Solving unconstrained discrete-time optimal control problems using trust region method

Aiping Liao[1]

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

Version of November 29, 1995

Abstract

Trust region method for a class of large-scale minimization problems, the unconstrained discrete-time optimal control (DTOC) problems, is considered. Although the trust region algorithms developed in [4] and [13] are very economical they lack the ability to handle the so-called hard case. In this paper, We show that the trust region subproblem can be solved within an acceptable accuracy without forming the Hessian explicitly. The new approach is based on the inverse power method for eigenvalue problem and possesses the ability to handle the hard case. Our proposed approach leads to more efficient algorithms for DTOC problems.

Key words. discrete-time optimal control, stagewise Newton's method, trust region method, inverse power method.

[1]This research was conducted using the resources of the Cornell Theory Center, which receives major funding from the National Science Foundation, and New York State. Additional funding comes from the Advanced Research Projects Agency, the National Institutes of Health, IBM Corporation and other members of the center's Corporate Research Institute.

1 Introduction

Discrete-time optimal control problems arise in many practical applications including multi-reservoir control problems[18], the treatment of polluted groundwater [7], supervised learning of feed-forward neural network [15], and inventory control [6]. Moreover, Liao [14] shows that any function given by some computer program (in Griewank-like format [11]) can be transformed into DTOC format. Therefore, to develop efficient method for solving DTOC problems is of great practical and theoretical importance.

In this paper we are concerned with the unconstrained discrete-time optimal control problem,

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

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;

$$F = F(y, x) : R^{n_y N \times n_x (N-1)} \to R$$

where y is a vector in $R^{n_y N}$ and x is a vector in $R^{n_x (N-1)}$ (we put them in the matrix form, i.e., $y = [y_1, \ldots, y_N]$ and $x = [x_1, \ldots, x_{N-1}]$). We assume that

$$T_i(y_i, x_i) : R^{n_y \times n_x} \to R, i = 1, \ldots, N-1$$

and $L_i, \ i = 1, \ldots, N$ are all twice continuously differentiable functions. Throughout this paper we 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.

Problem (P) is called the unconstrained DTOC problem since it can be reduced to a standard unconstrained optimization problem by eliminating the state variables:

$$\text{(Pu)} \qquad \begin{array}{l} \min f(x) \\ \text{s.t. } x \in R^n \end{array}$$

with $n = n_x(N-1)$. The objective function f will be referred to as the reduced objective function of (P).

Obviously, DTOC problem is a minimization problem with a dynamic structure. Taking advantage of this structure, several very fast and efficient local procedures have been developed. Such as the differential dynamic programming (DDP) algorithm [12], the stagewise Newton procedure [20], the Dunn-Bertsekas method [8], and the Pantoja-Mayne method [21]. All 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.

Some successful globalization procedures were recently proposed by Coleman and Liao [4] and Liao [13] which combine the trust region idea with some of the forementioned local procedures. However these algorithms lack the ability to handle the so-called hard case. Their convergence property is not as strong as that of Gay's [10] and the Moré and Sorensen [17] algorithms.

The main task of this paper is to show that, for the DTOC setting, the trust region subproblem can be solved within an acceptable accuracy without the need of the explicit form of the Hessian. More efficient algorithms can thus be developed. In section 2 we propose our method for solving the trust region subproblem. This method is based on the inverse power method for calculating the largest eigenvalue and its associated eigenvector. In section 3 we stated our trust region method for (P) and present some preliminary numerical results. Section 4 contains some concluding remarks.

The remainder of this section consists of some background material of the trust region method–one of the most popular methods for the unconstrained minimization problem (Pu).

The trust region method is an iterative method. During each iteration a trial step is calculated by solving a quadratic model, the trust region subproblem:

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

where $g^k = \nabla f(x^k)$, H_k is the Hessian of f at x^k and $\Delta_k > 0$ is a trust region radius. (Throughout this paper we assume that $\|\cdot\|$ is the l_2 norm.) The next iterate is determined by the solution of this model. The size of the trust region is updated according to the value of the ratio of the real reduction of the objective function over the predicted reduction. This ratio can be regarded as an index of the goodness of the approximation of the quadratic model: the larger, the better.

The following is a genetic algorithm of the trust region method.

Algorithm 1

Initialization. Given x^1 and $\Delta_1 > 0$, set c_1, c_2 such that $0 < c_1 < c_2 < 1$ and $0 < c_3 < 1 < c_4$. Set $k = 1$.

Until convergence do

(i). Solve (subP) approximately for d^k.

(ii). Compute $\rho_k = (f(x^k) - f(x^k + d^k))/(\phi_k(0) - \phi_k(d^k))$.

(iii). If $\rho_k \leq c_1$ then set $\Delta_{k+1} = c_3\Delta_k$,
if $\rho_k \geq c_2$ and $\|d^k\| = \Delta_k$ then set $\Delta_{k+1} = c_4\Delta_k$,
otherwise set $\Delta_{k+1} = \Delta_k$.

(iv). If $\rho_k \leq 0$ then set $x^{k+1} = x^k$ else $x^{k+1} = x^k + d^k$. Set $k = k + 1$.

End

For a more detailed description of the trust region method one is referred to Fletcher [9] and Moré [16].

Basically speaking, different ways of solving (subP) form different trust region algorithms. We thus have Gay's algorithm [10], the Moré and Sorensen algorithm [17], and the dogleg algorithm [24].

If (subP) is solved for d such that

$$\phi(d) \leq \beta_1 \min\{\phi(w) : \|w\| \leq \Delta\|\}, \quad \|d\| \leq \beta_2\Delta, \tag{1}$$

for some positive constraints β_1 and β_2, then the corresponding trust region method has a strong convergence result.

Theorem 1.1 *Let $f : R^n \to R$ be twice continuously differentiable and bounded below on R^n, and assume that H is bounded on the level set $\{x \in R^n : f(x) \leq f(x^0)\}$. Let $\{x^k\}$ be the sequence generated by Algorithm 1 under assumption (1) on the step d^k. Then*

a) *The sequence $\{\nabla f(x^k)\}$ converges to zero.*

b) *If $\{x^k\}$ is bounded then there is a limit point x^* with $H(x^*)$ positive semidefinite.*

c) *If x^* is an isolated limit point of $\{x^k\}$ then $H(x^*)$ is positive semidefinite.*

d) *If $H(x^*)$is nonsingular for some limit point x^* of $\{x^k\}$ then $H(x^*)$ is positive definite, $\{x^k\}$ converges to x^* quadratically.*

Proof. See Moré [16] and Fletcher [9]. □

A very nice property of (subP), as observed by Gay [10] and Sorenson [26], is that the first and second order conditions are sufficient for global minimality: a vector w is a global solution to (subP) if and only if $\|w\| \leq \Delta$ and for some $\lambda^* \geq 0$,

$$(H + \lambda^* I)w = -g, \quad \lambda^*(\Delta - \|w\|) = 0,$$

with $H + \lambda^* I$ positive semidefinite, where I is the identity matrix. If (subP) has no solution on the boundary then the solution of (subP) is given by $d = -H^{-1}g$ with H positive definite. Now we assume that (subP) has some solution on the boundary. We denote by λ_1 the smallest eigenvalue of the matrix H and $\mathcal{S}_1 := \{z : Hz = \lambda_1 z, z \neq 0\}$. If g is not orthogonal to $\mathcal{S}_1$ then the solution to (subP) can be easily approximated by $d_\alpha = -(H+\alpha I)^{-1}g$ with α chosen appropriately. However, if g is orthogonal to $\mathcal{S}_1$ then the solution of (subP) may not be approximated by d_α with any α such that $H + \alpha I$ is positive definite. This usually leads to numerical difficulties and is called the "hard case."

2 Solving the trust region subproblem

In this section we show that an approximate solution d of the trust region subproblem (subP) can be computed without the need of the explicit form of the Hessian and this d satisfies (1) with some $\beta_1 > 0$ and $\beta_2 > 0$. Our analysis is based on Pantoja's procedure of calculating the Newton step of DTOC problem and the inverse power method for eigenvalue problem.

For the non-hard case the solution to (subP) can be exploited by applying Newton's method to the zero finding problem (see Moré and Sorensen [17], for example.)

$$\omega(\lambda) := \frac{1}{\Delta} - \frac{1}{\|d_\lambda\|} = 0. \tag{2}$$

Reinsch [25] proves that if $g \neq 0$ then ω is convex and strictly decreasing on $(-\lambda_1, \infty)$. Moreover, ω is almost linear on $(-\lambda_1, \infty)$. Therefore Newton's method is very efficient when applied to (2). The following algorithm updates λ by Newton's method applied to (2).

Algorithm 2 [Algorithm(3.2) of [17]]

0. Let $\lambda > 0$ with $H + \lambda I$ positive definite and $\Delta > 0$ be given.

1. Factor $H + \lambda I = R^T R$, where R is upper triangular.

2. Solve $R^T R d = -g$ for d.

3. Solve $R^T q = d$ for q.

4. Let

$$\lambda := \lambda + \frac{\|d\|^2}{\|q\|^2} \frac{\|d\| - \Delta}{\Delta}. \tag{3}$$

With proper safeguard the iterations generated by Algorithm 2 converge to the solution of (subP) very fast. However, in the hard case this algorithm may not work. One way to calculate an approximation to the solution to (subP) in the hard case (hold also for non-hard case) is provided in the following result which can be found in [16].

Theorem 2.1 *Let $\sigma \in (0,1)$ be given and suppose that $w \in R^n$ satisfies*

$$(H + \lambda I)w = -g, \quad \lambda \geq 0.$$

Define $H_\lambda = H + \lambda I$. If $z \in R^n$ satisfies

$$\|w + z\| = \Delta, \quad z^T H_\lambda z \leq \sigma(w^T H_\lambda w + \lambda \Delta^2),$$

then $d = w + z$ satisfies (1) with $\beta_1 = 1 - \sigma$ and $\beta_2 = 1$.

In general, an algorithm for finding an approximate solution of (subP) such that (1) holds combines the above two techniques. The corresponding convergence criteria for such an algorithm are as follows ([17]).

Given $\sigma_1, \sigma_2 \in (0,1)$, and $\lambda \geq 0$ such that $H + \lambda I$ is positive definite. If

$$|\Delta - \|d_\lambda\|| \leq \sigma_1 \Delta, \quad \text{or } \|d_\lambda\| \leq \Delta, \ \lambda = 0, \tag{4}$$

then $d = d_\lambda$ is an approximate solution. The hard case is taken into account by calculating z whenever $\|d_\lambda\| < \Delta$, and if

$$z^T H_\lambda z \leq \sigma_1(2 - \sigma_2) \max\{\sigma_2, d_\lambda^T H_\lambda d_\lambda + \lambda \Delta^2\} \tag{5}$$

we take $d = d_\lambda + z$ as the approximate solution.

It is shown in [17] that d so obtained satisfies

$$\phi(d) - \phi^* \leq \sigma_1(2 - \sigma_2)\max\{\sigma_2, |\phi^*|\}, \quad \|d\| \leq (1 + \sigma_1)\Delta \tag{6}$$

where ϕ^* is the optimal value of (subP). As pointed out in [17], we can set $\sigma_2 = 0$ since σ_2 is only required to deal with the case when $g = 0$ and H is positive semidefinite and singular (in which a trust region method terminates).

So far we have described an outline of a framework of the method for solving (subP). We now turn to the trust region subproblem associated with our target problem (P). In the following we shall provide the building blocks and finally, put them together.

Pantoja [20] shows that the DTOC problem (P) can be transformed into the following equivalent problem:

$$\text{(PP)} \qquad \begin{array}{l} \min F' := y'_{N,n_y+1} + L_N(y_N) =: L_N(y'_N) \\ y'_{i+1} = T'_i(y'_i, x_i), \quad i = 1, \ldots, N-1 \\ y'_1 = \bar{y}'_1 \end{array}$$

where

$$\begin{aligned} y'_i &= [y_i^T, y'_{i,n_y+1}]^T \in R^{n_y+1}, \ i = 1, \ldots, N, \\ T'_i(y'_i, x_i) &= [T_i(y_i, x_i)^T, y'_{i,n_y+1} + L_i(y_i, x_i)]^T, \ i = 1, \ldots, N-1 \end{aligned}$$

and $\bar{y}'_1 = [\bar{y}_1^T, 0]^T$.

In the following we denote $n_{y'} := n_y + 1$, $(f)_x$ the gradient of f with respect to x and $\frac{\partial f}{\partial x}$ the Jacobian of f with respect to x. Algorithm 1 is Pantoja's algorithm for calculating the Newton direction $d = -H^{-1}g$, where H is the Hessian of $f(x)$ (we assume that H is invertible).

Algorithm 3 [Pantoja's stagewise Newton's method [20]]

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

$$
\begin{aligned}
A_i &= (\frac{\partial T_i'}{\partial y_i'})^T P (\frac{\partial T_i'}{\partial y_i'}) + \sum_{j=1}^{n_{y'}} G_j (T_{i,j}')_{y_i' y_i'} \\
B_i &= (\frac{\partial T_i'}{\partial y_i'})^T P (\frac{\partial T_i'}{\partial x_i}) + \sum_{j=1}^{n_{y'}} G_j (T_{i,j}')_{y_i' x_i} \\
C_i &= (\frac{\partial T_i'}{\partial x_i})^T P (\frac{\partial T_i'}{\partial x_i}) + \sum_{j=1}^{n_{y'}} G_j (T_{i,j}')_{x_i x_i} \\
D_i &= (\frac{\partial T_i'}{\partial x_i})^T Q \\
E_i &= (\frac{\partial T_i'}{\partial y_i'})^T Q.
\end{aligned}
$$

(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

Same as in Coleman and Liao [4], we define the procedure for computing Newton's step $d = -H^{-1}g$ as follows: we first transform (P) into (PP) and then use Algorithm 3 to obtain d. We denote this procedure by

$$\begin{aligned} d &= newton(\{x_i\}, \{(L_i)_{x_i}, (L_i)_{y_i}, (L_i)_{x_ix_i}, (L_i)_{x_iy_i}, (L_i)_{y_iy_i}\}, \\ &\quad \{(T_i)_{x_i}, (T_i)_{y_i}, (T_i)_{x_ix_i}, (T_i)_{x_iy_i}, (T_i)_{y_iy_i}\}) \end{aligned}$$

or simply $d = newton(x, L, T)$.

Some properties of Pantoja's Procedure are summarized in the following proposition.

Proposition 2.2 (i). *Algorithm 3 is equivalent to the backward Gaussian elimination and it can be carried through if and only if C_i is nonsingular for all $i = 1, \ldots, N-1$.*

(ii). *H is positive definite if and only if C_i is positive definite for all $i = 1, \ldots, N$;*

(iii). *If we ignore the lower order terms the total number of operations of Pantoja's Procedure is*

$$\sum_{i=1}^{N}(2n_i^3 + \frac{7}{2}n_i^2\bar{n}_i + 2n_i\bar{n}_i^2 + \frac{1}{3}\bar{n}_i^3) \sim O(N).$$

(iv). *The i-th block component of g, the gradient of f, is given by $g_i = D_i^1$ where*

$$D_i = D_i^1 + D_i^2 := (\frac{\partial T_i}{\partial x_i})^T E_{i-1} + (\frac{\partial T_i}{\partial x_i})^T B_{i-1}\alpha_{i-1}.$$

Proof. The proofs of (ii) and (iii) can be found in [4]. (iv) is quite obvious (see also [14]). Finally, (i) can be shown using direct yet tedious comparison of Algorithm 3 and the backward Gaussian elimination to the system that defines the Newton direction, i.e., $Hd = -g$. □

Lemma 2.3 *Assume that H and g are evaluated at $\bar{x}$. For any given block vector $z = (z_1^T, \ldots, z_{N-1}^T)^T$ and $h = (z_1^T, \ldots, z_{N-1}^T)^T$ with $z_i, h_i \in R^{n_x}$, $(H + \mathrm{diag}(z))^{-1}h$, if it exists, can be calculated by applying Pantoja's procedure to the following augmented DTOC with F being replaced by F^a at $\bar{x}$:*

$$F^a = F + \sum_{i=1}^{N-1}[\frac{\lambda}{2}\|x_i - \bar{x}_i\|^2 - (g_i + h_i)^T(x_i - \bar{x}_i)]$$

$$
\begin{aligned}
&= \sum_{i=1}^{N-1} [L_i(y_i, x_i) + \frac{\lambda}{2}\|x_i - \bar{x}_i\|^2 - (g_i + h_i)^T (x_i - \bar{x}_i)] + L_N(y_N) \\
&=: \sum_{i=1}^{N-1} L_i^a(y_i, x_i) + L_N^a(y_N).
\end{aligned}
$$

Proof. The proof is easy using Lemma 2.2 of Coleman and Liao [4]. □

We note that the update (3), which is based on Newton's method for equation $\omega(\lambda) = 0$, is equivalent to

$$
\lambda := \lambda + \frac{\|d\|^2}{d^T d'} \frac{\|d\| - \Delta}{\Delta}
$$

where $d = -(H + \lambda I)^{-1} g$ and $d' = (H + \lambda I)^{-1} d$.

We thus propose a modified version of Algorithm 2 which does not need the explicit form of the Hessian.

Algorithm 2'

0. Let $\lambda > 0$ with $H + \lambda I$ positive definite and $\Delta > 0$ be given.

1. Calculate $d = -(H + \lambda I)^{-1} g$.

2. Calculate $d' = (H + \lambda I)^{-1} d$.

3. Let

$$
\lambda := \lambda + \frac{\|d\|^2}{d^T d'} \frac{\|d\| - \Delta}{\Delta}.
$$

According to Lemma 2.3, $d = -(H + \lambda I)^{-1} g$ can be calculated by setting

$$
\begin{aligned}
d \;=\; & newton(\{x_i\}, \{(L_i)_{x_i}, (L_i)_{y_i}, (L_i)_{x_i x_i} + \lambda I_s, (L_i)_{x_i y_i}, (L_i)_{y_i y_i}\}, \\
& \{(T_i)_{x_i}, (T_i)_{y_i}, (T_i)_{x_i x_i}, (T_i)_{x_i y_i}, (T_i)_{y_i y_i}\})
\end{aligned}
$$

where $I_s \in R^{n_x \times n_x}$ is the identity matrix; and $d' = (H + \lambda I)^{-1} d$ can be calculated by setting

$$
\begin{aligned}
d \;=\; & newton(\{x_i\}, \{(L_i)_{x_i} - (g_i + d_i), (L_i)_{y_i}, (L_i)_{x_i x_i} + \lambda I_s, (L_i)_{x_i y_i}, (L_i)_{y_i y_i}\}, \\
& \{(T_i)_{x_i}, (T_i)_{y_i}, (T_i)_{x_i x_i}, (T_i)_{x_i y_i}, (T_i)_{y_i y_i}\}).
\end{aligned}
$$

Therefore the total number of operations needed for Algorithm 2' is $O(N)$.

To realize Theorem 2.1, we try to locate the desired λ using the bisection method and then perform one step of the inverse power method to calculate z. The following is the bisection algorithm to find an approximation of $-\lambda_1$.

Algorithm 4

Initialization. Given $\varepsilon > 0$. Let $l \geq 0$ be the smallest integer such that $H + 2^l I$ is positive definite. Set $\lambda = 2^l$ and $\lambda_L = 2^{l-1}$ if $l \geq 1$, $\lambda_L = 0$ otherwise.

Until $\lambda - \lambda_L \geq \varepsilon$ **do**

(i). Let $\lambda_M = (\lambda + \lambda_L)/2$.

(ii). If $H + \lambda_M I$ is positive definite, set $\lambda = \lambda_M$; otherwise, set $\lambda_L = \lambda_M$.

End

We note that if λ is the output of Algorithm 4, then $0 < \lambda + \lambda_1 < \varepsilon$ and $H + \lambda I$ is positive definite. It follows from Proposition 2.2 that to check if $H + \lambda I$ is positive definite we need only to check if each C_i, $i = 1, \ldots, N-1$ is positive definite during the calculation of

$$\begin{aligned} d &= newton(\{x_i\}, \{(L_i)_{x_i}, (L_i)_{y_i}, (L_i)_{x_i x_i} + \lambda I_s, (L_i)_{x_i y_i}, (L_i)_{y_i y_i}\}, \\ &\quad \{(T_i)_{x_i}, (T_i)_{y_i}, (T_i)_{x_i x_i}, (T_i)_{x_i y_i}, (T_i)_{y_i y_i}\}). \end{aligned}$$

The number of operations for one iteration of Algorithm 4 is $O(N)$.

We now show that if $\lambda \approx -\lambda_1$ then one iteration of inverse power method (see, for example, Parlett [22]) with (almost) any vector would produce a good approximate eigenvector. Suppose that the eigenvalues of H are labeled in increasing order,

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

Since H is similar to the diagonal matrix $\mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, we have

$$H = \sum_{i=1}^{n} \lambda_i o_i o_i^T$$

where $(o_1, \ldots, o_n)$ is a matrix of orthonormalized eigenvectors of H. Let $index_1 := \{i : \lambda_i = \lambda_1\}$. If $u_0 \in R^n$ with $\|u_0\| = 1$ is chosen such that at least one of $u_0^T o_i, i \in index_1$ is non-zero, then one iteration of the inverse power method gives

$$\begin{aligned} u_1 &:= (H + \lambda I)^{-1} u_0 \\ &= \sum_{i=1}^{n} \frac{o_i^T u_0}{\lambda_i + \lambda} o_i. \end{aligned}$$

Let

$$o' = \frac{\sum_{i \in index_1}(o_i^T u_0)o_i}{\|\sum_{i \in index_1}(o_i^T u_0)o_i\|}.$$

Obviously, if $\lambda + \lambda_1 \geq 0$ is very small, then

$$\hat{z} := u_1/\|u_1\| \approx o'$$

which is an eigenvector associated with λ_1.

We note that, according to Proposition 2.2 and Lemma 2.3, Pantoja's procedure (Algorithm 3) when applied to the augmented F^a is an implicit, backward, block Gauss elimination method for the system

$$(H + \lambda I)d = -g. \tag{7}$$

One may note that the closer λ is to $-\lambda_1$ $H + \lambda I$ is closer to becoming singular. However, according to Lemma 5.4.1 of Chatelin [3], the error made in solving (7) is mainly in the direction o'. Therefore, the singularity will not be a disadvantage for our calculations.

Once we have such $\hat{z}$ we use the following formula to calculate τ so that $\|d_\lambda - \tau\hat{z}\| = \Delta$ ([17]):

$$\tau = \frac{\Delta^2 - \|d_\lambda\|^2}{d_\lambda^T\hat{z} + sgn(d_\lambda^T\hat{z})[(d_\lambda^T\hat{z})^2 + (\Delta^2 - \|d_\lambda\|^2)]^{1/2}} \tag{8}$$

where $sgn(x) = 1$ if $x \geq 0$ and $sgn(x) = -1$ otherwise. To check (5) we need to calculate $z^T H z$ and $d_\lambda^T H d_\lambda$. Since H is not available explicitly, we calculate them using the technique in Coleman and Liao [4]: note that H can be partitioned as $H = H_1 + H_2$ with

$$\begin{aligned} H_1 &= \sum_{i=1}^{N-1}\sum_{j=1}^{n_y}[(\nabla F)_{y_{i,j}} \cdot \nabla_{xx}^2 T_{i,j}] \\ H_2 &= [I, (\frac{\partial y}{\partial x})^T]\nabla^2 F \begin{bmatrix} I \\ \frac{\partial y}{\partial x} \end{bmatrix}. \end{aligned}$$

For any give d consider the following DTOP problem:

$$(\mathrm{P}_1) \qquad \begin{array}{l} \min F_1(w,d) := \sum_{i=1}^{N-1}[(L_i)_{y_i}^T w_i + (L_i)_{x_i}^T d_i] + (L_N)_{y_N}^T w_N \\ w_{i+1} = W_i(w_i, d_i) := \frac{\partial T_i}{\partial y_i} w_i + \frac{\partial T_i}{\partial x_i} d_i + \frac{1}{2} d_i^T (T_i)_{x_i x_i} d_i, \quad i = 1, \ldots, N-1 \\ w_1 = 0, \end{array}$$

where $d_i^T(T_i)_{x_ix_i}d_i$ is an n_y-vector whose j-th component is $d_i^T(T_{i,j})_{x_ix_i}d_i$. Let $f_1(d)$ be the reduced function. Then

$$f_1(d) = g^T d + \frac{1}{2}d^T H_1 d$$

which can be calculated via (P_1) using (ignoring the lower order terms)

$$N \cdot (n_y n_x^2)$$

operations.

We now consider another DTOP problem:

$$(\mathrm{P}_2) \qquad \begin{array}{l} \min F_2(w,d) := \frac{1}{2}(\sum_{i=1}^{N-1}[w_i^T(L_i)_{y_iy_i}w_i + 2w_i^T(L_i)_{y_ix_i}d_i + d_i^T(L_i)_{x_ix_i}d_i]) \\ \quad +\frac{1}{2}w_N^T(L_N)_{y_Ny_N}w_N \\ w_{i+1} = W_i(w_i,d_i) := \frac{\partial T_i}{\partial y_i}w_i + \frac{\partial T_i}{\partial x_i}d_i, \quad i = 1,\ldots,N-1 \\ w_1 = 0. \end{array}$$

Let $f_2(d)$ be the reduced function which is also a quadratic function. It can be shown that

$$f_2(d) = \frac{1}{2}d^T H_2 d$$

$$N \cdot (n_y^2 + n_y n_x + n_x^2)$$

operations. Putting them together, $d^T H d$ can be calculated via $d^T H d = 2(f_1(d) + f_2(d) - g^d)$ using $O(N)$ operations.

We have discussed the ingredients for the two main techniques. Now the task is to twist them together to form an implementable algorithm. The following is our algorithm for solving (subP).

Algorithm (subP)

Initialization. If H is not positive definite we find an interval $[\lambda_L, \lambda)$ such that $\lambda_L \le -\lambda_1 < \lambda$ using the initialization step of Algorithm 3. If H is positive definite then set $[\lambda_L, \lambda) = [-1, 0)$. Set $k = 1$.

Until convergence do

(i). Calculate $d_\lambda = -(H + \lambda I)^{-1}g$.

(ii). If $\|d_\lambda\| < \Delta$, compute $\hat{z}$ using one iteration of the inverse power method with $u_0 = e_{[k|n]}$, where e_i is the ith column of the indentity matrix and $[k|n]$ is the remainder of k divided by n; compute τ using (8). Update λ_L and λ via steps (i) and (ii) of Algorithm 4. Set $d = d_\lambda + \tau\hat{z}$.

(iii). If H_λ is positive definite and $g \neq 0$, then update λ using one iteration of Algorithm 2' to get $\hat{\lambda}$. If $H + \hat{\lambda} I$ is positive definite then set $\lambda = \hat{\lambda}$; otherwise update λ_L and λ via steps (i) and (ii) of Algorithm 4.

(iv). Set $k = k + 1$.

End

The way we choose u_0 for the inverse power method guarantees that at least one of $u_0^T o_i, i \in index_1$ is non-zero for every n iterations. However, since the probability for this assumption to be failed is zero, in practice it is reasonable to choose u_0 randomly.

Proposition 2.4 *Algorithm (subP) terminates in a finite number of iterations.*

Proof. We first note that whenever a λ such that $\omega(\lambda) \geq 0$ is found Algorithm (subP) is reduced to Algorithm 1 and stops in finite many iterations. In the following we will show that after finite many iterations such a λ can be found or the algorithm terminates due to satisfying (5).

If the length of the interval $[\lambda_L, \lambda)$ converges to zero then the outcome of one iteration of the inverse power method converges to the invariant space associated with λ_1. (5) is therefore satisfied in finite many iterations. Now suppose that the length of the interval $[\lambda_L, \lambda)$ remains bounded away from zero. Then step (ii) happens only a finite many times which implies that a λ such that $\omega(\lambda) \geq 0$ can be found after finite many of iterations. This establishes the proposition. $\square$

3 Trust region algorithms for DTOC problems

With the method for solving the trust region subproblem at hand the associated trust region algorithms are easy to establish. We need only to embed Algorithm (subP) in the corresponding trust region methods. For completeness, we state below two such trust region algorithms. One is the pure trust region algorithm and the other is the Nocedal and Yuan algorithm [19]. The Nocedal-Yuan algorithm is actually an algorithm that employs both trust region techniques and line searches. We will take a simple version of it.

Algorithm I (pure trust region method)

Initialization. Given x^1 and $\Delta_1 > 0$, choose c_1 and c_2 such that $0 < c_1 < c_2 < 1$ and $0 < c_3 < 1 < c_4$. Set $k = 1$.

Until convergence do

(i). Solve (subP) approximately for d^k using algorithm (subP).

(ii). Compute $\rho_k = (f(x^k) - f(x^k + d^k))/(\phi_k(0) - \phi_k(d^k))$.

(iii). If $\rho_k \le c_1$ then set $\Delta_{k+1} = c_3\Delta_k$,
if $\rho_k \ge c_2$ and $\|d^k\| = \Delta_k$ then set $\Delta_{k+1} = c_4\Delta_k$,
otherwise set $\Delta_{k+1} = \Delta_k$.

(iv). If $\rho_k \le 0$ then set $x^{k+1} = x^k$ else $x^{k+1} = x^k + d^k$. Set $k = k + 1$.

End

Since the approximate solution d^k of (subP) satisfies (1), Theorem 1.1 holds for the above algorithm.

To incorporate Algorithm (subP) into the Nocedal-Yuan algorithm we require that the following criteria also hold in addition to (5) for the case when $g \neq 0$.

$$\lambda - \lambda_L \le \delta_1\|g^k\|/\Delta_k \tag{9}$$

$$\|H_\lambda z\| \le (\delta_2/\Delta_k)|d_\lambda^T g^k| \tag{10}$$

where δ_1 and δ_2 are some positive constants and $\delta_2 \in (0,1)$. We note that for any vector z Hz can be calculated as follows. Applying Algorithm g of [4] to problem (P_1) with d being replaced by z we thus obtain $g_1 := g + H_1 z$ where $H = H_1 + H_2$. Applying Algorithm g of [4] to problem (P_2) with d being replaced by z we thus obtain $g_2 := H_2 z$. Therefore Hz can be calculated via $Hz = g_1 + g_2 - g$ and the total number of operations is $O(N)$.

Algorithm II (Nocedal-Yuan-like method)

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

Until convergence do

(i). Solve (subP) using Algorithm (subP) so that (4) or (5) and (9)–(10) hold.

(ii). Calculate $f(x^k + d^k)$.
If $f(x^k + d^k) \ge f(x^k)$
use a simple binary search to find $0 < s_k < 1$ such that $f(x^k + s_k d^k) < f(x^k)$

and put $x^{k+1} = x^k + s_k d^k$, $\Delta_{k+1} = \|x^{k+1} - x^k\|$;
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

For this algorithm we have the following convergence results.

Theorem 3.1 *Assume that $f : R^n \to R$ is twice continuously differentiable on the level set $\Omega = \{x : f(x) \leq f(x^0)\}$ and the sequence $\{x^k\}$ be generated by our algorithm. If Ω is a compact set and $\nabla^2 f$ is bounded on Ω, then:*

1. *The sequence x^k satisfies*

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

2. *If f is convex it follows that*

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

3. *If x^k converges to a point x^* then $\nabla^2 f(x^*)$ is positive semi-definite.*

4. *If x^k converges to a point x^* such that $\nabla^2 f(x^*)$ is positive definite, the rate of convergence is quadratic.*

Proof. What we need to show is that the approximate solution d^k of (subP) satisfies

$$-\phi_k(d^k) \geq \tau\|g^k\| \min\{\Delta_k, \frac{\|g^k\|}{\|H_k\|}\} \tag{11}$$

and

$$(d^k)^T g^k \leq -\tau\|g^k\| \min\{\Delta_k, \frac{\|g^k\|}{\|H_k\|}\} \tag{12}$$

where τ is some positive constant, and the rest of the proof is the same as that of Nocedal and Yuan [19].

Obviously d^k satisfies (1), it thus follows from Lemma (4.8) of Moré [16] that (11) holds. On the other hand, we assume that $g^k \neq 0$ (otherwise (12) already holds). If $d^k = d_\lambda$ and d_λ satisfies (4), then either H is positive definite and $\|d^k\| \leq \Delta_k$ in which case d^k is an optimal solution and (12) holds followed by Lemma 2.4 of [19], or, $d^k = d_\lambda$ such that

$$(1-\sigma_1)\Delta_k \leq \|d_\lambda\| \leq (1+\sigma_1)\Delta_k.$$

Since $(H_k + \lambda I)d_\lambda = -g^k$, we have

$$\lambda_1 + \lambda \leq \frac{\|g^k\|}{\|d_\lambda\|} \leq \frac{\|g^k\|}{(1-\sigma_1)\Delta_k}.$$

Thus

$$\begin{aligned}
(d^k)^T g^k &= -(g^k)^T(H_k+\lambda I)^{-1}g^k \\
&\leq -\|g^k\|^2/(\lambda_n+\lambda) \\
&\leq -\|g^k\|^2/(\lambda_n - \lambda_1 + \frac{\|g^k\|}{(1-\sigma_1)\Delta_k}) \\
&\leq -\|g^k\|^2/(2\|H_k\| + \frac{\|g^k\|}{(1-\sigma_1)\Delta_k}) \\
&\leq -\|g^k\|^2/(2\max\{2\|H_k\|, \frac{\|g^k\|}{(1-\sigma_1)\Delta_k}\}) \\
&= -\frac{1}{2}\|g^k\|\min\{(1-\sigma_1)\Delta_k, \|g^k\|/(2\|H_k\|)\} \\
&\leq -\tau\|g^k\|\min\{\Delta_k, \frac{\|g^k\|}{\|H_k\|}\}
\end{aligned}$$

where $\tau = \min\{1/4, (1-\sigma_1)/2\}$. Thus (12) holds. Now we assume that $d^k = d_\lambda + z$ and (5), (9) and (10) hold. It follows from (9) that $\lambda \leq \|H_k\| + \delta_1\|g^k\|/\Delta_k$. Therefore, same as in [19],

$$\begin{aligned}
(d_\lambda)^T g^k &= -(g^k)^T(H_k+\lambda I)^{-1}g^k \\
&\leq -\|g^k\|^2/(\|H_k\|+\lambda) \\
&\leq -\|g^k\|^2/(2\|H_k\| + \delta_1\|g\|/\Delta_k)
\end{aligned}$$

$$
\begin{aligned}
&\le -\frac{1}{2}\|g^k\| \min\{\Delta_k/\delta_1, \|g^k\|/(2\|H_k\|)\} \\
&= -\tau'\|g^k\| \min\{\Delta_k, \frac{\|g^k\|}{\|H_k\|}\}
\end{aligned}
$$

where $\tau' = \min\{1/4, 1/(2\delta_1)\}$. On the other hand, it follows from (10) that

$$
\begin{aligned}
|z^T g^k| &= |z^T H_\lambda d_\lambda| \\
&\le \Delta_k \|H_\lambda z\| \\
&\le \delta_2 |d_\lambda^T g^k|.
\end{aligned}
$$

We thus have

$$
\begin{aligned}
(d^k)^T g^k &= d_\lambda^T g^k + z^T g^k \\
&\le d_\lambda^T g^k + |z^T g^k| \\
&\le (1-\delta_2) d_\lambda^T g^k \\
&\le -\tau'(1-\delta_1) \min\{\Delta_k, \frac{\|g^k\|}{\|H_k\|}\}.
\end{aligned}
$$

(12) thus holds for $\tau := \tau'(1-\delta_1)$. □

Coleman and Liao [4] show that the quantities such as g, $\phi(d)$ and ρ can be calculated very efficiently for the DTOC setting. Therefore our new approaches remain as economical as those methods of Coleman and Liao [4] and Liao [13] yet have a convergence property as strong as that of the Moré-Sorensen [17] and Gay's [10] algorithms.

We tested our approaches with the problems collected in the Appendix of Coleman and Liao [4]. Our algorithm was written in MATLAB and all runs were performed on a SUN Sparcstation. We take $\sigma_1 = 0.1$ and $\sigma_2 = 0$ for Algorithm (subP), and $c_1 = 0.1$, $c_2 = 0.5$ in Algorithm I and Algorithm II. We also take $\delta_1 = 10.0$ and $\delta_1 = 0.9$ in Algorithm II. The line search algorithm in Algorithm II is carried out so that

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

We choose the initial trust region radius as $\Delta_1 = 1$ for all trust region algorithms, and terminate the iterations when $\|g\| < 10^{-6}$.

The numerical results are presented in Tables 1–4. C-L stands for the algorithm developed by Coleman and Liao [4] and the corresponding data are taken from

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

		C-L	Algorithm I	Algorithm II
μ	N	Iter(Feva)	Iter	Iter(Feva)
0	10	10(10)	10	10(10)
	50	9(9)	10	10(10)
1/200	10	10(10)	10	10(10)
	50	9(9)	10	10(10)
1/20	10	8(9)	10	8(9)
	50	9(9)	10	10(10)
1/2	10	7(7)	7	7(7)
	50	7(7)	7	7(7)
1	10	6(6)	6	6(6)
	50	6(6)	6	6(6)

[4]. For the trust region algorithms with enhancement of line searches, we list the number of iterations and the number of function evaluations. For the pure trust region algorithm we only give the number of iterations since this is the same as the number of function evaluation. We also note that the average number of iterations of Algorithm (subP) is quite small for all these test problems (for most cases only 1 or 2 iterations are needed).

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

	C-L	Algorithm I	Algorithm II
N	Iter(Feva)	Iter	Iter(Feva)
10	13(13)	13	10(10)
20	12(12)	15	12(14)
30	15(16)	25	12(17)
40	15(16)	27	15(18)
50	15(16)	37	15(18)

Table 3: Numerical results for problem 3–5

		C-L	Algorithm I	Algorithm II
problem	N	Iter(Feva)	Iter	Iter(Feva)
Prob. 3	10	17(27)	13	12(24)
	100	19(29)	25	20(29)
	500	25(36)	24	20(28)
	1000	27(33)	30	26(34)
Prob. 4	10	14(14)	26	19(24)
	100	12(12)	11	11(11)
	500	25(25)	17	17(17)
	1000	35(35)	16	16(16)
Prob. 5	10	4(4)	4	4(4)
	100	9(9)	8	8(8)
	500	17(17)	10	10(10)
	1000	23(23)	9	9(9)

Table 4: Numerical results for problem 6

	C-L	Algorithm I	Algorithm II
n $(N-1)$	Iter	Iter	Iter
10	9	9	9
20	13	13	13
30	17	17	17
40	21	20	20
50	24	23	23
70	30	29	29
90	36	34	34
100	39	37	37

4 Concluding remarks

We have proposed a method for solving the trust region subproblem based on the fact that Pantoja's procedure can be employed to detect the positive definiteness of the matrix $H + \lambda I$ for any scale λ. Our method differs from those of Moré and Sorensen [17] and Gay [10] in the way of handling the hard case. For the hard case we use the inverse power method to find an approximate of the desired eigenvector. Moreover, our approach is still very economical. All the quantities are calculated without the need of the explicit Hessian. The convergence property of our new approach is stronger than those of Coleman and Liao [4] and Liao [13].

Finally, let's consider how to solve constrained DTOC problems. The general constrained DTOC problem has the following format

$$
\text{(CP)} \qquad \begin{array}{l} \min F := \sum_{i=1}^{N-1} L_i(y_i, x_i) + L_N(y_N) \\ y_{i+1} = T_i(y_i, x_i), \quad i = 1, \ldots, N-1 \\ G_i(y_i, x_i) \leq 0, \quad i = 1, \ldots, N-1 \\ y_1 = \bar{y}_1. \end{array}
$$

A natural approach is to use the modified barrier function method of Polyak [23]. During each iteration Polyak's method solves an unconstrained problem with two parameters. This method is shown to be superior to the classic barrier methods. Some comments and modifications of Polyak's method can be found in Conn et al. [5], Breitfeld and Shanno [1], and [2]. The advantage of using the modified barrier function method for (CP) is the following: the corresponding logarithmic barrier function for the kth iteration is just an unconstrained DTOC problem:

$$
\text{(CPB)} \qquad \begin{array}{l} \min \mathcal{M}^{(k)} := \sum_{i=1}^{N-1} (L_i(y_i, x_i) - \mu^{(k)} \lambda_i^{(k)} \log(1 + \frac{G_i}{\mu^{(k)}})) + L_N(y_N) \\ y_{i+1} = T_i(y_i, x_i), \quad i = 1, \ldots, N-1 \\ y_1 = \bar{y}_1. \end{array}
$$

Our approach is thus a suitable choice for problem (CPB).

Acknowledgment: I would like to thank Tom Coleman for many helpful discussions.

References

[1] M. G. Breitfeld and D. F. Shanno. Computational experience with penalty-barrier methods for nonlinear programming. RRR 17–93, RUTCOR, 1993.

[2] M. G. Breitfeld and D. F. Shanno. A globally convergent penalty-barrier algorithm for nonlinear programming and its computational performance. RRR 12–94, RUTCOR, 1994.

[3] F. Chatelin. *Eigenvalues of Matrix.* John Wiley & Sons, 1993.

[4] T. F. Coleman and A. Liao. An efficient trust region method for unconstrained discrete-time optimal control problems. *Computational Optimization and Applications*, 4:47–66, 1995.

[5] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Large-scale nonlinear constrained optimization: a current survey. Technical Report TR/PA/94/03, CERFACS, France, 1994.

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

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

[8] J. C. Dunn and D. P. Bertsekas. Efficient Dynamic Programming Implementations of Newton's method for Unconstrained Pptimal Control Problems. *J. of Optimization Theory and Applications*, 63:23–38, 1989.

[9] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, Ltd, 1987.

[10] D. M. Gay. Computing optimal local constrained steps. *SIAM J. Sci. Stat. Comput.*, 2:186–197, 1981.

[11] A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108, Boston, 1989. Kluwer Academic Publishers.

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

[13] A. Liao. Some efficient algorithms for unconstrained discrete-time optimal control problems. Technical Report ctc93tr159, Advanced Computing Research Institute, Cornell University, 1993.

[14] A. Liao. Automatic optimization. Technical Report 94–4, Cornell Computational Optimization Project, Cornell University, 1994.

[15] A. Liao. Second order methods for supervised learning of a multilayer feedforward neural network. Technical Report 95-6, Cornell Computational Optimization Project, Cornell University, 1995.

[16] 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.

[17] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM J. Sci. Stat. Comput.*, 4:553–572, 1983.

[18] 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.

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

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

[21] J. F. A. De O. Pantoja and D. Q. Mayne. Sequential quadratic programming algorithm for discrete optimal control problems with control inequality constraints. *Intnl. J. Control*, 53:823–836, 1991.

[22] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentics–Hall, Inc., Englewood Cliffs, 1980.

[23] R. Polyak. Modified barrier functions (theory and methods). *Mathematical Programming*, 54:177–222, 1992.

[24] 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.

[25] C. H. Reinsch. Smoothing by spline functions II. *Numer. Math.*, 16:451–454–572, 1971.

[26] D. C. Sorensen. Collinear scaling and sequential estimation in sparse optimization algorithms. *Mathematical Programming Study*, 18:135–159, 1982.