Advantages of Differential Dynamic Programming Over Newton's Method for Discrete-Time Optimal Control Problems

Li-zhi Liao and Christine A. Shoemaker

Cornell University

Abstract

Differential Dynamic Programming (**DDP**) and stagewise Newton's method are both quadratically convergent algorithms for solving discrete time optimal control problem. Although these two algorithms share many theoretical similarities, they demonstrate significantly different numerical performance. In this paper, we will compare and analyze these two algorithms in detail, and derive another quadratically convergent algorithm which is a combination of the DDP algorithm and Newton's method. This new second-order algorithm plays a key role in the explanation of the numerical differences between the DDP algorithm and Newton's method. The detailed algorithmic and structural differences for these three algorithms and their impact on numerical performance will be discussed and explored. Two test problems with various dimensions solved by these three algorithms will be presented. One nonlinear test problem demonstrates that the DDP algorithm can be as much as 28 times faster than the stagewise Newton's method. The numerical comparison indicates that the DDP algorithm is numerically superior to the stagewise Newton's method.

Abbreviated Title: Differential Dynamic Programming versus Newton for OCP

Key Words: differential dynamic programming, Newton's method, optimal control, numerical optimization

AMS (MOS) subject classification: 49L20, 49M15, 49N35, 93B40, 93B52, 93C55

Li-zhi Liao is Research Associate, Advanced Computing Research Institute, Cornell Theory Center, Cornell University, Ithaca, N.Y. 14853, (607)-254-8863.

Christine A. Shoemaker is Professor, School of Civil and Environmental Engineering and Center for Applied Mathematics, Hollister Hall, Cornell University, Ithaca, N.Y. 14853, (607)-255-9233.

1. Introduction

The unconstrained discrete time optimal control problem has the following format

$$\min_{(u_1,\ldots,u_{N-1})} \quad J = \sum_{t=1}^{N-1} g(x_t, u_t, t) + g(x_N, N) \tag{1.1}$$

$$\text{where} \quad x_{t+1} = \hat{f}(x_t, u_t, t) \quad t = 1, \ldots, N-1, \tag{1.2}$$
$$x_1 \equiv \bar{x}_1 \text{ is given and fixed}$$

where $x_t \in R^{\hat{n}}$ is called state variable, $u_t \in R^m$ is called control variable, $g: R^{\hat{n}+m+1} \to R^1$ in C^2 is called loss functions, and $\hat{f}: R^{\hat{n}+m+1} \to R^{\hat{n}}$ in C^2 is called transition function.

The optimal control problem (**OCP**) (1.1) – (1.2) is a special type of optimization problem in that the underlying system is dynamic and the state variables are required to connect the different time periods. Notice that in problem (1.1) – (1.2), the state variable x_t can be eliminated by transition function $\hat{f}$ so that problem (1.1) – (1.2) can be transformed into the following unconstrained nonlinear programming problem

$$\min_{\vec{U} \in R^{(N-1)\cdot m}} \quad J(\vec{U}) \qquad \text{where} \quad \vec{U} = (u_1^T, \cdots, u_{N-1}^T)^T. \tag{1.3}$$

Due to the special structure in the optimal control problem (1.1) – (1.2), many algorithms have been proposed to explore and take advantage of this feature. Mitter [8], Mayne [7] and Jacobson [3] introduced three efficient second order algorithms for optimal control problems. Since the introduction of the DDP algorithm, additional research has explored and enhanced both theoretical and numerical properties of the DDP algorithm (see Jacobson and Mayne [4], Murray and Yakowitz [9], Yakowitz and Rutherford [17], and Liao and Shoemaker [5], [6]). Ohno [10] and Yakowitz [15] have developed methods for inclusion of constraints in optimal control algorithms related to DDP. Dunn [1] proposed some gradient projection methods. Wright [13], [14] suggested several second order algorithms for the optimal control problem in the parallel processing environment. Other algorithms for solving optimal control problems can be found in the survey paper by Yakowitz [16]. Liao and Shoemaker [6] have shown that the computational complexity of the DDP algorithm per iteration for (1.1) – (1.2) is

$$N \cdot \left(2\hat{n}^3 + \frac{7}{2}\hat{n}^2 m + 2\hat{n}m^2 + \frac{1}{3}m^3\right). \tag{1.4}$$

The direct application of Newton's method to problem (1.3) will result in $O(N^3)$ operations. For large number of time periods N, which is very common in practice, this method is very expensive and does not exploit the special structure of problem (1.1) – (1.2). Although the direct application of Newton's method to optimal control problems as mentioned above is numerically inefficient, it has important theoretical interest. Murray and Yakowitz [9] used the local quadratic convergence property of Newton's method to assist in the proof of quadratic convergence of the DDP algorithm.

A recent paper by Pantoja [11] developed an algorithm that can generate exactly the same Newton step as solving problem (1.3) directly by Newton's method. One significant feature of his modified DDP algorithm is that it generates the Newton step while keeping the computation in $O(N)$ flops per iteration. From now on, we will call his modified DDP algorithm the "stagewise Newton" method or just Newton's method. Dunn and Bertsekas [2] proposed another algorithm that also generates the Newton step for problem (1.1)–(1.2) with computation in $O(N)$ flops. The derivation of their algorithm is different from the stagewise Newton's method proposed by Pantoja [11] but generates the same search direction.

Since both the DDP algorithm and Newton's method have local quadratic convergence, and same order of computational complexity, it might be expected that these two algorithms would have similar numerical performance. However, our numerical experiments demonstrate significant differences in convergence. Therefore, it is the purpose of this paper to investigate and analyze the detailed algorithmic and structural differences of these two algorithms, and explain the observed different numerical performance.

In Section 2 of this paper, we will discuss the structural differences between the DDP algorithm and Newton's method. This analysis enables us to introduce a new algorithm which shares some features of the DDP algorithm and Newton's method. The impact of the differences among these three algorithms on numerical performance will also be addressed in this section. The issue of relative numerical efficiency is of considerable practical significance because currently in engineering practice nonlinear programming packages (related to Newton's method) are often used to solve control application problems rather than to use an algorithm that is specially designed for optimal control problems (like DDP). In Section 3, we will present numerical results of these three algorithms on two test problems with various dimensions.

2. Theoretical Analysis of the Algorithms

The optimal control problem $(1.1)-(1.2)$ can be easily transformed into the following format

$$\min_{(u_1,\ldots,u_{N-1})} \quad J = g(x_N, N) \tag{2.1}$$

$$\text{where} \;\; x_{t+1} = f(x_t, u_t, t) \;\; t = 1, \ldots, N-1, \tag{2.2}$$
$$x_1 \equiv \bar{x}_1 \text{ is given and fixed and } \vec{U} = (u_1^T, \cdots, u_{N-1}^T)^T.$$

The transformed problem $(2.1)-(2.2)$ has a larger dimension for state variable (increased by 1, i.e. $n = \hat{n} + 1$) and different transition function. The definition of the function f in problem $(2.1) - (2.2)$ is not important here since it has no impact in the following discussion. However, it should be mentioned that nonlinearity is preserved, i.e. f is nonlinear if and only if $\hat{f}$ is nonlinear. The detailed transformation and the definitions of the new functions and state variables are discussed by Pantoja in [11]. This new format for optimal control problems enables us to see the algorithmic differences between the DDP algorithm and Newton's method more clearly.

2.1 Newton's Method

The following is our algorithmic description of the stagewise Newton procedure developed by Pantoja [11]. Pantoja's stagewise Newton's method gives the identical value of the decision variable $\vec{U}$ at each iteration as the conventional Newton's method, but does the computations differently. Pantoja's paper gives a proof of the equivalence of the stagewise Newton and conventional Newton procedures. However, Pantoja [11] does not give an algorithmic description of the stagewise Newton's method nor provide any numerical results for stagewise Newton or comparison of DDP to Newton. An algorithmic description is very useful for analyzing the numerical aspects of the algorithm and for developing code. In the algorithmic description and numerical results, the only modification we have made in Pantoja's algorithm is to add a line search procedure (Step 3) in order to improve convergence.

Stagewise Newton's Method for Solving OCP

Step 1: (Initialize parameters and compute loss and trajectory associated with the given policy $\bar{\vec{U}}$)

i) select starting guess $\bar{\vec{U}} = (\bar{u}_1^T, \cdots, \bar{u}_{N-1}^T)^T$ for policy and select $\bar{\theta}$ as stopping criterion for θ.

ii) compute $\bar{x}_{t+1} = f(\bar{x}_t, \bar{u}_t, t)$, recursively for $t = 1, \ldots, N-1$, $\bar{x}_1 \equiv \bar{x}_1$;

iii) compute $J(\bar{\vec{U}}) = g(\bar{x}_N, N)$;

iv) initialize $P_N = g_{xx}$, $Q_N = g_x$, $G_N = g_x$, $\theta_N = 0$.

Step 2: (Perform the backward sweep: Perform steps i) to iv) below, recursively, for $t = N-1, \ldots, 1$)

i) compute A_t, B_t, C_t, D_t, E_t and G_t according to

$$A_t = \left[\left(\frac{\partial f}{\partial x}\right)^T P_{t+1}\left(\frac{\partial f}{\partial x}\right) + \sum_{i=1}^{n}(G_{t+1})_i (f_i)_{xx}\right]\Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{N.1}$$

$$B_t^T = \left[\left(\frac{\partial f}{\partial x}\right)^T P_{t+1}\left(\frac{\partial f}{\partial u}\right) + \sum_{i=1}^{n}(G_{t+1})_i (f_i)_{xu}\right]\Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{N.2}$$

$$C_t = \left[\left(\frac{\partial f}{\partial u}\right)^T P_{t+1}\left(\frac{\partial f}{\partial u}\right) + \sum_{i=1}^{n}(G_{t+1})_i (f_i)_{uu}\right]\Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{N.3}$$

$$D_t = \left[\left(\frac{\partial f}{\partial u}\right)^T Q_{t+1}\right]\Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{N.4}$$

$$E_t = \left[\left(\frac{\partial f}{\partial x}\right)^T Q_{t+1}\right]\Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{N.5}$$

$$G_t = \left[\left(\frac{\partial f}{\partial x}\right)^T G_{t+1}\right]\Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{N.6}$$

ii) compute P_t, Q_t according to

$$P_t = A_t - B_t^T C_t^{-1} B_t, \tag{N.7}$$

$$Q_t = -B_t^T C_t^{-1} D_t + E_t, \tag{N.8}$$

and store P_t and Q_t in memory, replacing P_{t+1} and Q_{t+1}.

iii) Compute α_t, β_t according to

$$\alpha_t = -C_t^{-1} D_t, \tag{N.9}$$

$$\beta_t = -C_t^{-1} B_t, \tag{N.10}$$

and store in memory.

iv) Compute

$$\theta_t = D_t^T C_t^{-1} D_t + \theta_{t+1}, \tag{N.11}$$

and store θ_t in place of θ_{t+1} in memory.

Step 3: (Perform the forward sweep: compute the successor policy and trajectory)

i) Set $\varepsilon = 1$;

ii) (Compute the new value of the control vector)

Compute $u_t(\varepsilon)$ recursively for $t = 1, \ldots, N-1$ according to

$$u_t(\varepsilon) = \bar{u}_t + \varepsilon \cdot \alpha_t + \beta_t \cdot (\hat{x}_t - \bar{x}_t), \tag{N.12}$$

$$\hat{x}_{t+1} = \bar{x}_{t+1} + \left.\frac{\partial f}{\partial x}\right|_{(x=\bar{x}_t, u=\bar{u}_t)} \cdot (\hat{x}_t - \bar{x}_t) + \left.\frac{\partial f}{\partial u}\right|_{(x=\bar{x}_t, u=\bar{u}_t)} \cdot (u_t(\varepsilon) - \bar{u}_t), \tag{N.13}$$

iii) (Compute the new value of state vector)

Compute $x_t,\ t = 2, \cdots, N$ according to

$$x_t = f(x_{t-1}, u_{t-1}(\varepsilon), t-1), \tag{N.14}$$

iv) Compute $J(\vec{U}(\varepsilon)) = g(x_N, N)$;

iv) If $J(\vec{U}(\varepsilon)) - J(\bar{\vec{U}}) < \varepsilon \dfrac{\theta_1}{2}$, then

a) Set $\bar{u}_t = u_t(\varepsilon)$ and $\bar{x}_t = x_t$ for all t,

b) If $\theta_1 \geq \bar{\theta}$, go to step 2 with new values of $(\bar{x}_t, \bar{u}_t)$,
If $\theta_1 < \bar{\theta}$, stopping criterion is satisfied, so stop.

If $J(\vec{U}(\varepsilon)) - J(\bar{\vec{U}}) \geq \varepsilon \dfrac{\theta_1}{2}$, then set $\varepsilon = \varepsilon/2$ and go to Step 3 ii).

The line search procedure, Step 3 in the above Newton's method, is our addition since this is needed for most nonlinear problems. Below we will prove that the line search procedure used in the above Newton's method preserves the Newton search direction. But, first let us introduce some notation.

Notation:

a) Let $\vec{U}(\varepsilon) = \left((u_1(\varepsilon))^T, \cdots, (u_{N-1}(\varepsilon))^T\right)^T$, where $u_t(\varepsilon)$ $t = 1, \cdots, N-1$ is determined by equations (N.12) and (N.13).

b) Let $\vec{p}(\varepsilon) = \left((u_1(\varepsilon) - \bar{u}_1)^T, \cdots, (u_{N-1}(\varepsilon) - \bar{u}_{N-1})^T\right)^T$, where $u_t(\varepsilon)$ $t = 1, \cdots, N-1$ is determined by equations (N.12) and (N.13), and $(\bar{u}_1, \cdots, \bar{u}_{N-1})$ is nominal policy.

Then obviously, $\vec{p}(1)$ is the search direction for Newton's method.

Theorem 1: Let $\vec{p}(\varepsilon)$ be the vector defined above, then, $\vec{p}(\varepsilon) = \varepsilon \cdot \vec{p}(1)$ and hence Step 3 of our stagewise Newton procedure (N.12)–(N.14) preserves the search direction.

Proof: Let

$$\begin{cases} p_t(\varepsilon) = u_t(\varepsilon) - \bar{u}_t \quad t = 1, \cdots, N-1 \\ q_t = x_t - \bar{x}_t \quad t = 1, \cdots, N \end{cases} . \tag{2.3}$$

Then, $\vec{p}(\varepsilon) = ((p_1(\varepsilon))^T, \cdots, p_{N-1}(\varepsilon))^T)^T$. From equations (N.12) and (N.13), we have

$$\begin{cases} p_t(\varepsilon) = \varepsilon \cdot \alpha_t + \beta_t \cdot q_t \quad t = 1, \cdots N-1 \\ q_{t+1} = \frac{\partial f}{\partial x} \cdot q_t + \frac{\partial f}{\partial u} \cdot p_t(\varepsilon) \quad t = 1, \cdots, N-1 \end{cases} . \tag{2.4}$$

Notice that since $x_1 \equiv \bar{x}_1$ is given and fixed, then $q_1 = 0$. Therefore, by induction on (2.4), it is easy to check that $p_t(\varepsilon) = \varepsilon \cdot p_t(1)$ for $t = 1, \cdots, N-1$. ∎

Theorem 1 indicates that the line search procedure in the above Newton's method does not change the search direction.

2.2 The DDP Algorithm

The DDP algorithm is an iterative algorithm. Within each iteration, there are two sweeps: one "backward sweep" to compute the feedback functions in the form of $u_t = \alpha_t + \beta_t \cdot x_t$ backward in time following the dynamic programming optimization scheme; and one "forward sweep" to update the sequences of state and control variables (x_t, u_t) through transition equation (2.2) and feedback functions $u_t = \alpha_t + \beta_t \cdot x_t$ forward in time.

Below is an algorithmic description of the DDP algorithm. This description follows closely the algorithmic description of DDP provided by Yakowitz and Rutherford [17]. The major modification in our algorithmic description is a more detailed discussion of points at which derivatives are evaluated and the updating procedure.

The DDP Algorithm for Solving OCP

Step 1: (Initialize parameters and compute loss and trajectory associated with the given policy $\bar{\vec{U}}$)

i) select starting guess $\bar{\vec{U}} = (\bar{u}_1^T, \cdots, \bar{u}_{N-1}^T)^T$ for policy and select $\bar{\theta}$ as stopping criterion for θ.

ii) compute $\bar{x}_{t+1} = f(\bar{x}_t, \bar{u}_t, t)$, recursively for $t = 1, \ldots, N-1$, $\bar{x}_1 \equiv \bar{x}_1$;

iii) compute $J(\bar{\vec{U}}) = g(\bar{x}_N, N)$;

iv) initialize $P_N = g_{xx}$, $Q_N = g_x$, $\theta_N = 0$.

Step 2: (Perform the backward sweep: Perform steps i) to iv) below, recursively, for $t = N-1, \ldots, 1$)

i) compute A_t, B_t, C_t, D_t and E_t according to

$$A_t = \left[\left(\frac{\partial f}{\partial x} \right)^T P_{t+1} \left(\frac{\partial f}{\partial x} \right) + \sum_{i=1}^{n} (Q_{t+1})_i (f_i)_{xx} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{D.1}$$

$$B_t^T = \left[\left(\frac{\partial f}{\partial x} \right)^T P_{t+1} \left(\frac{\partial f}{\partial u} \right) + \sum_{i=1}^{n} (Q_{t+1})_i (f_i)_{xu} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{D.2}$$

$$C_t = \left[\left(\frac{\partial f}{\partial u} \right)^T P_{t+1} \left(\frac{\partial f}{\partial u} \right) + \sum_{i=1}^{n} (Q_{t+1})_i (f_i)_{uu} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{D.3}$$

$$D_t = \left[\left(\frac{\partial f}{\partial u} \right)^T Q_{t+1} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{D.4}$$

$$E_t = \left[\left(\frac{\partial f}{\partial x} \right)^T Q_{t+1} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{D.5}$$

ii) compute P_t, Q_t according to

$$P_t = A_t - B_t^T C_t^{-1} B_t, \tag{D.6}$$

$$Q_t = -B_t^T C_t^{-1} D_t + E_t, \tag{D.7}$$

and store P_t and Q_t in memory, replacing P_{t+1} and Q_{t+1}.

iii) Compute α_t, β_t according to

$$\alpha_t = -C_t^{-1} D_t, \tag{D.8}$$

$$\beta_t = -C_t^{-1} B_t, \tag{D.9}$$

and store in memory.

iv) Compute

$$\theta_t = D_t^T C_t^{-1} D_t + \theta_{t+1}, \tag{D.10}$$

and store θ_t in place of θ_{t+1} in memory.

Step 3: (Perform the forward sweep: compute the successor policy and trajectory)

i) Set $\varepsilon = 1$;

ii) Compute $u_t(\varepsilon)$ and x_t recursively for $t = 1, \ldots, N-1$ according to

$$u_t(\varepsilon) = \bar{u}_t + \varepsilon \cdot \alpha_t + \beta_t \cdot (x_t - \bar{x}_t), \tag{D.11}$$

$$x_{t+1} = f(x_t, u_t(\varepsilon), t), \tag{D.12}$$

iii) Compute $J(\vec{U}(\varepsilon)) = g(x_N, N)$,

iv) If $J(\vec{U}(\varepsilon)) - J(\bar{\vec{U}}) < \varepsilon \dfrac{\theta_1}{2}$,

a) Set $\bar{u}_t = u_t(\varepsilon)$ and $\bar{x}_t = x_t$ for all t,

b) If $\theta_1 \geq \bar{\theta}$, go to step 2 with new values of $(\bar{x}_t, \bar{u}_t)$,
If $\theta_1 < \bar{\theta}$, stopping criterion is satisfied, so stop.

If $J(\vec{U}(\varepsilon)) - J(\bar{\vec{U}}) \geq \varepsilon \dfrac{\theta_1}{2}$, then set $\varepsilon = \varepsilon/2$ and go to Step 3 ii).

In the DDP algorithm, C_t (equation (D.3)) is the stagewise Hessian matrix, while D_t (equation (D.4)) is the stagewise gradient vector. Mayne [7] proved that with the line search strategy of equation (D.11), the reduction in $J(\vec{U})$ is

$$J\big(\vec{U}(\varepsilon)\big) - J\big(\bar{\vec{U}}\big) \;=\; -\varepsilon\Big(1 - \frac{\varepsilon}{2}\Big)\theta_1 + o(\varepsilon^2) \tag{2.5}$$

Therefore, θ_1 in equation (D.10) represents both the improvement of the new policy and a measure of closeness to the optimal solution. ($D_t = 0$ at the optimal solution.) Yakowitz and Rutherford [17] provide further discussion about the role of θ_1, and use the value of θ_1 as the stopping criterion. Our stopping criterion is same as the one used in [17].

2.3 Differences Between DDP and Stagewise Newton's Method

Comparison between the equations in Section 2.1 and 2.2 shows the following differences between DDP and stagewise Newton:

Differences Between DDP and Newton In:

Forward sweep: In Step 3 for DDP, the optimal control u_t is computed using the value of x_t computed from the full nonlinear function $f(x_t, u_t, t)$ (equations (D.11)–(D.12)) whereas for Newton the optimal control u_t is computed using a linearized version of $f(x_t, u_t, t)$ (equations (N.12) – (N.13)) and then (for this fixed value of u_t), x_t is computed by forward simulation of the full nonlinear function $f(x_t, u_t, t)$ (N.14). Hence, for Newton's method there are two forward simulations ((N.13) and (N.14)) whereas for DDP there is a single forward simulation (D.12).

Backward sweep: In Step 2, both methods compute Q_t recursively and use it to compute the matrices D_t and E_t. However, Newton's method computes an additional matrix G_t (N.6). In the equations for A_t, B_t and C_t, DDP multiplies second derivative terms by Q_t, whereas Newton's method uses the matrix G_t in the equations for A_t, B_t and C_t.

Notice that the difference between the equations for G_t and Q_t is the terms $B_t^T C_t^{-1} D_t$. The term $B_t^T C_t^{-1} D_t$ tends to zero. (D_t can be viewed as the gradient, so $D_t \to 0$.) Hence as the algorithm converges, the differences between the DDP and Newton algorithms will decrease near the optimal solution. But the differences between the algorithms will be significant when the current solution is far away from the optimal solution. Our numerical results later also observe this phenomenon.

In Appendix II, analytical solutions from the stagewise Newton's method in §2.1, the DDP algorithm in §2.2, and direct Newton's method for a three-stage generic optimal control problem are provided to demonstrate the detailed differences between the DDP algorithm and Newton's method for a general algebraic problem.

[9] and [11] comment that DDP and Newton's method give different search directions; however, neither paper does an explicit comparison of DDP and Newton's method. Hence, the example in Appendix II is the first formal proof that DDP and the direct Newton's method do not necessarily give the same search directions.

2.4 A Mixed Algorithm and Its Relationship to DDP and Newton

In order to examine the relative importance of the differences between the forward and backward sweeps for DDP versus Newton, we construct a third algorithm call the "Mixed" algorithm. The Mixed algorithm uses the forward sweep of DDP and the backward sweep of the stagewise Newton's method. Because the DDP forward sweep uses the full nonlinear expression for f to compute the control u_t, we would expect it to be better than the Newton forward sweep. However, it is not obvious which backward sweep approach is better. Comparing the results for the DDP algorithm to the Mixed algorithm examines the comparative advantages of the backward sweeps of the Newton versus DDP methods. Comparing the results for the Mixed versus Newton methods quantifies the comparative advantages of the DDP versus Newton forward sweeps. A detailed description of the Mixed algorithm is provided in Appendix III.

Since both the DDP algorithm and Newton's method have local quadratic convergence, from the structure of this Mixed algorithm, it is expected that this algorithm also has local quadratic convergence. The following Theorem 2 establishes this property.

Theorem 2: The Mixed algorithm has quadratic convergence.

Proof: Following exactly the same arguments in the proof of the Theorem in Liao and Shoemaker [5], we can conclude that the Mixed algorithm has quadratic convergence. ∎

Similarities Between The Three Algorithms

From the formats of the three algorithms, it is straightforward to see that the stagewise Newton's method and Mixed method share the same computational complexity per iteration as the DDP algorithm.

In summary, Newton's method, DDP and the Mixed algorithm for solving optimal control problem (2.1) – (2.2) share the followings:

a) All three algorithms are second order algorithms, i.e. all first and second derivatives of objective and transition functions are required.

b) They all have local quadratic convergence.

c) The leading terms of the computational complexity for all three algorithm are same.

d) When the transition functions f are linear, these three algorithms generate the same sequence of control variables at each iteration of the optimization.

3. Numerical Comparison of the Algorithms

In all the codes for the following test problems, the specific sparsity involved in each computation has been considered. For example $(f_i)_{xx} = 0 \; i = 1, \cdots, n$ in Test Problem 1, and $(f_i)_{xx}$ is diagonal for $i = 1, \cdots, n$ in Test Problem 2. All runs were performed on an IBM 3090-600E computer, and all CPU times are in seconds. The stopping criterion for the first problem is $\bar{\theta} = 10^{-3}$, and $\bar{\theta} = 10^{-4}$ for the second problem.

In all the numerical runs, a "Standardized Shift Scheme" procedure is adopted for nonconvex situations to generate convergent scheme for each case. Liao and Shoemaker [6] show that if f is nonlinear, then the "stagewise Hessian" C_t in equation (D.3) may be nonpositive definite for some t. If C_t is nonpositive definite for any t, the DDP algorithm is not guaranteed to converge [6]. Since for nonconvex problems a procedure is required to guarantee convergence if the stagewise Hessian C_t is nonpositive definite, we adopt the adaptive shift procedure suggested for the DDP algorithm by Liao and Shoemaker [6]. The adaptive shift procedure adds values to the diagonal of the C_t matrix to make it positive definite if it is not. These values depend upon the minimum eigenvalue of C_t and upon two parameters δ and ε. In Appendix I, a description is given for the shift procedure and a general standardized shift scheme is provided for these three algorithms to establish the values of the parameters δ and ε used in the shift schemes in nonconvex cases.

Test Problem 1: A Modestly Nonlinear Problem With Convex Object Function

We construct the first test problem to demonstrate a) the quadratic convergence of the DDP algorithm, the Mixed algorithm and Newton's method (Table 1); b) the impact of theoretical differences among these three algorithms on the convergence in a problem that is modestly nonlinear (Table 2 and Table 3); and c) the occurrence of nonpositive definite C_t matrices that must be modified (at cost of loss of efficiency) to guarantee convergence for the three algorithms from various starting points (Table 4 and Table 6). This test problem has a convex quartic objective function and a nonlinear transition equation. The amplitude of nonlinearity in the transition function is controlled by a scalar μ. The larger the parameter μ is, the more nonlinear the transition function is. The following Table 1 through Table 6 summarize all the numerical results for this test problem.

Test Problem 1:

$$\min\ J \;=\; \sum_{t=1}^{N-1}\ \Big(\sum_{i=1}^{n}(x_t^i+\frac{1}{4})^4+\sum_{j=1}^{m}(u_t^j+\frac{1}{2})^4\Big)+\sum_{i=1}^{n}(x_N^i+\frac{1}{4})^4$$

$$\text{where } x_{t+1} = f(x_t,u_t,t) = Ax_t+Bu_t+(x_t^T C u_t)\cdot\gamma \qquad t=1,\cdots,N-1,$$

$$x_1=(0,\cdots,0)^T\in R^n,$$

$A\in R^{n\times n}$ and

$$(A)_{i,j}=\begin{cases}0.5 & \text{if } i=j\\ 0.25 & \text{if } j=i+1\\ -0.25 & \text{if } j=i-1\end{cases}\qquad i,j=1,\cdots,n,$$

$B \in R^{n\times m}$, $(B)_{i,j} = \frac{i-j}{n+m}$, $i = 1, \cdots, n$ and $j = 1, \cdots, m$,

$C \in R^{n\times m}$ and $(C)_{i,j} = \mu \cdot \frac{i+j}{(n+m)}$, μ is a scalar,

$\gamma \in R^n$ and $\gamma = (1, \cdots, 1)^T$,

Initial policy $\bar{u}_t = (0, \cdots, 0)^T \in R^m$, $t = 1, \cdots, N$.

Slight Nonlinearity From $x_t^T C u_t$ Term

In the first experiment, μ is equal to 1/200, which results in a transition function f that is only slightly nonlinear. Table 1 indicates that for a transition function that is close to linear, the stagewise Hessian matrices C_t are all positive definite in all three algorithms for the origin as the starting point. Results in Table 1 imply that in convex situation, the number of iterations for the DDP algorithm and Mixed algorithm is a little less than Newton's method. This is because a different forward sweep (linear approximation and open loop) is used in Newton's method. Table 1 also indicates that the CPU time per iteration for Newton's method (2.40 sec./iter.) is a little more than the CPU time for the DDP algorithm (2.37 sec./iter.). This is because Newton's method requires one more forward simulation than the DDP algorithm in each iteration (see equation (N.13)).

Table 1. Values of Objective Function at Each Iteration

with $n = 100$, $m = 50$, $N = 20$ and $\mu = 1/200$ for Test Problem 1

	Objective Function Value		
	DDP	Mixed	Newton
Iter. No.	No Shift	No Shift	No Shift
0	67.187500	67.187500	67.187500
1	60.951912	60.962273	60.165072
2	59.197939	59.261030	58.409507
3	58.217720	58.368417	58.033024
4	57.761776	57.805362	57.913669
5	57.728644	57.729955	57.746615
6	57.727773	57.727779	57.729144
7	57.727771	57.727771	57.727822
8	optimal	optimal	57.727773 optimal
Vir. CPU (sec.)	16.58	16.63	19.19

Low Nonlinearity From $x_t^T C u_t$ Term

In the experiments summarized in Table 2, we increase the nonlinearity of the problem by increasing μ to $1/75$. The results in Table 2 indicate that for this value of μ, the stagewise Hessian matrices C_t for the DDP algorithm are all positive definite, therefore no shift procedure is needed for the convergence. Since the stagewise Hessian matrices C_t for the Mixed algorithm and Newton's method are not all positive definite during the iterations, a shift procedure is needed to guarantee the convergence. The shift procedure is discussed in the second paragraph of Section 3 "Numerical Comparison of the Algorithms" and in Appendix IV.

Table 2. Numerical Results for Test Problem 1

with $n = 100$, $m = 50$, $N = 20$ and $\mu = 1/75$

	DDP	Mixed		Newton	
	No Shift	No Shift	Scheme 1*	No Shift	Scheme 1*
# of iter.	6	does not	8	does not	8
optimal obj. value	57.90802	converge	57.90802	converge	57.90803
Vir. CPU (sec.)	14.19		19.98		20.44

*: Shift Scheme 1 is defined in Appendix IV.

Medium Nonlinearity From $x_t^T C u_t$ Term

In the third experiment, we increase the nonlinearity; and as a result, shifts are required for all three algorithms. Table 3 summarizes the numerical results for $\mu = 1/20$. The Newton and Mixed methods require substantially more iterations and CPU than the DDP method.

Table 3. Numerical Results for Test Problem 1

with $n = 100$, $m = 50$, $N = 20$ and $\mu = 1/20$

	DDP	Mixed	Newton
	Scheme 2*	Scheme 3*	Scheme 3*
# of iter.	9	13	14
optimal obj. value	58.32138	58.32138	58.32139
Vir. CPU (sec.)	23.14	32.55	35.94

*: Shift Scheme 2 and Scheme 3 are defined in Appendix IV.

Effects Of Starting Points On "Slight" and "Low" Nonlinear Cases

It is well known that the number of iterations required for convergence depends upon the starting points $\{\bar{u}_t \mid t = 1, \cdots, N\}$. Table 4 and 5 show the effect of starting values on the number of iterations required by the three algorithms. In the example n has been increased to 100, whereas in Table 1, 2 and 3 $n = 20$.

The tables indicate that DDP never required more iterations than Newton's method and on average required fewer iterations. The results also indicate that the DDP algorithm required shifts less often than did either of the other algorithms.

Table 4. Numerical Results for "Slightly Nonlinear" Test Problem 1 with $n = 100$, $m = 50$, $N = 100$ and $\mu = 1/200$

	DDP		Mixed		Newton	
starting point	shift required	No. of iter.	shift required	No. of iter.	shift required	No. of iter.
1	No	8	No	8	No	8
2	No	9	Yes	10	Yes	10
3	Yes	5	Yes	5	Yes	6
4	No	9	Yes	9	Yes	9
5	No	8	Yes	7	Yes	9
average	20%	7.9	80%	7.9	80%	8.4

Table 5. Starting points for Table 4 and Table 6

Starting Points	
1	$(\bar{u}_t)_i = 0.0 \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$
2	$(\bar{u}_t)_i = 0.01 \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$
3	$(\bar{u}_t)_i = -0.01 \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$
4	$\begin{cases} (\bar{u}_t)_i = 0.01 & t \text{ is odd} \\ (\bar{u}_t)_i = -0.01 & t \text{ is even} \end{cases} \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$
5	$\begin{cases} (\bar{u}_t)_i = -0.01 & t \text{ is odd} \\ (\bar{u}_t)_i = 0.01 & t \text{ is even} \end{cases} \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$

Table 6. Numerical Results for "Low Nonlinear" Test Problem 1 with $n = 100$, $m = 50$, $N = 100$ and $\mu = 1/75$

	DDP		Mixed		Newton	
starting point*	shift required	No. of iter.	shift required	No. of iter.	shift required	No. of iter.
1	No	8	Yes	9	Yes	9
2	Yes	11	Yes	14	Yes	14
3	Yes	7	Yes	8	Yes	9
4	Yes	8	Yes	10	Yes	11
5	Yes	7	Yes	8	Yes	9
average	80%	8.2	100%	9.8	100%	10.4

*: see Table 5.

We should note that Test Problem 1 is not very nonlinear both because of the small value of μ and because $(f_i)_{xx} = 0$ and $(f_i)_{xu} = 0$. The differences between the DDP, Newton and Mixed methods are relatively small when averaged over a number of starting points. The next section discusses a more nonlinear test problem where the advantages of DDP are more significant.

Test Problem 2: A Highly Nonlinear Problem with Nonconvex Objective and Transition Functions

The second test problem we constructed is a modification of the test problem used by Yakowitz and Rutherford [17]. This is a nonconvex and very nonlinear problem; both the objective function and transition function are nonconvex and nonlinear. Therefore, in general, the stagewise Hessian matrices C_t in all three algorithms are not positive definite. The standardized shift scheme in Appendix I was used in all the runs to guarantee and enhance the convergence.

Test Problem 2:

$$\min \ J = \sum_{t=1}^{N-1} \|x_t\|_2^2 \cdot [\sin^2(\frac{\|u_t\|_2^2}{m}) + 1] + \|x_N\|_2^2$$

where $x_{t+1}^i = f(x_t, u_t, t) = \sin(x_t^i) + [F \cdot W(u_t)]_i \quad i = 1, \cdots, n; \ t = 1, \cdots, N-1,$

x_t^i is the ith component of x_t, $x_t \in R^n$ and $u_t \in R^m$,

$x_1^i = \frac{i}{2n} \quad i = 1, \cdots, n,$

$(F)_{i,j} = \frac{(i+j)}{2n} \quad i = 1, \cdots, n$ and $j = 1, \cdots, m,$

$W(u_t) = (\sin(u_t^1), \cdots, \sin(u_t^m))^T \in R^m,$

u_t^i is the ith component of u_t,

Initial policy $u_t = (0, \cdots, 0)^T \in R^m, \quad t = 1, \cdots, N-1.$

The results in Table 7 for the standardized shift scheme indicate that the Newton's method requires more than twice as many iterations and CPU as the DDP algorithm for Test Problem 2 with $N = 20$. The differences among the three algorithms are more significant for this test problem than for Test Problem 1. This is presumably because the transition function $f(x_t, u_t, t)$ is more nonlinear since $(f_i)_{xx}$ and $(f_i)_{uu}$ are nonzero in Test Problem 2 and are zero in Test Problem 1. In addition the objective function in Test Problem 2 is nonconvex, whereas it is convex in Test Problem 1.

The results for the Mixed algorithm in Table 7 indicate that it performs similarly to the DDP algorithm. The Mixed algorithm has the same forward sweep as the DDP algorithm and the same backward sweep as Newton's method. The results in Table 7 indicate that for Test Problem 2, with $N = 20$ the difference between DDP and Newton's method in the forward sweep is more important than the differences in the backward sweep. Later Tables support this observation for larger values of N.

Table 7. Numerical Results for $n = 100$, $m = 10$, $N = 10$ of Test Problem 2

	DDP	Mixed	Newton	Ratio
standardized shift scheme	Scheme 4*	Scheme 5*	Scheme 6*	Newton/DDP
# of iteration	7	8	16	2.3
optimal obj. value	8.46798	8.46798	8.46799	
Vir. CPU (sec.)	3.04	3.53	7.57	2.5
individualized shift scheme	Scheme 12*	Scheme 13*	Scheme 14*	
# of iteration	4	7	12	3
optimal obj. value	8.46798	8.46798	8.46798	
Vir. CPU (sec.)	1.76	3.17	5.65	3.2

*: Shift Schemes are defined in Appendix IV.

Table 7 also gives results for "individualized shift scheme". In these cases the selection of the δ and ε used in the shift scheme for each algorithm was selected to minimize the number of iterations. This was based on human experience and a few initial trial runs of the optimization. As the results in Table 7 (and later) indicate, it is possible to reduce computational time still further for all these algorithms by selecting an individualized shift scheme. Note that in this case, Newton requires three times as many iterations as DDP.

The results reported in Table 8 and Table 9 are for the same problem but with longer time periods ($N = 50$ or $N = 100$). Since there are difference between DDP and Newton in each time step, we would expect the CPU time differences to grow with the number of time periods N.

Table 8. Numerical Results for $n = 100,\ m = 10,\ N = 50$ of Test Problem 2

	DDP	Mixed	Newton	Ratio
standardized shift scheme	Scheme 4*	Scheme 7*	Scheme 8*	Newton/DDP
# of iteration	7	11	68	9.7
optimal obj. value	8.49002	8.49002	8.49004	
Vir. CPU (sec.)	17.19	28.43	198.03	11.5
individualized shift scheme	Scheme 12*	Scheme 15*	Scheme 16*	
# of iteration	5	7	61	12.2
optimal obj. value	8.49002	8.49002	8.49005	
Vir. CPU (sec.)	12.54	17.55	171.48	13.7

* : Shift Schemes are defined in Appendix IV.

Table 9. Numerical Results for $n = 100,\ m = 10,\ N = 100$ of Test Problem 2

	DDP	Mixed	Newton	Ratio
standardized shift scheme	Scheme 4*	Scheme 9*	Scheme 10*	Newton/DDP
# of iteration	8	15	132	16.5
optimal obj. value	8.51757	8.51757	8.51759	
Vir. CPU (sec.)	40.45	76.33	811.51	20.1
individualized shift scheme	Scheme 12*	Scheme 17*	Scheme 18*	
# of iteration	5	9	122	24.4
optimal obj. value	8.51757	8.51757	8.51761	
Vir. CPU (sec.)	25.73	46.18	737.57	28.7

* : Shift Schemes are defined in Appendix IV.

Comparison of results in Table 10 supports the expectation that the advantages of DDP over Newton grow as the number of time periods increase. Newton's method takes 16.5 times as many iterations and 20 times as much CPU as DDP for the longest time period ($N = 100$). The difference for $N = 100$ is even greater (ratios = 24.4 and 28.7, respectively) when the individualized shift procedure is used.

Table 10. Convergence Comparison of the Three Algorithms with $n = 100$, $m = 10$ and $N = 10,\ 50$ and 100 for Test Problem 2

	DDP	Mixed	Newton	Ratio
standardized shift scheme	# of iter. to the optimal	# of iter. to the optimal	# of iter. to the optimal	Newton/DDP Iter. (CPU)
$N = 10$	7	8	16	2.3 (2.5)
$N = 50$	7	11	68	9.7 (11.5)
$N = 100$	8	15	132	16.5 (20.1)
individualized shift scheme	# of iter. to the optimal	# of iter. to the optimal	# of iter. to the optimal	
$N = 10$	4	7	12	3 (3.2)
$N = 50$	5	7	61	12.2 (13.7)
$N = 100$	5	9	122	24.4 (28.7)

Notice that as the number of time periods N increases, that the differences between DDP and the Mixed method also grow. DDP required only one more iteration to solve the problem with $N = 100$ than required for $N = 10$. By contrast the number of iterations required to solve the same problem by the Mixed method increased substantially going from $N = 10$ to $N = 100$.

Table 10 indicates that the forward sweep difference is more important than the backward sweep, but that the backward sweep difference also contributes to DDP's success. Recall that the Mixed method has the forward sweep of DDP and the backward sweep of Newton's method. Comparison of the DDP and Mixed method for $N = 100$ indicates that the DDP backward sweep is more efficient than the Newton backward sweep. This difference is related to the use of the extra term $B_t^T C_t^{-1} D_t$ in computing matrices A_t, B_t and C_t ((D.1) – (D.3) and (D.7) versus (N.1) – (N.3) and (N.6)). The value of this term is most important in early iterations since it approaches zero as the algorithm converges. Comparison of the Mixed and Newton methods indicates that the DDP forward sweep is much more effective than the Newton forward sweep for all values of N.

Effects of Starting Values for Highly Nonlinear Problem

Table 11 shows the effect of different starting values on the performance of the DDP versus Newton's method. Table 12 gives the values of the starting point. Starting point 1 is the value used in Tables 7 – 10.

Table 11. Convergence Comparison of the Three Algorithms for Several Starting Values with $n = 100$, $m = 10$, $N = 100$ for Test Problem 2

	Number of iterations to the optimal		
starting point*	DDP	Mixed	Newton
1	8	15	132
2	9	18	127
3	10	16	128
4	10	15	134
5	10	15	134
average	9.4	15.8	131

*: see Table 12.

Table 12. Starting points for Table 11

Starting Points	
1	$(\bar{u}_t)_i = 0.0 \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$
2	$(\bar{u}_t)_i = 0.01 \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$
3	$(\bar{u}_t)_i = -0.01 \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$
4	$\begin{cases} (\bar{u}_t)_i = 0.01 & t \text{ is odd} \\ (\bar{u}_t)_i = 0.0 & t \text{ is even} \end{cases} \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$
5	$\begin{cases} (\bar{u}_t)_i = -0.01 & t \text{ is odd} \\ (\bar{u}_t)_i = 0.0 & t \text{ is even} \end{cases} \quad t = 1, \cdots, N-1, \quad i = 1, \cdots, m$

4. Conclusions

Our theoretical analysis has concluded that the forward sweep in the DDP algorithm is always better than the forward sweep in Newton's method. This is expected since the forward sweep in Newton's method uses only linear approximation for the transition function in updating the control vector. In the backward sweeps, the DDP algorithm uses an extra term $(B_t^T C_t^{-1} D_t)$ in computing matrices A_t, B_t and C_t. This extra term provides a better approximation for the original problem; therefore, it results in both faster convergence and larger convergence region. But, the advantage of this extra term will disappear as the current solution approaches the optimal solution (since $D_t \to 0$).

Pantoja's stagewise Newton's method [11] computes exactly the same search direction as a direct application of Newton's method or as the algorithm in Dunn and Bertsekas [2]. Hence, these other methods would have required the same number of iterations. The CPU per iteration is approximately the same for DDP, Pantoja's stagewise Newton's method [11] and Dunn and Bertsekas' [2] Newton's method. The CPU per iteration is much higher $O(N^3)$ for a direct application of Newton's method. Hence, the iteration results in Table 1 – Table 11 indicate that DDP will be numerically superior to a direct application of Newton's method and to the algorithm in [2] as well as the Pantoja's stagewise Newton's method [11].

Hence, our analytical and numerical investigation indicate that Differential Dynamic Programming is significantly more efficient than Newton-based methods at solving nonlinear optimal control problems. The advantages of DDP over Newton increase as the problem becomes more difficult, that is as the number of time steps becomes larger and as the transition equation becomes more nonlinear. DDP is preferable over Pantoja's stagewise Newton's method in both the forward and backward sweeps. The numerical results indicate the differences in the forward sweep are even more important than the differences in the backward sweep.

Acknowledgements

This work has been supported through two grants from the Advanced Scientific Computing Division of the National Science Foundation (ASC-89153326 and ASC-9211109) to Prof. Shoemaker. This research was conducted using the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which is funded in part by the National Science Foundation, New York State, the IBM Corporation and members of the Center's Corporate Research Institute. We re-coded both the Newton and DDP codes to incorporate sparsity after discussions with Shirish Chinchalkar. Incorporation of sparsity reduces run times for all methods and hence does not affect the ratios of Newton/DDP in Tables 7 – 11.

Appendix I. Standardized Shift Scheme

The following standardized shift scheme is based on the adaptive shift procedure introduced by Liao and Shoemaker [6]. This standardized shift scheme will generate a convergent scheme for solving nonconvex discrete time optimal control problem.

Step 1: Determine the scheme for active shift

Run the algorithm with active shift $\delta = 0.005$, if the algorithm stops due to overflows, go to step 2; otherwise the optimal solution will be found.

Step 2: Determine the scheme for adaptive shift procedure

a) Select a positive value $\varepsilon^c > 0$, run the algorithm with constant shift value ε^c.

b) If the algorithm stops within the next a few iterations (say 5) due to overflow or nonpositive C_t matrix, then set $\varepsilon^c = 2\varepsilon^c$ if ε^c is not large (say less than 100), or set $\varepsilon^c = 5\varepsilon^c$ if ε^c is large, and repeat b); otherwise go to c).

c) Switch from constant shift to active shift with $\delta = 0.005$.

d) If the algorithm stops within the next a few iterations (say 5) due to overflow, go back to constant shift in b) with $\varepsilon^c = \frac{1}{2}\varepsilon^c$ if ε^c is not large (say less than 100), $\varepsilon^c = \frac{1}{5}\varepsilon^c$ if ε^c is large. Otherwise, the optimal solution will be reached.

Appendix II. Analytical Solutions For A Three-Stage OCP

The purpose of this appendix is to illustrate on a simple algebraic example that a) DDP and direct application of Newton's method give different search directions and b) Pantoja's stagewise Newton method gives the same search direction as a direct application of Newton's method. The simple example is a three stage problem with an objective function that depends only upon the third time period.

Problem:

$$\min_{(u_1, u_2)} \; g(x_3) \tag{OCP}$$

where $x_3 = f_2(x_2, u_2)$, $x_2 = f_1(x_1, u_1)$, x_1 is given and fixed,

x_1, x_2, x_3, u_1 and u_2 are all scalars,

Nominal policy and trajectory: $(\bar{u}_1, \bar{u}_2)$ and $(\bar{x}_1, \bar{x}_2, \bar{x}_3)$.

Solution from DDP algorithm

Applying the DDP algorithm in §2.1 to problem (OCP), we have

$$P_3 = g_{xx}, \quad Q_3 = g_x,$$

$$A_2 = [(f_2)_x]^2 \cdot g_{xx} + g_x \cdot (f_2)_{xx},$$

$$B_2 = (f_2)_x \cdot (f_2)_u \cdot g_{xx} + g_x \cdot (f_2)_{xu},$$

$$C_2 = [(f_2)_u]^2 \cdot g_{xx} + g_x \cdot (f_2)_{uu},$$

$$D_2 = (f_2)_u \cdot g_x, \quad E_2 = (f_2)_x \cdot g_x,$$

$$P_2 = A_2 - \frac{(B_2)^2}{C_2}, \quad Q_2 = E_2 - \frac{B_2 \cdot D_2}{C_2},$$

$$\alpha_2 = -\frac{D_2}{C_2}, \quad \beta_2 = -\frac{B_2}{C_2},$$

$$C_1 = [(f_1)_u]^2 \cdot P_2 + Q_2 \cdot (f_1)_{uu}, \quad D_1 = (f_1)_u \cdot Q_2, \quad \alpha_1 = -\frac{D_1}{C_1}.$$

Then, the application of feedback equations (D.11) and (D.12) results in

$$u_1 - \bar{u}_1 = -\frac{(f_1)_u \cdot (E_2 \cdot C_2 - B_2 \cdot D_2)}{[(f_1)_u]^2 \cdot (A_2 \cdot C_2 - (B_2)^2) + (E_2 \cdot C_2 - B_2 \cdot D_2) \cdot (f_1)_{uu}}, \tag{A.2.1}$$

$$x_2 - \bar{x}_2 = f_1(\bar{x}_1, u_1) - f_1(\bar{x}_1, \bar{u}_1), \tag{A.2.2}$$

$$u_2 - \bar{u}_2 = -\frac{D_2 + B_2 \cdot (x_2 - \bar{x}_2)}{C_2}. \tag{A.2.3}$$

Replacing A_2, B_2, C_2, D_2 and E_2 into above expression, we have

$$u_1 - \bar{u}_1 = \frac{(f_1)_u \cdot (g_x)^2 \cdot \Big((f_2)_u \cdot (f_2)_{xu} - (f_2)_x \cdot (f_2)_{uu}\Big)}{W_{DDP}}, \tag{A.2.4}$$

$$u_2 - \bar{u}_2 = \frac{(f_2)_u \cdot g_x + \Big((f_2)_x \cdot (f_2)_u \cdot g_{xx} + g_x \cdot (f_2)_{xu}\Big) \cdot (x_2 - \bar{x}_2)}{[(f_2)_u]^2 \cdot g_{xx} + g_x \cdot (f_2)_{uu}}, \tag{A.2.5}$$

where

$$\begin{aligned} W_{DDP} = & \Big([(f_1)_u]^2 \cdot [(f_2)_x]^2 \cdot g_{xx} + [(f_1)_u]^2 \cdot g_x \cdot (f_2)_{xx} + (f_2)_x \cdot g_x \cdot (f_1)_{uu}\Big) \\ & \cdot \Big([(f_2)_u]^2 \cdot g_{xx} + g_x \cdot (f_2)_{uu}\Big) - [(f_1)_u]^2 \cdot \Big((f_2)_x \cdot (f_2)_u \cdot g_{xx} + g_x \cdot (f_2)_{xu}\Big)^2 \\ & - (f_2)_u \cdot g_x \cdot (f_1)_{uu} \cdot \Big((f_2)_x \cdot (f_2)_u \cdot g_{xx} + g_x \cdot (f_2)_{xu}\Big). \end{aligned} \tag{A.2.6}$$

Newton's Method (direct application)

The original optimal control problem can be transformed into a nonlinear programming in the form of problem (1.3) as follows:

$$\min_{u_1, u_2} F(u_1, u_2)$$

$$\text{where} \quad F(u_1, u_2) = g\Big(f_2\big(f_1(x_1, u_1), u_2\big)\Big).$$

Then, the direct application of Newton's method results in

$$u_1 - \bar{u}_1 = \frac{(f_1)_u \cdot (g_x)^2 \cdot \Big((f_2)_u \cdot (f_2)_{xu} - (f_2)_x \cdot (f_2)_{uu}\Big)}{W_{Newton}}, \tag{A.2.7}$$

$$u_2 - \bar{u}_2 = \frac{(g_x)^2 \cdot \Big([(f_1)_u]^2 \cdot (f_2)_x \cdot (f_2)_{xu} - [(f_1)_u]^2 \cdot (f_2)_u \cdot (f_2)_{xx} - (f_2)_x \cdot (f_2)_u \cdot (f_1)_{uu}\Big)}{W_{Newton}},$$

$$(A.2.8)$$

where

$$W_{Newton} = \Big([(f_1)_u]^2 \cdot [(f_2)_x]^2 \cdot g_{xx} + [(f_1)_u]^2 \cdot g_x \cdot (f_2)_{xx} + (f_2)_x \cdot g_x \cdot (f_1)_{uu}\Big)$$

$$\cdot \Big([(f_2)_u]^2 \cdot g_{xx} + g_x \cdot (f_2)_{uu}\Big) - [(f_1)_u]^2 \cdot \Big((f_2)_x \cdot (f_2)_u \cdot g_{xx} + g_x \cdot (f_2)_{xu}\Big)^2. \quad (A.2.9)$$

Comparison of (A.2.4) – (A.2.6) to (A.2.7) – (A.2.9) proves that there are differences between DDP and Newton's method.

Newton's Method (Pantoja's stagewise Newton's method)

Applying Newton's method in §2.2 to problem (OCP), we have

$$P_3 = g_{xx}, \quad Q_3 = g_x, \quad G_3 = g_x,$$

$$A_2 = [(f_2)_x]^2 \cdot g_{xx} + g_x \cdot (f_2)_{xx},$$

$$B_2 = (f_2)_x \cdot (f_2)_u \cdot g_{xx} + g_x \cdot (f_2)_{xu},$$

$$C_2 = [(f_2)_u]^2 \cdot g_{xx} + g_x \cdot (f_2)_{uu},$$

$$D_2 = (f_2)_u \cdot g_x, \quad E_2 = (f_2)_x \cdot g_x, \quad G_2 = (f_2)_x \cdot g_x,$$

$$P_2 = A_2 - \frac{(B_2)^2}{C_2}, \quad Q_2 = E_2 - \frac{B_2 \cdot D_2}{C_2},$$

$$\alpha_2 = -\frac{D_2}{C_2}, \quad \beta_2 = -\frac{B_2}{C_2},$$

$$C_1 = [(f_1)_u]^2 \cdot P_2 + G_2 \cdot (f_1)_{uu}, \quad D_1 = (f_1)_u \cdot Q_2, \quad \alpha_1 = -\frac{D_1}{C_1}.$$

Then, the application of feedback equations (N.12) and (N.13) results in

$$u_1 - \bar{u}_1 = -\frac{(f_1)_u \cdot (E_2 \cdot C_2 - B_2 \cdot D_2)}{[(f_1)_u]^2 \cdot (A_2 \cdot C_2 - (B_2)^2) + (f_2)_x \cdot g_x \cdot (f_1)_{uu} \cdot C_2}, \quad (A.2.10)$$

$$x_2 - \bar{x}_2 = -\frac{[(f_1)_u]^2 \cdot (E_2 \cdot C_2 - B_2 \cdot D_2)}{[(f_1)_u]^2 \cdot (A_2 \cdot C_2 - (B_2)^2) + (f_2)_x \cdot g_x \cdot (f_1)_{uu} \cdot C_2}, \quad (A.2.11)$$

$$u_2 - \bar{u}_2 = -\frac{D_2}{C_2} + \frac{B_2}{C_2} \cdot \frac{[(f_1)_u]^2 \cdot (E_2 \cdot C_2 - B_2 \cdot D_2)}{[(f_1)_u]^2 \cdot (A_2 \cdot C_2 - (B_2)^2) + (f_2)_x \cdot g_x \cdot (f_1)_{uu} \cdot C_2}. \quad (A.2.12)$$

Replacing A_2, B_2, C_2, D_2 and E_2 into above expression, we have

$$u_1 - \bar{u}_1 = \frac{(f_1)_u \cdot (g_x)^2 \cdot \Big((f_2)_u \cdot (f_2)_{xu} - (f_2)_x \cdot (f_2)_{uu}\Big)}{W_{Newton}}, \tag{A.2.13}$$

$$u_2 - \bar{u}_2 = \frac{(g_x)^2 \cdot \Big([(f_1)_u]^2 \cdot (f_2)_x \cdot (f_2)_{xu} - [(f_1)_u]^2 \cdot (f_2)_u \cdot (f_2)_{xx} - (f_2)_x \cdot (f_2)_u \cdot (f_1)_{uu}\Big)}{W_{Newton}}, \tag{A.2.14}$$

where W_{Newton} is given in (A.2.9).

Comparison of (A.2.13) and (A.2.14) to (A.2.7) – (A.2.9) indicates that the algorithm description we give of the stagewise Newton method in §2.1 does in fact generate exactly the same search direction as the direct application of the Newton's method.

Appendix III. The Mixed Algorithm for Solving OCP

This is an algorithmic description of the Mixed Algorithm that follows the same form as the Newton and DDP algorithms discussed in §2.1 and §2.2.

Step 1: (Initialize parameters and compute loss and trajectory associated with the given policy $\bar{\vec{U}}$)

i) select starting guess $\bar{\vec{U}} = (\bar{u}_1^T, \cdots, \bar{u}_{N-1}^T)^T$ for policy and select $\bar{\theta}$ as stopping criterion for θ.

ii) compute $\bar{x}_{t+1} = f(\bar{x}_t, \bar{u}_t, t)$, recursively for $t = 1, \ldots, N-1$, $\bar{x}_1 \equiv \bar{x}_1$;

iii) compute $J(\bar{\vec{U}}) = g(\bar{x}_N, N)$;

iv) initialize $P_N = g_{xx}$, $Q_N = g_x$, $G_N = g_x$, $\theta_N = 0$.

Step 2: (Perform the backward sweep: Perform steps i) to iv) below, recursively, for $t = N-1, \ldots, 1$)

i) compute A_t, B_t, C_t, D_t and E_t according to

$$A_t = \left[\left(\frac{\partial f}{\partial x} \right)^T P_{t+1} \left(\frac{\partial f}{\partial x} \right) + \sum_{i=1}^{n} (G_{t+1})_i (f_i)_{xx} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{M.1}$$

$$B_t^T = \left[\left(\frac{\partial f}{\partial x} \right)^T P_{t+1} \left(\frac{\partial f}{\partial u} \right) + \sum_{i=1}^{n} (G_{t+1})_i (f_i)_{xu} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{M.2}$$

$$C_t = \left[\left(\frac{\partial f}{\partial u} \right)^T P_{t+1} \left(\frac{\partial f}{\partial u} \right) + \sum_{i=1}^{n} (G_{t+1})_i (f_i)_{uu} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{M.3}$$

$$D_t = \left[\left(\frac{\partial f}{\partial u} \right)^T Q_{t+1} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{M.4}$$

$$E_t = \left[\left(\frac{\partial f}{\partial x} \right)^T Q_{t+1} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{M.5}$$

$$G_t = \left[\left(\frac{\partial f}{\partial x} \right)^T G_{t+1} \right] \Bigg|_{(x=\bar{x}_t, u=\bar{u}_t)}, \tag{M.6}$$

ii) compute P_t, Q_t according to

$$P_t = A_t - B_t^T C_t^{-1} B_t, \tag{M.7}$$

$$Q_t = -B_t^T C_t^{-1} D_t + E_t, \tag{M.8}$$

and store P_t and Q_t in memory, replacing P_{t+1} and Q_{t+1}.

iii) Compute α_t, β_t according to

$$\alpha_t = -C_t^{-1} D_t, \tag{M.9}$$

$$\beta_t = -C_t^{-1} B_t, \tag{M.10}$$

and store in memory.

iv) Compute

$$\theta_t = D_t^T C_t^{-1} D_t + \theta_{t+1}, \tag{M.11}$$

and store θ_t in place of θ_{t+1} in memory.

Step 3: (Perform the forward sweep: compute the successor policy and trajectory)

i) Set $\varepsilon = 1$;

ii) Compute $u_t(\varepsilon)$ and x_t recursively for $t = 1, \ldots, N-1$ according to

$$u_t(\varepsilon) = \bar{u}_t + \varepsilon \cdot \alpha_t + \beta_t \cdot (x_t - \bar{x}_t), \tag{M.12}$$

$$x_{t+1} = f(x_t, u_t(\varepsilon), t), \tag{M.13}$$

iii) Compute $J(\vec{U}(\varepsilon)) = g(x_N, N)$,

iv) If $J(\vec{U}(\varepsilon)) - J(\bar{\vec{U}}) < \varepsilon \dfrac{\theta_1}{2}$,

a) Set $\bar{u}_t = u_t(\varepsilon)$ and $\bar{x}_t = x_t$ for all t,

b) If $\theta_1 \geq \bar{\theta}$, go to step 2 with new values of $(\bar{x}_t, \bar{u}_t)$,
If $\theta_1 < \bar{\theta}$, stopping criterion is satisfied, so stop.

If $J(\vec{U}(\varepsilon)) - J(\bar{\vec{U}}) \geq \varepsilon \dfrac{\theta_1}{2}$, then set $\varepsilon = \varepsilon/2$ and go to Step 3 ii).

Appendix IV. Shift Schemes for the Numerical Runs

The shift schemes used in Section 3 for Test Problem 1 and Test Problem 2 are summarized in the following tables. The procedure for the standardized shift scheme (which is the same procedure for all three algorithms) is given in Appendix I. The "Individualized" Shift Scheme were based on several trial iterations of the optimization and on human experience about how to best select the parameter ε^c and δ. The individualized shift schemes are not necessarily optimal, but they do give better results than the standardized shift scheme.

Scheme 1: Constant shift was used with $\varepsilon^c = 100.0$ for 2 iterations, $\varepsilon^c = 10.0$ for 2 iterations, $\varepsilon^c = 1.0$ for 2 iterations, then active shift with $\delta = 0.005$ was used for the rest of iterations.

Scheme 2: Constant shift was used with $\varepsilon^c = 100.0$ for 2 iterations, $\varepsilon^c = 10.0$ for 4 iterations, then active shift with $\delta = 0.005$ was used for the rest of iterations.

Scheme 3: Constant shift was used with $\varepsilon^c = 1000.0$ for 4 iterations, $\varepsilon^c = 100.0$ for 4 iterations, $\varepsilon^c = 10.0$ for 2 iterations, then active shift with $\delta = 0.005$ for the rest of the iterations.

Table 7a. Shift Schemes for Table 7

Standardized Shift Scheme		Individualized Shift Scheme	
Scheme 4	constant shift $\varepsilon^c = 1.$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 12	constant shift $\varepsilon^c = 1.$ for 2 iter., then, active shift $\delta = 0.005$ for the rest.
Scheme 5	constant shift $\varepsilon^c = 5.$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 13	constant shift $\varepsilon^c = 5.$ for 2 iter., then, active shift $\delta = 0.005$ for the rest.
Scheme 6	constant shift $\varepsilon^c = 5.$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 14	constant shift $\varepsilon^c = 5.$ for 3 iter., $\varepsilon^c = 1.$ for 3 iter., then, active shift $\delta = 0.005$ for the rest.

Table 8a. Shift Schemes for Table 8

Standardized Shift Scheme		Individualized Shift Scheme	
Scheme 4	constant shift $\varepsilon^c = 1.$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 12	constant shift $\varepsilon^c = 1.$ for 2 iter., then, active shift $\delta = 0.005$ for the rest.
Scheme 7	constant shift $\varepsilon^c = 5000.$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 15	constant shift $\varepsilon^c = 5000.$ for 2 iter., $\varepsilon^c = 50.$ for 2 iter., then, active shift $\delta = 0.005$ for the rest.
Scheme 8	constant shift $\varepsilon^c = 5000.$ for 5 iter., $\varepsilon^c = 1000.$ for 5 iter., $\varepsilon^c = 200.$ for 5 iter., $\varepsilon^c = 40.$ for 5 iter., $\varepsilon^c = 20.$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 16	constant shift $\varepsilon^c = 10000.$ for 5 iter., $\varepsilon^c = 1000.$ for 5 iter., $\varepsilon^c = 10.$ for 5 iter., $\varepsilon^c = 1.$ for 10 iter., $\varepsilon^c = 0.5.$ for 10 iter., then, active shift $\delta = 0.005$ for the rest.

Table 9a. Shift Schemes for Table 9

Standardized Shift Scheme		Individualized Shift Scheme	
Scheme 4	constant shift $\varepsilon^c = 1.$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 12	constant shift $\varepsilon^c = 1.$ for 2 iter., then, active shift $\delta = 0.005$ for the rest.
Scheme 9	constant shift $\varepsilon^c = 50000.$ for 5 iter., $\varepsilon^c = 10000.$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 17	constant shift $\varepsilon^c = 50000.$ for 2 iter., $\varepsilon^c = 1000.$ for 2 iter., then, active shift $\delta = 0.005$ for the rest.
Scheme 10	constant shift $\varepsilon^c = 50000.$ for 5 iter., $\varepsilon^c = 10000.$ for 5 iter., $\varepsilon^c = 2000.$ for 5 iter., $\varepsilon^c = 400.$ for 5 iter., $\varepsilon^c = 80.$ for 5 iter., $\varepsilon^c = 40.$ for 5 iter., $\varepsilon^c = 20.$ for 5 iter., $\varepsilon^c = 10.$ for 5 iter., $\varepsilon^c = 5.$ for 5 iter., $\varepsilon^c = 2.5$ for 5 iter., then, active shift $\delta = 0.005$ for the rest.	Scheme 18	constant shift $\varepsilon^c = 100000.$ for 5 iter., $\varepsilon^c = 10000.$ for 10 iter., $\varepsilon^c = 1000.$ for 10 iter., $\varepsilon^c = 100.$ for 10 iter., $\varepsilon^c = 10.$ for 10 iter., $\varepsilon^c = 1.$ for 10 iter., $\varepsilon^c = .5$ for 10 iter., then, active shift $\delta = 0.005$ for the rest.

References

[1] J. C. Dunn, *Gradient projection methods for systems optimization problems*, Advances in Control and Dynamic, (C. T. Leondes, ed.), London, Academic Press, 1988.

[2] J. Dunn and D. P. Bertsekas, *Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems*, JOTA, Vol. 63 (1989), pp. 23-38.

[3] D. Jacobson *Second-order and second-variation methods for determining optimal control: a comparative study using differential dynamic programming*, Int. J. Control, Vol. 7 (1968), pp. 175-196.

[4] D. Jacobson and D. Mayne, *Differential Dynamic Programming*, Elsevier Sci. Publ., New York, 1970.

[5] L.-Z. Liao and C. A. Shoemaker, *The proof of quadratic convergence of differential dynamic programming*, Technical Report No. 917, School of Operations Research and Industrial Engineering, Cornell University, August, 1990.

[6] L.-Z. Liao and C. A. Shoemaker, *Convergence in unconstrained discrete-time differential dynamic programming*, IEEE Trans. Automat. Contr., Vol. 36, No. 6 (1991), pp. 692-706.

[7] D. Mayne, *A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems*, Intnl. J. Control, Vol. 3 (1966), pp. 85-95.

[8] S. K. Mitter, *Successive approximation methods for the solution of optimal control problems*, Automatica, Vol. 3 (1966), pp. 135-149.

[9] D. M. Murray and S. Yakowitz, *Differential dynamic programming and Newton's method for discrete optimal control problems*, JOTA, Vol. 43 (1984), pp. 395-414.

[10] K. Ohno, *A new approach of differential dynamic programming for discrete time systems*, IEEE Trans. Automat. Contr., Vol. 23 (1978), pp. 37-47.

[11] J. F. A. De O. Pantoja, *Differential dynamic programming and Newton's method*, Int. J. Control, Vol. 47 (1988), pp. 1539-1553.

[12] W. L. Voland, *Control Systems Modeling and Analysis*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1986.

[13] S. J. Wright, *Solution of discrete-time optimal control problems on parallel computers*, Parallel Computing, Vol. 16 (1990), pp. 221-237.

[14] S. J. Wright, *Partitioned dynamic programming for optimal control*, SIAM J. Opt., Vol. 1 (1991), No. 4, pp. 620-642.

[15] S. Yakowitz, *The Stagewise Kuhn-Tucker Condition and Differential Dynamic Programming*, IEEE Trans. Automat. Contr., Vol. 31 (1986), pp. 25-30.

[16] S. Yakowitz, *Theoretical and computational advances in differential dynamic programming*, Control and Cybernetics, Vol. 17 (1988), No. 2-3, pp. 173-189.

[17] S. Yakowitz and B. Rutherford, *Computational aspects of discrete-time optimal control*, Appl. Math. Comput., Vol. 15 (1984), pp. 29-45.