{x}$ or the control $\mathbf{u}$ variables [17, 9] so the mesh selection ratio is not relevant, while the SDP here does discretize the state but the control is computed as output with the value. The computational and storage cost per time step of SDP is exponential $O(n \cdot M^n)$ in (28) consistent with the curse of dimensionality and with M finite difference nodes per state.

$$O\left(n \cdot M^n\right) = O\left(n \cdot e^{n \cdot \ln(M)}\right), \tag{28}$$

However, for DDP the cost per stage is the cost of computing the coefficient arrays such as A_κ^c, B_κ^c and C_κ^c from the associated Hessian arrays, as well as the control law coefficients $\boldsymbol{\alpha}_\kappa^c$ and β_κ^c, so the storage cost is $O(n^2 + m \cdot n)$ or $O(n^3 + m \cdot n^2 + m^2 \cdot n)$ if the Hessian of the dynamics vector $\mathbf{F}_\kappa$ is stored for the stage and the computational cost is $O(m^3 + m^2 \cdot n)$ for the control law and $O(n^4 + m \cdot n^3 + m^2 \cdot n^2)$ for A_κ^c, B_κ^c and C_κ^c. depending how the

multiplications in (20) are computed (see also [17, 9] for a somewhat different accounting). The DDP manages to keep the curse of dimensionality under control. A major reduction in storage and computation occurs in DDP since calculations are concentrated in the neighborhood of an optimal trajectory, whereas SDP covers the whole state space.

The assumption that cost are quadratic in the control for SDP is somewhat similar to the quadratic assumption in DDP, except that in DDP the quadratic is in both costs and dynamics with respect to both control and state. In SDP, the critical calculation of the optimal control requires only that the cost be quadratic and the dynamics be linear in the control, which is more realistic where the nonlinearity in the dynamics is an essential part of the model. In the version of SDP presented here, the quadratic costs were presented an part of the model rather than part of the method of solution, although more complex cost and dynamics could be treated by a quadratic Taylor approximation of the cost and a linear Taylor approximation of the dynamics to formulate an iterative procedure to solve the more general SDP problem. In DDP computation is mainly the computation of the temporal functional coefficients, where as in the SDP here the computation permits general state and time dependence.

Quadratic convergence for DDP has been shown by Murray and Yakowitz [13]. For SDP in continuous time, Naimipour and Hanson [14, 7] have a heuristic comparison argument for convergence, and the convergence is linear, in that,

$$V_{\mathbf{j},k}^{(\gamma+1)} - V_{\mathbf{j},k}^{(\infty)} = \theta \cdot \left(V_{\mathbf{j},k}^{(\gamma)} - V_{\mathbf{j},k}^{(\infty)} \right),$$

where γ is the correction counter and θ is the corrector convergence parameter for which the σ in (29) is an approximation.

$$\sigma = \Delta T \cdot \sqrt{\left(\sum_{i=1}^{n} \frac{2A_{ii}}{h_i^2} \right)^2 + \left(\sum_{i=1}^{n} \frac{B_i}{h_i} \right)^2} < 1. \tag{29}$$

The forward sweeps are quite different in the two cases, due to the difference in stochastic and deterministic formulations. The explicit recursion in DDP is relatively trivial compared to solving the forward Kolmogorov equations for the state density given the stochastic optimal control to get the optimal expected state trajectory.

6. DDP Variations and Applications

Yakowitz [15, 16] has given a thorough survey of the computation and techniques of differential dynamic programming in 1989. Liao and Shoemaker [10] studied convergence in unconstrained DDP methods and have found that adaptive shifts in the Hessian are very robust and yield the fastest convergence in the case that the problem Hessian matrix is not positive definite. Chang, Shoemaker and Liu [2] solve for optimal pumping rates to remediate groundwater pollution contamination using finite elements and hyperbolic penalty functions to include constraints in the DDP method. Culver and Shoemaker [3, 4] include flexible management periods into the model and use a faster Quasi-Newton version of DDP. Earlier, Murray and Yakowitz [13] had compared DDP and Newton's methods to show that DDP inherited the quadratic convergence of Newton's method. Caffey, Liao and Shoemaker [1] develop a parallel implementation of DDP that is speeded up by reducing the number of synchronization points over time steps.

I. REFERENCES

1. H. Caffey, L.-Z. Liao and C. A. Shoemaker, "Parallel Processing of Large Scale Discrete–Time Unconstrained Differential Dynamic Programming," *Parallel Computing* **19**, pp. 1003-1017 (1993).

2. L.-C. Chang, C. A. Shoemaker, and P. L.-F. Liu, "Optimal Time–Varying Pumping Rates for Groundwater Remediation: Application of a Constrained Optimal Control Algorithm," *Water Resources Research* **28** (12), pp. 3157-3173 (1992).

3. T. Culver and C. A. Shoemaker "Dynamic Optimal Control for Groundwater Remediation with Flexible Management Periods," *Water Resources Research* **28** (3), pp. 629-641 (1992).

4. T. Culver and C. A. Shoemaker "Optimal Control for Groundwater Remediation by Differential Dynamic Programming with Quasi-Newton Approximations," *Water Resources Research* **29** (4), pp. 823-831 (1993).

5. P. Dyer and S. R. McReynolds, *The Computation and Theory of Optimal Control*, Academic Press, New York, NY, 1970.

6. *Future Directions in Control Theory: A Mathematical Perspective* (W. H. Fleming, Chairman), Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

7. F. B. Hanson and K. Naimipour, "Convergence of Numerical Method for Multistate Stochastic Dynamic Programming," *Proceedings of International Federation of Automatic Control 12th World Congress* **9**, pp. 501-504 (1993).

8. D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*, American Elsevier, New York, NY, 1970.

9. L.D. Jones, R. Willis and W. W.-G. Yeh, "Optimal Control of Nonlinear Groundwater Hydraulics Using Differential Dynamic Programming," *Water Resources Research* **23**, pp. 2097-2106 (1987).

10. L.-Z. Liao and C. A. Shoemaker, "Convergence in Unconstrained Discrete-Time Differential Dynamic Programming," *IEEE Transactions on Automatic Control* **36** (6), pp. 692-706 (1991).

11. D. Q. Mayne, "A Second–Order Gradient Method for Determining Optimal Control of Non–Linear Discrete Time Systems," *International Journal of Control* **3**, pp. 85-95 (1966).

12. D. Q. Mayne, "Differential Dynamic Programming — A Unified Approach to the Optimization of Dynamical Systems," *Control and Dynamical Systems: Advances in Theory and Applications* **10** (C. T. Leondes, Editor), Academic Press, New York, NY, pp. 179-254, 1973.

13. D. M. Murray and S. J. Yakowitz, "Differential Dynamic Programming and Newton's Method for Discrete Optimal Control Problems," *Journal of Optimization and Applications* **43**, pp. 395-414 (1984).

14. K. Naimipour and F. B. Hanson, "Convergence of a Numerical Method for the Bellman Equation of Stochastic Optimal Control with Quadratic Costs and Constrained Control," *Dynamics and Control* **3** (3), pp.237-259 (1993).

15. S. Yakowitz, "Dynamic Programming Applications in Water Resources," *Water Resources Research* **18**, pp. 673-696 (1982).

16. S. Yakowitz, "Algorithms and Computational Techniques in Differential Dynamic Programming," *Control and Dynamical Systems: Advances in Theory and Applications* **31** (C. T. Leondes, Editor), Academic Press, New York, NY, pp. 75-91, 1989.

17. S. Yakowitz and B. Rutherford, "Computational Aspects of Discrete-Time Control," *Applied Mathematics and Computation* **15**, pp. 29-45 (1984).