Some efficient algorithms for unconstrained discrete-time optimal control problems

Aiping Liao[1]

Advanced Computing Research Institute
Cornell Theory Center
Cornell University
Ithaca, NY 14853

October 26 1993
Revised November 4 1993

Abstract

The differential dynamic programming algorithm (DDP) and the stagewise Newton procedure are two typical examples of efficient local procedures for discrete-time optimal control (DTOC) problems. It is desirable to generalize these local procedures to globally convergent methods. One successful globalization was recently proposed by Coleman and Liao [3] which combines the trust region idea with Pantoja's stagewise Newton procedure.

In this paper we propose several algorithms for DTOC problems which combine a modified "dogleg" algorithm with DDP or Pantoja's Newton procedure. These algorithms possess advantages of both the dogleg algorithm and the DDP or the stagewise procedure, i.e., they have strong global and local convergence properties yet remain economical. Numerical results are presented to compare these algorithms and the Coleman-Liao algorithm.

[1]This research was partially supported by the Cornell Theory Center, which receives major funding from the National Science Foundation and IBM Corporation, with additional support from the State of New York and members of its Corporate Research Institute.

1 Introduction

Discrete-time optimal control (DTOC) problems are large-scale optimization problems with a dynamic structure and represent a large class of practical problems. Some examples are: multi-reservoir control problems [14], the treatment of polluted groundwater [5], inventory control [4], and aircraft control [12]. These many applications, especially present-day large-scale problems, stimulate research for efficient algorithms for DTOC problems.

In this work we are concerned with the unconstrained discrete-time optimal control problem:

$$\begin{array}{ll} \min F := \sum_{i=1}^{N-1} L_i(y_i, x_i) + L_N(y_N) & \\ y_{i+1} = T_i(y_i, x_i), \quad i = 1, \ldots, N-1 & \qquad (\mathrm{P}) \\ y_1 = \bar{y}_1. & \end{array}$$

The vectors $y_i \in R^{n_y}, i = 1, \ldots, N$ are called state variables and the vectors $x_i \in R^{n_x}, i = 1, \ldots, N-1$ are called control variables, $\bar{y}_1$ is a constant vector.

We are aware of two efficient local procedures for (P): the differential dynamic programming algorithm (DDP) [11] and the stagewise Newton procedure [16]. Both of these procedures possess a locally quadratically convergent rate and require only $O(N)$ operations per iteration. It is thus desirable to generalize these local procedures to globally convergent methods. One successful globalization was recently proposed by Coleman and Liao [3] which combines the trust region idea with Pantoja's stagewise Newton procedure. Their algorithm is quite satisfactory both theoretically and numerically. In this paper, based on the techniques developed in [3], we propose a global algorithm framework that combines a modified dogleg algorithm with the stagewise Newton procedure or the DDP algorithm. The proposed algorithms are conceptually simple and need fewer calculations per iteration than that of Coleman and Liao [3] if n_y is large.

Throughout this paper, we are mainly concerned with the following DTOC form which is equivalent to (P) (see Coleman and Liao [3] or Pantoja [16] for details):

$$\begin{array}{ll} \min F := L_N(y_N) & \\ y_{i+1} = T_i(y_i, x_i), \quad i = 1, \ldots, N-1 & \qquad (\mathrm{PP}) \\ y_1 = \bar{y}_1. & \end{array}$$

We denote $y = (y_1^T, \ldots, y_N^T)^T \in R^{n_y N}$ and $x = (x_1^T, \ldots, x_{N-1}^T)^T \in R^{n_x(N-1)}$. We also denote $n = n_x(N-1)$, y_i the i-th state variable, x_i the i-th control variable, $y_{i,j}$ the j-th component of the i-th state variable, $x_{i,j}$ the j-th component of the i-th control variable and $T_{i,j}$ the j-th component of the i-th transition function. We assume that all functions are twice continuously differentiable. As pointed out in Coleman and Liao [3], form (PP) helps illuminate the connection between the Pantoja's stagewise Newton procedure and Newton method. It also serves as a basis for comparison between the stagewise Newton procedure and the DDP algorithm.

We note that some preliminary comparisons between the stagewise Newton procedure and the DDP algorithm are made in [10]. However, we feel that more extensive and precise comparisons are desired for our purpose since we are concerned with only the search directions induced by the DDP algorithm and the stagewise Newton procure. We will empirically compare the various algorithms on a suite of test problems.

2 Description of the algorithms

In this section we describe our algorithms which combine the DDP algorithm or the stagewise Newton's method of Pantoja [16] with a modified dogleg method, similar to the dogleg algorithm of Powell [17] (see also Moré [13]). We first discuss the two efficient local procedures – the stagewise Newton's method of Pantoja [16] and the DDP algorithm (Mayne [11] and Jacobson and Mayne [9]).

2.1 The DDP algorithm and the stagewise Newton method

The DDP method was originally proposed by Mayne [11] and Jacobson and Mayne [9]. This kind of method, making use of the dynamic structure of (PP), combines Bellman's optimality principle [2] and Newton's method. It has been proved, see Pantoja [16] and Murray and Yakowitz [14] for example, that this algorithm is locally quadratically convergent.

The following is a single step of the DDP algorithm for (PP) (see, for example, Liao and Shoemaker [10], Yakowitz and Rutherford [19]). We assume that C_i is positive definite for all $i = 1, \ldots, N-1$.

Procedure 1 [a single step of the DDP algorithm]

Step 1. Given the current control variable x; calculate the current state variable y via the transition function T; calculate $P = (L_N)_{y_N y_N}, Q = (L_N)_{y_N}$.

Step 2. Perform backward recursion:

For $i = N-1, \ldots, 1$ **do**

(i). Calculate A_i, B_i, C_i, D_i and E_i according to

$$\begin{aligned}
A_i &= (\frac{\partial T_i}{\partial y_i})^T P(\frac{\partial T_i}{\partial y_i}) + \sum_{j=1}^{n_y} Q_j (T_{i,j})_{y_i y_i} \\
B_i &= (\frac{\partial T_i}{\partial y_i})^T P(\frac{\partial T_i}{\partial x_i}) + \sum_{j=1}^{n_y} Q_j (T_{i,j})_{y_i x_i} \\
C_i &= (\frac{\partial T_i}{\partial x_i})^T P(\frac{\partial T_i}{\partial x_i}) + \sum_{j=1}^{n_y} Q_j (T_{i,j})_{x_i x_i} \\
D_i &= (\frac{\partial T_i}{\partial x_i})^T Q \\
E_i &= (\frac{\partial T_i}{\partial y_i})^T Q.
\end{aligned}$$

(ii). Calculate:

$$\begin{aligned}
\alpha_i &= -C_i^{-1} D_i \\
\beta_i &= -C_i^{-1} B_i^T.
\end{aligned}$$

(iii). Update P and Q:

$$\begin{aligned}
P &\leftarrow A_i - \beta_i^T C_i \beta_i \\
Q &\leftarrow E_i + B_i \alpha_i.
\end{aligned}$$

End

Step 3. Calculate the next iterate – $\bar{x}$.

For $i = 1, \ldots, N-1$ **do**

$$\begin{aligned}
\bar{x}_i &= x_i + \alpha_i + \beta_i(\bar{y}_i - y_i) \\
\bar{y}_{i+1} &= T_i(\bar{y}_i, \bar{x}_i).
\end{aligned}$$

End

For convenience, we denote the "search direction" induced by one single iteration of the DDP algorithm by $d_{ddp} := \bar{x} - x = DDP(x, \{C_i\}, L, T)$.

Pantoja [16] proposes a modified DDP method for solving (PP). This method produces iterates identical to those that Newton's method would produce for problem $(\bar{P})$,

$$\begin{array}{l} \min f(x) \\ x \in R^n \end{array} \qquad (\bar{P})$$

which is obtained by eliminating the state variables in (PP). Pantoja's algorithm is regarded as a stagewise Newton's method. Unlike the conventional Newton's method for $(\bar{P})$ which requires $\Omega(N^3)$ works per iteration[1] Pantoja's algorithm needs only $O(N)$ operations per iteration. The analysis of this algorithm is addressed in Pantoja [16] and Coleman and Liao [3].

The following is a single step of Pantoja's algorithm for (PP). The Newton direction $d = -H^{-1}g$ is calculated, where H is the Hessian of $f(x)$ (we assume that H is invertible).

Procedure 2 [a single step of Pantoja's stagewise Newton's method [16]]

Step 1. Given the current control variable x; calculate the current state variable y via the transition function T; calculate $P = (L_N)_{y_N y_N}, Q = (L_N)_{y_N}, G = Q$.

Step 2. Perform backward recursion:

For $i = N-1, \ldots, 1$ **do**

(i). Calculate A_i, B_i, C_i, D_i and E_i according to

$$\begin{aligned}
A_i &= (\frac{\partial T_i}{\partial y_i})^T P (\frac{\partial T_i}{\partial y_i}) + \sum_{j=1}^{n_y} G_j (T_{i,j})_{y_i y_i} \\
B_i &= (\frac{\partial T_i}{\partial y_i})^T P (\frac{\partial T_i}{\partial x_i}) + \sum_{j=1}^{n_y} G_j (T_{i,j})_{y_i x_i} \\
C_i &= (\frac{\partial T_i}{\partial x_i})^T P (\frac{\partial T_i}{\partial x_i}) + \sum_{j=1}^{n_y} G_j (T_{i,j})_{x_i x_i} \\
D_i &= (\frac{\partial T_i}{\partial x_i})^T Q \\
E_i &= (\frac{\partial T_i}{\partial y_i})^T Q.
\end{aligned}$$

[1] We denote by $p = \Omega(q)$ if there are $0 < c_1 < c_2$ such that $c_1 q < p < c_2 q$.

(ii). Calculate:

$$\begin{aligned}\alpha_i &= -C_i^{-1}D_i \\ \beta_i &= -C_i^{-1}B_i^T.\end{aligned}$$

(iii). Update P, Q and G:

$$\begin{aligned}P &\leftarrow A_i - \beta_i^T C_i \beta_i \\ Q &\leftarrow E_i + B_i\alpha_i \\ G &\leftarrow (\frac{\partial T_i'}{\partial y_i'})^T G.\end{aligned}$$

End

Step 3. Calculate the Newton direction. Let $(\delta y)_1 = 0$.

For $i = 1, \ldots, N-1$ **do**

$$\begin{aligned}d_i &= \alpha_i + \beta_i(\delta y)_i \\ (\delta y)_{i+1} &= \frac{\partial T_i}{\partial y_i}(\delta y)_i + \frac{\partial T_i}{\partial x_i} d_i.\end{aligned}$$

End

For convenience, we denote the output of Procedure 2 by $d_n = newton(x, L, T)$. A relationship between these two procedures is stated below.

Lemma 2.1 *If the sequence $\{x^k\}$ generated by the DDP algorithm converges to x^* at which H is positive definite, then*

$$\frac{\|d_{ddp}^k - d_n^k\|}{\|d_{ddp}^k\|} \longrightarrow 0, \tag{1}$$

where d_n^k is the Newton's direction.

Proof. We first note that for any two non-zero vectors $x, y \in R^n$ there is a nonsingular matrix B such that $x = By$. Thus the iterates generated by the DDP algorithm can be regarded as $B^k d_{ddp}^k = -g^k$ for some nonsingular matrix. Since the DDP algorithm is quadratically convergent, the theorem follows by Theorem 3.1 of Dennis and Moré [7]. □

We note that (1) shows that d_{ddp}^k approaches d_n^k in both length and direction.

Both Procedure 1 and Procedure 2 have the same computational complexity (we ignore the lower order terms) [3]:

$$N \cdot (2n_y^3 + \frac{7}{2}n_y^2 n_x + 2n_y n_x^2 + \frac{1}{3}n_x^3), \tag{2}$$

though in Procedure 2 an n_y-vector G needs to be calculated which is not required in Procedure 1. There appears to be two distinct ways to program these algorithms:

Prog. 1. During each iteration (including both the backward recursion and the forward recursion) we calculate all the relevant quantities and store them for use within the current iteration.

Prog. 2. Relevant quantities are calculated and stored only within the subiterations of both the backward recursion and the forward recursion.

Prog. 1 is the usual method, it avoids re-computations. However, for DTOC problems, it is more difficult to program and requires much more storage than Prog. 2. If Prog. 2 is used for Pantoja's stagewise Newton's method, then the quantities such as $\frac{\partial T_i}{\partial y_i}$ and $\frac{\partial T_i}{\partial x_i}$ will be re-computed in the forward recursion (Step 3). No recomputations will be needed if Prog. 2 is used for the DDP algorithm. If we are not concerned about storage requirement, Procedure 2 should be programmed in Prog. 1 and we have the following remarks:

Remark 1. Procedure 1 needs more function evaluations than Procedure 2, since in Step 3 of Procedure 1 the transition function needs to be evaluated for every stage $i = 1, \ldots, N$ while it is not required for Procedure 2.

Remark 2. If n_y is not large and N is large or the evaluation of the transition function is not easy, then the re-evaluation of the transition function in Procedure 1 may require more operations than that of the calculation caused by the extra matrix G in Procedure 2. Thus, in this case, Procedure 2 has some advantage over Procedure 1.

Remark 3. If n_y is large and calculations of the transition functions are easy, then the calculation caused by the extra matrix G in Procedure 2 might be overweight and the DDP algorithm might have advantage over the stagewise Newton method.

2.2 A modified dogleg algorithm

The dogleg algorithm can be viewed as a trust region method for the unconstrained minimization problem of the form

$$\min_{x \in R^n} f(x).$$

A trust region method calculates a trial step by (approximately) solving the subproblem

$$\begin{array}{ll} \min \phi_k(d) := (g^k)^T d + \frac{1}{2} d^T H_k d \\ \text{subject to} \quad \|d\| \le \Delta_k, \end{array} \qquad \text{(subP)}$$

where $g^k = \nabla f(x^k)$, H_k is an $n \times n$ symmetric matrix which is either the Hessian of f or some approximation to it and $\Delta_k > 0$ is a trust region radius. (Throughout this paper we assume that H is the Hessian of f and $\|\cdot\|$ is the l_2 norm.) Then, based on the ratio between the actual reduction in the function and the predicted reduction, the step d^k –the solution of (subP)– is either accepted or rejected and H_k and Δ_k are updated.

We propose below a modified dogleg algorithm with line search:

Algorithm 1

Initialization. Given x^1 and $\Delta_1 > 0$, choose c_1, c_2 such that $0 < c_1 < 1$ and $0 < c_2 < 1$. Set $k = 1$.

Until convergence do

(i). Find the Cauchy point d_c:

$$d_c = \begin{cases} -\frac{g^T g}{g^T H g} g & \text{if } g^T H g > 0 \text{ and } \|d_c\| < \Delta, \\ -\frac{\Delta}{\|g\|} g & \text{o.w.} \end{cases}$$

(ii). Take $\bar{d}^k = d_l^k$ where d_l^k is the trial step of any locally quadratically convergent algorithm, for example, we can take $d_l^k = d_n^k = newton(x^k, LT)$ or $d_l^k = d_{ddp}^k = DDP(x^k, \{C_i\}, L, T)$.

(iii). If $\|\bar{d}^k\| \le \Delta_k$ and $\phi_k(\bar{d}^k) \le \beta \phi_k(d_c)$, where $\beta < 1$ is a positive constant, then set $d^k = \bar{d}^k$; otherwise set $d^k = d_c$.

(iii). Calculate $f(x^k + d^k)$.

If $f(x^k + d^k) \geq f(x^k)$

use a simple binary search to find $0 < \tau_k < 1$ such that $f(x^k + \tau_k d^k) < f(x^k)$ and put $x^{k+1} = x^k + \tau_k d^k$, $\Delta_{k+1} = \|x^{k+1} - x^k\|$;

else

set $x^{k+1} = x^k + d^k$ and

$$\Delta_{k+1} = \begin{cases} \Delta_k & \text{if } \rho_k \geq c_1 \\ c_2 \Delta_k & \text{o.w.} \end{cases}$$

where

$$\rho_k = \frac{f(x^k) - f(x^{k+1})}{\phi_k(0) - \phi_k(d^k)}.$$

End if

Calculate g^{k+1}; set $k = k + 1$.

End

We note that the method for updating the trust region radius Δ_k in Algorithm 1 is a simplified one. A more sophisticated update method can be found in Nocedal and Yuan [15].

All the quantities in the above algorithm can be calculated in $O(N)$ flops using the techniques developed in Coleman and Liao [3]. For example, $g^T H g$ can be obtained by first calculating $t_1 = g^T g + \frac{1}{2} g^T H_1 g$ and $t_2 = \frac{1}{2} g^T H_2 g$, where $H = H_1 + H_2$ is a partition of H, using two sub-DTOC problems and then taking $g^T H g = 2(t_1 + t_2 - g^T g)$.

In Theorem 2.4 below we show that Algorithm 1 has the same convergence properties as the dogleg algorithm of Powell [17]. We first give a characterization of quadratic convergence and state a technical result.

Lemma 2.2 *Suppose* $\{x^k\}$ *converges to* x^* *at which* $g(x^*) = 0$ *and* $H^* = H(x^*)$ *is positive definite. Then there is a constant* μ_1 *such that, for all large* k,

$$\|x^{k+1} - x^*\| \leq \mu_1 \|x^k - x^*\|^2 \tag{3}$$

if and only if there is a constant μ_2 *such that, for all large* k,

$$\|d^k - d_n^k\| \leq \mu_2 \|d_n^k\|^2 \tag{4}$$

where $d^k = x^{k+1} - x^k$.

Proof. The proof is similar to that of Theorem 2.2 of Dennis and Moré [6]. We first note that, same as in the proof of Lemma 2.1, we can assume there is a sequence of nonsingular matrices $\{B^k\}$ such that $d^k = -(B^k)^{-1}g^k$. We will show that (3) is equivalent to

$$\|[B^k - H^*](x^{k+1} - x^k)\| \leq \mu_3 \|x^{k+1} - x^k\|^2, \tag{5}$$

for all sufficiently large k and some μ_3, which in turn is obviously equivalent to (4) by Lemma 3.2 of Dennis and Moré [7]. Assume first that (5) holds. Since

$$[B^k - H^*](x^{k+1} - x^k) = g^{k+1} - g^k - H^*(x^{k+1} - x^k) - g^{k+1}, \tag{6}$$

the continuity of H at x^* and (4) imply that

$$\|g^{k+1}\| \leq \gamma \|x^{k+1} - x^k\|^2, \tag{7}$$

for large k, where γ is a constant. Since H^* is nonsingular, there is a $\beta > 0$ such that

$$\|g^{k+1}\| = \|g^{k+1} - g^*\| \geq \beta \|x^{k+1} - x^*\|.$$

On the other hand, by Theorem 2.2 and Lemma 2.1 of Dennis and Moré [6], we have

$$\lim_{k\to\infty} \frac{\|x^{k+1} - x^k\|}{\|x^k - x^*\|} = 1.$$

Therefore, noting that

$$\frac{\|g^{k+1}\|}{\|x^{k+1} - x^k\|^2} \geq \frac{\beta\|x^{k+1} - x^*\|}{\|x^k - x^*\|^2} \cdot \frac{\|x^k - x^*\|^2}{\|x^{k+1} - x^k\|^2}$$

(3) thus holds for $\mu_1 > \gamma/\beta$.

Conversely, assume that $\{x^k\}$ converges quadratically to x^* i.e., (3) holds. Since

$$\frac{\|g^{k+1}\|}{\|x^{k+1} - x^k\|^2} = \frac{\|g^{k+1} - g^*\|}{\|x^k - x^*\|^2} \cdot \frac{\|x^k - x^*\|^2}{\|x^{k+1} - x^k\|^2},$$

Lemma 2.1 of [6] and the continuity of H imply that, for large k, (7) holds for $\gamma > \beta\mu_1$. It then follows from (6) that there is μ_3 such that (5) holds for all large k. The proof is thus complete. □

The following is a variant of the Cauchy-Schwarz inequality, called Bergstrom's inequality, which can be found in Beckenbach and Bellman [1].

Lemma 2.3 *If the matrix A is positive definite, then*

$$x^T Ax \cdot y^T A^{-1} y \geq (x^T y)^2$$

for all x and y, and equality holds if and only if Ax and y are parallel.

$\square$

Theorem 2.4 *Let $f : R^n \to R$ be twice continuously differentiable and bounded below on R^n. If $\{x^k\}$ is the sequence generated by Algorithm 1 then we have*

$$\lim_{k\to\infty} \inf \|g^k\| = 0.$$

Moreover, suppose that $\nabla^2 f$ is bounded on the level set $\{x \in R^n : f(x) \leq f(x^0)\}$ and $\{x^k\}$ is the sequence generated by Algorithm 1. If x^* is a limit point of $\{x^k\}$ with $\nabla^2 f(x^*)$ positive definite then $\{x^k\}$ converges to x^* superlinearly.

Proof. Since the trial step d^k generated in Algorithm 1 always satisfies

$$\phi_k(d^k) \leq \beta \min\{\phi_k(d) : d = \mu g^k, \|d\| \leq \Delta_k\}, \quad \|d^k\| \leq \Delta_k,$$

it follows from Lemma (4.8) and Theorem (4.10) of Moré [13] that

$$\lim_{k\to\infty} \inf \|g^k\| = 0.$$

If x^* is a limit point of $\{x^k\}$ with $\nabla^2 f(x^*)$ positive definite then, by Theorem (4.19) of Moré [13], $\{x^k\}$ converges to x^* and $\{\Delta_k\}$ is bounded away from zero. We now claim that $d^k = d_l^k$ for all sufficiently large k. It follows from Lemma 2.3 that, for sufficiently large k,

$$\begin{aligned}
\phi_k(d_n) &= -g^T H^{-1} g + \frac{1}{2} g^T H^{-1} g \\
&= -\frac{1}{2} g^T H^{-1} g \\
&\leq -\frac{1}{2} \frac{\|g\|^4}{g^T H g} \\
&= \phi_k(d_c). \qquad (8)
\end{aligned}$$

On the other hand, by Lemma 2.2, for sufficiently large k we have

$$\begin{aligned}\phi_k(d_l) &= g^T d_l + \frac{1}{2} d_l^T H d_l \\ &= g^T d_n + \frac{1}{2} d_n^T H d_n + E \\ &= \phi_k(d_n) + E, \end{aligned} \tag{9}$$

where $\|E\| = O(\|g\|^3)$. Thus, noting that $0 < \beta < 1$, it follows from (8) and (9) that, for sufficiently large k:

$$\phi_k(d_l) \leq \beta \min\{\phi_k(d) : d = \mu g^k, \|d\| \leq \Delta_k\}, \quad \|d^k\| \leq \Delta_k.$$

The claim is thus true and the theorem follows since Algorithm 1 reduces to the locally quadratically convergent algorithm for large k. □

2.3 The proposed algorithms

Our algorithms are in the framework of Algorithm 1, the modified dogleg algorithm, but with different choices for the search direction $\bar{d}$.

1. **Dogleg-NT**. In Algorithm 1 define $\bar{d}$ as follows: $\bar{d} = newton(x, L, T)$ if all C_i are nonsingular and $\bar{d} = -g$ otherwise.

2. **Dogleg-DDP1**. In Algorithm 1 define $\bar{d}$ as follows: $\bar{d} = DDP(x, \{\bar{C}_i\}, L, T)$ where $\bar{C}_i = C_i + \kappa I$ and $\kappa = 0$ if $\lambda_s(C_i) \geq \delta$ and $\kappa = \delta - \lambda_s(C_i)$ otherwise. Here $\lambda_s(C_i)$ denotes the minimum eigenvalue of C_i.

3. **Dogleg-DDP2**. In Algorithm 1 define $\bar{d}$ as follows: $\bar{d} = DDP(x, \{C_i\}, L, T)$ if all C_i are positive definite and $\bar{d} = -g$ otherwise.

Obviously, Theorem 2.4 holds for all these variants.

We note that in Algorithm 1 d^k is chosen from a set of two points: $\{d_c, d_l\}$. Sometimes this leads to a slow convergence; for example, our numerical tests show that when the Hessian of f at the solution, H^*, is singular Algorithm 1 may converge very slowly. For improving convergence, in (iii) of Algorithm 1 we choose d^k to be the (approximate) minimizer on a one-dimensional set:

$$\min\{\phi(d) : d = d_c + \gamma(d_l - d_c), \ \|d\| \leq \Delta\}. \tag{10}$$

This problem can be solved using the following procedure (iii-a)–(iii-c) which we use to replace (iii) of Algorithm 1 to calculate d:

(iii-a). Define $d(\gamma) := d_c + \gamma(d_l - d_c)$. Then set $\|d(\gamma)\| \leq \Delta$ whose solution can be expressed as $\gamma_L \leq \gamma \leq \gamma_U$ where

$$\begin{aligned}\gamma_L &= \frac{-d_c^T(d_l - d_c) - \sqrt{(d_c^T(d_l-d_c))^2 + \|d_l - d_c\|^2(\Delta^2 - \|d_c\|^2)}}{\|d_l - d_c\|^2} \\ \gamma_U &= \frac{-d_c^T(d_l - d_c) + \sqrt{(d_c^T(d_l-d_c))^2 + \|d_l - d_c\|^2(\Delta^2 - \|d_c\|^2)}}{\|d_l - d_c\|^2}.\end{aligned}$$

Let

$$\gamma_{min} = -\frac{g^T(d_l - d_c) + d_c^T H(d_l - d_c)}{(d_l - d_c)^T H(d_l - d_c)}.$$

(iii-b). Determine the best γ, $\gamma^* = \arg\min\{\phi(d(\gamma)) : \gamma_L \leq \gamma \leq \gamma_U\}$:

- If $(d_l - d_c)^T H(d_l - d_c) > 0$:

$$\gamma^* = \begin{cases} \gamma_{min} & \text{if } \gamma_{min} \in [\gamma_L, \gamma_U] \\ \gamma_L & \text{if } \gamma_{min} < \gamma_L \\ \gamma_U & \text{if } \gamma_{min} > \gamma_U. \end{cases}$$

- If $(d_l - d_c)^T H(d_l - d_c) < 0$:

$$\gamma^* = \begin{cases} \gamma_U & \text{if } |\gamma_{min} - \gamma_L| < |\gamma_{min} - \gamma_U| \\ \gamma_L & \text{otherwise.} \end{cases}$$

- If $(d_l - d_c)^T H(d_l - d_c) = 0$:

$$\gamma^* = \begin{cases} \gamma_L & \text{if } g^T(d_l - d_c) + d_c^T H(d_l - d_c) \geq 0 \\ \gamma_U & \text{otherwise.} \end{cases}$$

(iii-c). Let $d = d(\gamma^*)$.

We note that in the above procedure $g^T(d_l - d_c) + d^T H(d_l - d_c)$ needs to be calculated. But it seems there is no efficient way to carry out this calculation in general. However, if the Pantoja's procedure is used, i.e., $d_l = d_n = -H^{-1}g$, then

$$g^T(d_l - d_c) + d^T H(d_l - d_c) = -(d_l - d_c)^T H(d_l - d_c), \tag{11}$$

thus $\gamma_{min} = 1$ the above procedure can be carried out efficiently. If the DDP algorithm is used, $d_l = d_{ddp}$ does not satisfy (11). We note that (11) asymptotically holds as $x^k \to x^*$. Therefore, for simplification of computations, we suppose that (11) holds for $d_l = d_{ddp}$ in the above procedure. However the d calculated in this way is only an approximate solution to (10). To guarantee the global convergence, if $\phi_k(d) \leq \beta\phi_k(d_c)$ does not hold, we simply take $d = d_c$.

3 Numerical results

We tested our algorithms with the problems collected in the Appendix of Coleman and Liao [3]. Our algorithms were written in MATLAB and all runs were performed on a SUN Sparcstation. We take $c_1 = 0.1$, $c_2 = 0.5$ and $\beta = 0.9$ in Algorithm 1 and $\delta = 0.01$ in Dogleg-DDP1. The line search algorithm in Algorithm 1 is carried out so that

$$f(x^{k+1}) \leq f(x^k) + 0.0001(x^{k+1} - x^k)^T g^k.$$

For algorithm Dogleg-NT we use the MATLAB function "**rcond**" to check if C_i is singular. For algorithm Dogleg-DDP1, the minimum eigenvalue of C_i is calculated as: $\lambda_s(C_i) = \mathbf{min}(\mathbf{eig}(C_i))$, i.e., we first calculate all the eigenvalues of C_i and then choose the smallest one. We note that there are more efficient ways to calculate the the minimum eigenvalue of a matrix, see Golub and Van Loan [8] for example. For algorithm Dogleg-DDP2, we use the Cholesky factorization to check if C_i is positive definite. The convergence criterion is $\|g\| < 10^{-6}$. For comparison we also include the algorithm proposed in Coleman and Liao [3], the TR-NY algorithm. The numerical results are presented in Table 1-4. We note that the numbers in the column "Feva" are the numbers of objective function evaluations.

Our numerical tests show that, at least for these test problems, the proposed algorithms, including TR-NY, perform quite satisfactory, though Dogleg-DDP1 fails for Problem 4 because some of the matrices C_i are too ill-conditioned. (It is interesting to note that 3 of 5 parallel algorithms of Wright [18] also failed for this test problem.) Among these algorithms the TR-NY seems to be the best followed by Dogleg-NT, Dogleg-DDP. However, we note that the values of n_y in these tests are small. Since, after d_l being calculated, the main calculation for obtaining the trial step d^k for TR-NY algorithm needs $O(N(n_x + n_y)^3)$ flops while it is $O(N(n_y n_x^2))$ for Dogleg-NT (see [3]), thus if n_y is large Dogleg-NT and Dogleg-DDP should have some advantage over TR-NY. Table 5 presents the numerical results for Test Problem 1 (with $\mu = 1$, $N = 10$, $n_x = 2$ and a "large" $n_y = 10$) which shows that the number of flops per iteration for Dogleg-NT is less than that for TR-NY. We also note that algorithm Dogleg-DDP needs more flops than Dogleg-NT per iteration. The reason is mainly due to the fact that the DDP algorithm needs more function evaluations than Pantoja's Newton procedure (see Remark 1 in Section 2.1). For example, for Problem 1 with $n_y = 4$, $n_x = 2$, $\mu = 1$, and $N = 50$ the total number of flops for Dogleg-NT is 715716 of which 139577 flops are spent on function evaluations (including objective function evaluation

Table 1: Numerical results for problem 1 ($n_y = 4, n_x = 2$)

μ	N	method	Iter	Feva	Flops(flops on fun. evaluation)
0	10	TR-NY	10	10	176066(25320)
		Dogleg-NT	10	10	156059(25320)
		Dogleg-ddp1	10	10	184386(42188)
		Dogleg-ddp2	10	10	183535(42188)
	50	TR-NY	9	9	980127(116115)
		Dogleg-NT	10	10	837226(129920)
		Dogleg-ddp1	10	10	986713(219628)
		Dogleg-ddp2	10	10	982910(219628)
1/200	10	TR-NY	10	10	226288(38587)
		Dogleg-NT	10	10	199777(38587)
		Dogleg-ddp1	10	10	231641(58809)
		Dogleg-ddp2	10	10	230710(58809)
	50	TR-NY	9	9	1240196(184228)
		Dogleg-NT	10	10	1081053(206547)
		Dogleg-ddp1	10	10	1252698(316289)
		Dogleg-ddp2	10	10	1248159(316289)
1/20	10	TR-NY	8	9	151196(31094)
		Dogleg-NT	8	9	155820(31094)
		Dogleg-ddp1	8	9	180621(46748)
		Dogleg-ddp2	8	9	179920(46748)
	50	TR-NY	9	9	1240222(184228)
		Dogleg-NT	9	9	959125(184228)
		Dogleg-ddp1	9	9	1112031(281422)
		Dogleg-ddp2	9	9	1108135(281422)
1/2	10	TR-NY	7	7	130536(26110)
		Dogleg-NT	7	7	132389(26110)
		Dogleg-ddp1	7	7	153739(39408)
		Dogleg-ddp2	7	7	153177(39408)
	50	TR-NY	7	7	840459(139590)
		Dogleg-NT	7	7	716421(139590)
		Dogleg-ddp1	7	7	831101(211688)
		Dogleg-ddp2	7	7	827371(211688)
1	10	TR-NY	6	6	108620(21940)
		Dogleg-NT	6	6	110082(21940)
		Dogleg-ddp1	6	6	127713(32880)
		Dogleg-ddp2	6	6	127178(32880)
	50	TR-NY	6	6	587394(117260)
		Dogleg-NT	7	7	715716(139577)
		Dogleg-ddp1	7	7	830342(211375)
		Dogleg-ddp2	7	7	826804(211375)

Table 2: Numerical results for problem 2 ($n_y = 4, n_x = 2$)

N	method	Iter	Feva	Flops(flops on fun. evaluation)
10	TR-NY	13	14	320289(40485)
	Dogleg-NT	17	22	338715(55313)
	Dogleg-ddp1	25	25	528917(105733)
	Dogleg-ddp2	23	24	358442(87974)
20	TR-NY	12	12	534371(76088)
	Dogleg-NT	25	32	1055687(169211)
	Dogleg-ddp1	21	21	922875(182849)
	Dogleg-ddp2	> 100	> 120	> 3756606
30	TR-NY	15	16	977750(147003)
	Dogleg-NT	28	41	1793488(296056)
	Dogleg-ddp1	19	19	1264428(250057)
	Dogleg-ddp2	100	120	5751614(1213987)
40	TR-NY	15	16	1291350(197073)
	Dogleg-NT	51	73	4432699(724425)
	Dogleg-ddp1	18	18	1603746(316961)
	Dogleg-ddp2	10	10	733040(165055)
50	TR-NY	15	16	1682102(247143)
	Dogleg-NT	49	68	5455563(866813)
	Dogleg-ddp1	18	18	2013450(397511)
	Dogleg-ddp2	10	10	990138(210464)

Table 3: Numerical results for problem 3–5

problem	N	method	Iter	Feva	Flops(flops on fun. evaluation)
Prob. 3	10	TR-NY	17	27	177898(17700)
		Dogleg-NT	16	42	110836(21318)
		Dogleg-ddp1	16	42	125220(40784)
		Dogleg-ddp2	16	42	125355(40784)
	100	TR-NY	19	29	2230166(195708)
		Dogleg-NT	19	41	1450080(229332)
		Dogleg-ddp1	19	41	1610737(459091)
		Dogleg-ddp2	19	41	1612519(459091)
Prob. 4	10	TR-NY	14	14	157474(9125)
		Dogleg-NT	14	14	98674(9125)
		Dogleg-ddp1			(overflow)
		Dogleg-ddp2	31	33	190491(33837)
	100	TR-NY	12	12	991505(80697)
		Dogleg-NT	10	10	742637(66949)
		Dogleg-ddp1			(overflow)
		Dogleg-ddp2	11	11	711230(122737)
Prob. 5	10	TR-NY	4	4	13010(974)
		Dogleg-NT	4	4	13945(974)
		Dogleg-ddp1	4	4	14311(1973)
		Dogleg-ddp2	4	4	14338(1973)
	100	TR-NY	9	9	545441(22549)
		Dogleg-NT	9	9	392975(22549)
		Dogleg-ddp1	9	9	402591(51133)
		Dogleg-ddp2	9	9	403383(51133)

Table 4: Numerical results for problem 6

n $(N-1)$	method	Iter (Feva)	Flops(flops on fun. evaluation)
10	TR-NY	9	56293(3373)
	Dogleg-NT	9	42836(3373)
	Dogleg-ddp1	9	44468(7061)
	Dogleg-ddp2	9	44548(7061)
20	TR-NY	13	221226(9621)
	Dogleg-NT	13	126613(9621)
	Dogleg-ddp1	13	131461(20673)
	Dogleg-ddp2	13	131701(20673)
30	TR-NY	17	467989(18829)
	Dogleg-NT	17	251788(18829)
	Dogleg-ddp1	17	261452(40925)
	Dogleg-ddp2	17	261932(40925)
40	TR-NY	21	805632(30997)
	Dogleg-NT	21	418424(30997)
	Dogleg-ddp1	20	412897(64479)
	Dogleg-ddp2	20	413657(64479)
50	TR-NY	24	1171598(44258)
	Dogleg-NT	24	600476(44258)
	Dogleg-ddp1	24	623568(97181)
	Dogleg-ddp2	24	624718(97181)
70	TR-NY	30	2101290(77440)
	Dogleg-NT	30	1057820(77440)
	Dogleg-ddp1	30	1098536(170849)
	Dogleg-ddp2	30	1100566(170849)
90	TR-NY	36	3294662(119502)
	Dogleg-NT	35	1592831(116155)
	Dogleg-ddp1	35	1654167(256949)
	Dogleg-ddp2	35	1657227(256949)
100	TR-NY	39	3945383(143863)
	Dogleg-NT	38	1925012(140146)
	Dogleg-ddp1	38	1999160(310383)
	Dogleg-ddp2	38	2002860(310383)

Table 5: Numerical results for problem 1 ($n_y = 10, n_x = 2$)

μ	N	method	Iter	Feva	Flops	Flops per Iter
1	10	TR-NY	16	19	3392486	212030
		Dogleg-NT	10	11	1180611	118061

and transition function evaluation); while for Dogleg-DDP1, the total number of flops is 830342 of which 211375 flops are spent on function evaluations.

4 Concluding remarks

We have proposed several efficient algorithms for the unconstrained DTOC problem. These algorithms combine the dogleg algorithm with the DDP or Pantoja's Newton procedure, and possess strong global and local convergence properties yet remain economical. To our knowledge, Algorithm Dogleg-DDP is the first globalization of the DDP method maintains its fast local convergence rate. However, as pointed out in Coleman and Liao [3], the convergence property of dogleg algorithm is weaker than the Nocedal-Yuan algorithm which, in turn, is weaker than the Moré-Sorensen and Gay's algorithms. It is thus desired to adapt our approach to these trust region algorithms.

We have also drawn some comparisons between the single step of the DDP algorithm and the single step of Pantoja's stagewise Newton procedure. Both of these algorithms possess a locally quadratic convergence rate. However, their implementations are different. If the storage requirement is an important issue then it is recommended to use Prog. 1 with the DDP Procedure (Procedure 1) since the DDP Procedure needs less storage than the stagewise Newton procedure (Procedure 2); otherwise, Pantoja's procedure should be used since it needs fewer function evaluations than the DDP algorithm. We should also note that in the original DDP algorithm ([10], [19]) no extra transition function evaluations are needed since the transition function evaluations in the forward recursion (Step 3) of the DDP algorithm are directly used in the next iteration which is not the case for our proposed algorithm.

Acknowledgment: I would like to thank Professor Thomas F. Coleman and Dr. Li-zhi Liao for many discussions relating to this work and for their helpful

comments and suggestions on the manuscript.

References

[1] E. F. Beckenbach and R. Bellman. *Inequalities.* Springer-Verlag, 1983.

[2] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[3] T. F. Coleman and A. Liao. An efficient trust region method for unconstrained discrete-time optimal control problems. Technical Report ctc93tr144, Advanced Computing Research Institute, Cornell University, 1993.

[4] B. D. Craven. *Mathematical Programming and Control Theory.* Chapman and Hall, London, 1978.

[5] T. B. Culver and C. A. Shoemaker. Dynamic optimal control for groundwater remediation with flexible management periods. *Water Resources Research*, 28:629–641, 1992.

[6] J. E. Dennis and J. J. Moré. A characterization of superlinear convergence and its application to quasi-Newton methods. *Math. Comp.*, 28:549–560, 1974.

[7] J. E. Dennis and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Rev.*, 19:46–89, 1977.

[8] G. H. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, Maryland, 1989.

[9] D. Jacobson and D. Mayne. *Differential dynamic programming.* Elsevier Sci. Publ., 1970.

[10] L.-Z. Liao and C. A. Shoemaker. Advantages of differential dynamic programming over Newton's method for discrete-time optimal control problems. Technical Report ctc92tr97, Advanced Computing Research Institute, Cornell University, 1992.

[11] D. Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *Intnl. J. Control*, 3:85–95, 1966.

[12] A. Miele and W. Y. Lee. Optimal trajectories for hypervelocity flight. *Proceedings of 1989 American Control Conference, Pittsburgh*, 3:2017–2023, 1989.

[13] J. J. Moré. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming*, pages 258–287, New York, 1983. Springer-Verlag.

[14] D. M. Murray and S. J. Yakowitz. Differential dynamic programming and Newton's method for discrete optimal control problems. *J. of Optimization Theory and Applications*, 43:395–414, 1984.

[15] J. Nocedal and Y. Yuan. Combining trust region and line search techniques. Technical Report, Dept of EE and Computer Science, Northwestern University, 1992.

[16] J. F. A. De O. Pantoja. Differential dynamic programming and Newton's method. *Intnl. J. Control*, 47:1539–1553, 1988.

[17] M. J. D. Powell. A new algorithm for unconstrained optimization. In J. B. Rosen, O. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65, New York, 1970. Academic Press.

[18] S. J. Wright. Solution of discrete-time optimal control problems on parallel computers. *Parallel Computing*, 16:221–237, 1990.

[19] S. Yakowitz and B. Rutherford. Computational aspects of discrete-time optimal control. *Applied Mathematics and Computation*, 15:29–45, 1984.