Journal of Global Optimization **23:** 401–421, 2002.

A Globally Convergent and Efficient Method for Unconstrained Discrete-Time Optimal Control

CHI KONG NG[1], LI-ZHI LIAO[2,*] and DUAN LI[3]

[1]*Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong SAR, China. (E-mail: ckng@se.cuhk.edu.hk)* [2]*Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong SAR, China.* [*]*Corresponding author. (e-mail: liliao@hkbu.edu.hk)* [3]*Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong SAR, China. (E-mail: dli@se.cuhk.edu.hk)*

Abstract. Shift schemes are commonly used in non-convex situations when solving unconstrained discrete-time optimal control problems by the differential dynamic programming (DDP) method. However, the existing shift schemes are inefficient when the shift becomes too large. In this paper, a new method of combining the DDP method with a shift scheme and the steepest descent method is proposed to cope with non-convex situations. Under certain assumptions, the proposed method is globally convergent and has q-quadratic local conve rgence. Extensive numerical experiments on many test problems in the literature are reported. These numerical results illustrate the robustness and efficiency of the proposed method.

Key words: Unconstrained discrete-time optimal control, Non-convex optimization, Differential dynamic programming, Steepest descent, Global convergence, Quadratic convergence

1. Introduction

Optimal control has broad applications in diverse areas, such as engineering, economics, ecology, etc. This paper considers unconstrained discrete-time optimal control problems (UDOCP) in the following format:

$$\min_{\mathbf{U}} \; L(\mathbf{U}) = \sum_{t=1}^{N} g_t(\mathbf{x}_t, \mathbf{u}_t) + g_{N+1}(\mathbf{x}_{N+1}) \tag{1}$$

$$\text{where } \mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t), \quad t = 1, 2, \ldots, N, \tag{2}$$

$$\mathbf{x}_1 \equiv \bar{\mathbf{x}}_1 \quad \text{(given and fixed)}, \tag{3}$$

$$\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_{N+1}^T]^T, \; \mathbf{x}_t \in I\!R^n, \; t = 1, \ldots, N+1, \tag{4}$$

$$\mathbf{U} = [\mathbf{u}_1^T, \mathbf{u}_2^T, \ldots, \mathbf{u}_N^T]^T, \; \mathbf{u}_t \in I\!R^m, \; t = 1, 2, \ldots, N. \tag{5}$$

Here, $\mathbf{U}$ is the control policy with controls $\mathbf{u}_t$, $t = 1, 2, \ldots, N$; $\mathbf{X}$ is the trajectory with states $\mathbf{x}_t$, $t = 1, 2, \ldots, N+1$, associated with control policy $\mathbf{U}$; $L : I\!R^{Nm} \mapsto I\!R$ is the total loss function with stagewise loss functions $g_t : I\!R^n \times I\!R^m \mapsto I\!R$,

$t = 1, 2, \ldots, N$, and $g_{N+1} : \mathbb{R}^n \mapsto \mathbb{R}$; $f_t : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$, $t = 1, 2, \ldots, N$, are transition functions. In general, the following standard assumptions are imposed:

STANDARD ASSUMPTIONS

1. g_t and f_t are twice continuously differentiable functions for all t.
2. The level set $\Omega(\mathbf{U}|_0) = \{\mathbf{U} : L(\mathbf{U}) \leq L(\mathbf{U}|_0)\}$, where $\mathbf{U}|_0$ is the initial control policy, is bounded for any $\mathbf{U}|_0 \in \mathbb{R}^{Nm}$.

The differential dynamic programming (DDP) method, a combination of the dynamic programming (DP) method and Newton's method, is the first algorithm which takes advantage of the special structure of the general UDOCP. The DDP method was introduced by Mayne [10] in 1966, and was further developed later by Jacobson and Mayne [6] in 1970. The DDP method does not require discretization of the control or state space. Hence, it overcomes the dimensionality problem of DP. Moreover, the DDP method has been proved to have q-quadratic local convergence under the standard assumptions [7, 12]. Besides these, when Mayne [10] introduced the DDP method, he suggested an inexact line search scheme for the DDP method as well. This was further refined by Jacobson and Mayne [6] in 1970. The line search scheme ensures the DDP method to be globally convergent [14].

However, the DDP method could break down when the problem is non-convex. In 1984, Yakowitz and co-workers [12, 14] employed a shifting technique from nonlinear programming (NLP) to deal with non-convex situations. They also proved that the DDP method with a shift scheme together with the standard inexact line search scheme ensures global convergence. Nevertheless, they did not mention how the shift should be computed. In 1991, Liao and Shoemaker [7] suggested an 'adaptive shift' procedure to determine the shift. Their procedure consists of two steps. In their Step 1, a constant shift is selected. While in their Step 2, an active shift is calculated for each stage by using the minimum eigenvalue of the Hessian matrix. They also proved that the DDP method with the adaptive shift will eventually converge quadratically. The adaptive shift, however, is problem dependent. In addition, shift schemes are inefficient when the shift is too large. The goal of this paper is to introduce a new scheme to overcome these shortfalls.

In the context of NLP, shift schemes can be viewed as a kind of compromises between Newton's method and the steepest descent (SD) method [9]. When solving some very nonlinear UDOCP, all existing shift schemes become inefficient, partly because the SD method was not used. This fact motivates the authors to develop a new efficient one for UDOCP by adopting the SD method.

The SD method for UDOCP was also sketched by Mayne [10] in 1966. Since its convergence rate is only linear, it is rarely used in numerical implementation. However, when the second derivatives could not provide useful information, the SD method becomes attractive. In this paper, the SD method is revised so that it can be combined with the DDP method with a shift scheme to form a new efficient method for solving the general UDOCP.

This paper is organized as follows. Following this introduction, a globally convergent DDP method with a shift scheme is summarized in Section 2. In Section 3, the SD method for UDOCP is addressed. A modified algorithm together with a global convergence analysis for the SD met hod is provided. In Section 4, a new efficient method for solving the general UDOCP is proposed. This method is a combination of the DDP method with a shift scheme and the revised SD method. Global and local convergences of the new method are also proved. In Section 5, numerical results on eight test problems available in the literature are presented to show the robustness and efficiency of the new method. Finally, in Section 6, some concluding remarks are drawn.

2. The DDP method for UDOCP

The DDP method is an iterative method that combines the DP technique and Newton's method. In each iteration, a nominal control policy $\bar{\mathbf{U}}$ is given. The aim of each iteration is to find a successor control policy $\mathbf{U}$ so that the total loss $L(\mathbf{U}) < L(\bar{\mathbf{U}})$. This successor control policy will be used as the new nominal control policy for the next iteration unless some stopping criterion is met.

Generally speaking, the DDP method consists of two sweeps in each iteration: the backward sweep which computes the feedback laws and the forward sweep which updates the control policy and trajectory. These sweeps and some convergence results are summarized in the following sub-sections. Notice that all the functions, derivatives, vectors and matrices in the following are evaluated at the nominal control policy $\bar{\mathbf{U}}$ and the corresponding trajectory $\bar{\mathbf{X}}$.

2.1. THE BACKWARD SWEEP WITH A SHIFT SCHEME

The goal in the backward sweep is to find linear feedback laws for N individual stages,

$$\mathbf{u}_t = \bar{\mathbf{u}}_t + \rho_t + \mathbf{P}_t(\mathbf{x}_t - \bar{\mathbf{x}}_t), \quad t = 1, 2, \ldots, N, \tag{6}$$

where $\rho_t \in I\!R^m$ and $\mathbf{P}_t \in I\!R^{m\times n}$ are computed backwardly by solving a sequence of sub-problems. Suppose that $[\mathbf{u}]_i \in I\!R, i = 1, 2, \ldots, m,$ is the i-th element of any vector $\mathbf{u} \in I\!R^m$; $\nabla_{\mathbf{u}} g \in I\!R^m$ is the gradient vector of any function $g(\mathbf{u}) \in C^1(I\!R^m, I\!R)$; $\mathrm{J}_{\mathbf{u}} f \in I\!R^{n\times m}$ is the Jacobian matrix of any function $f(\mathbf{u}) \in C^1(I\!R^m, I\!R^n)$; and $\nabla^2_{\mathbf{u},\mathbf{u}} g \in I\!R^{m\times m}$, $\nabla^2_{\mathbf{x},\mathbf{u}} g \in I\!R^{n\times m}$, $\nabla^2_{\mathbf{u},\mathbf{x}} g \in I\!R^{m\times n}$ and $\nabla^2_{\mathbf{x},\mathbf{x}} g \in I\!R^{n\times n}$ are the second-order derivatives of any function $g(\mathbf{x}, \mathbf{u}) \in C^2(I\!R^{n\times m}, I\!R)$, where $\mathbf{x} \in I\!R^n$ and $\mathbf{u} \in I\!R^m$. The following procedure is the backward sweep of the DDP method with a shift scheme presented by Yakowitz and Rutherford [14]. The aim of using a shift scheme is to deal with non-convex situations.

2.1.1. *The backward sweep with a shift scheme:*

1. Initialize the following variables,

$$\theta_{N+1} = 0, \tag{7}$$
$$\sigma_{N+1} = \nabla_{\mathbf{x}_{N+1}} g_{N+1}, \tag{8}$$
$$\mathbf{S}_{N+1} = \nabla^2_{\mathbf{x}_{N+1},\mathbf{x}_{N+1}} g_{N+1}. \tag{9}$$

2. For stage $t = N, N-1, \ldots, 1$, compute the followings,

$$\alpha_t = \nabla_{\mathbf{x}_t} g_t + (\mathrm{J}_{\mathbf{x}_t} f_t)^T \sigma_{t+1}, \tag{10}$$
$$\beta_t = \nabla_{\mathbf{u}_t} g_t + (\mathrm{J}_{\mathbf{u}_t} f_t)^T \sigma_{t+1}, \tag{11}$$
$$\mathbf{A}_t = (\mathrm{J}_{\mathbf{x}_t} f_t)^T \mathbf{S}_{t+1} (\mathrm{J}_{\mathbf{x}_t} f_t) + \nabla^2_{\mathbf{x}_t,\mathbf{x}_t} g_t + \sum_{i=1}^{n} [\sigma_{t+1}]_i (\nabla^2_{\mathbf{x}_t,\mathbf{x}_t} [f_t]_i), \tag{12}$$
$$\mathbf{B}_t = (\mathrm{J}_{\mathbf{u}_t} f_t)^T \mathbf{S}_{t+1} (\mathrm{J}_{\mathbf{x}_t} f_t) + \nabla^2_{\mathbf{u}_t,\mathbf{x}_t} g_t + \sum_{i=1}^{n} [\sigma_{t+1}]_i (\nabla^2_{\mathbf{u}_t,\mathbf{x}_t} [f_t]_i), \tag{13}$$
$$\mathbf{C}_t = (\mathrm{J}_{\mathbf{u}_t} f_t)^T \mathbf{S}_{t+1} (\mathrm{J}_{\mathbf{u}_t} f_t) + \nabla^2_{\mathbf{u}_t,\mathbf{u}_t} g_t + \sum_{i=1}^{n} [\sigma_{t+1}]_i (\nabla^2_{\mathbf{u}_t,\mathbf{u}_t} [f_t]_i) + s_t \cdot \mathbf{I}_m, \tag{14}$$
$$\rho_t = -\mathbf{C}_t^{-1} \beta_t, \tag{15}$$
$$\mathbf{P}_t = -\mathbf{C}_t^{-1} \mathbf{B}_t, \tag{16}$$
$$\theta_t = \theta_{t+1} - \rho_t^T \beta_t, \tag{17}$$
$$\sigma_t = \alpha_t + \mathbf{P}_t^T \beta_t, \tag{18}$$
$$\mathbf{S}_t = \mathbf{A}_t + \mathbf{P}_t^T \mathbf{B}_t, \tag{19}$$

where s_t is a shift parameter to ensure that $\mathbf{C}_t$ is positive definite.

2.2. THE FORWARD SWEEP WITH A LINE SEARCH SCHEME

In the forward sweep, the nominal control policy and the corresponding trajectory are updated based upon the feedback laws obtained in the backward sweep and the transition functions. The following procedure is the forward sweep of the DDP method with the inexact line search scheme in Jacobson and Mayne [6]. The aim of using a line search scheme is to ensure global convergence.

2.2.1. *The forward sweep with a line search scheme*

1. Select three small positive scalars $\bar{\theta}$, $\kappa \leq 0.01$ and $\bar{\lambda} \leq 0.01$.
2. Initialize the line search parameter $\lambda = 1$.

3. Repeat the following inexact line search procedure until the control policy for this iteration is found.
   (a) Forward sweep:
      (i) Set $\mathbf{x}_1(\lambda) \equiv \bar{\mathbf{x}}_1$.
      (ii) For stage $t = 1, 2, \ldots, N$, compute:

$$\mathbf{u}_t(\lambda) = \bar{\mathbf{u}}_t + \lambda\, \rho_t + \mathbf{P}_t[\mathbf{x}_t(\lambda) - \bar{\mathbf{x}}_t], \tag{20}$$

$$\mathbf{x}_{t+1}(\lambda) = f_t\big(\mathbf{x}_t(\lambda), \mathbf{u}_t(\lambda)\big). \tag{21}$$

   (b) Compute the loss $L\big(\mathbf{U}(\lambda)\big)$.
   (c) Check the reduction in loss:

   If the right-hand extreme of the Goldstein and Armijo conditions is satisfied, that is,

$$L\big(\mathbf{U}(\lambda)\big) - L\big(\bar{\mathbf{U}}\big) \leq -\lambda \cdot \kappa \cdot \theta_1, \tag{22}$$

   then the control policy for this iteration is found. So, let

$$\mathbf{U} = \mathbf{U}(\lambda), \tag{23}$$

$$\mathbf{X} = \mathbf{X}(\lambda), \tag{24}$$

   and stop the inexact line search procedure. Furthermore, if

$$\theta_1 \leq \bar{\theta} \cdot \max\{1, |L(\mathbf{U})|\}, \tag{25}$$

   then $\mathbf{U}$ is the optimal control policy.
   Otherwise, the inexact line search procedure should be continued by cutting the step size, such as, setting $\lambda = \lambda/2$ and go to a). However, if the new step size is too small, that is

$$\lambda < \bar{\lambda}, \tag{26}$$

   then the inexact line search procedure fails to reduce the total loss while satisfying (22). Then the algorithm fails.

2.3. GLOBAL CONVERGENCE AND LOCAL CONVERGENCE

Under some assumptions, the DDP method has been shown to converge globally [14] and locally with q-quadratic rate [7, 12]. The following lemmas ensure that the DDP method given above converges globally and is locally q-quadratically convergent.

THEOREM 2.1 *Suppose that the standard assumptions hold. Let* $\{\mathbf{U}|_k\}$ *be the sequence generated by the DDP method. Then any accumulation point of the control policy sequence* $\{\mathbf{U}|_k\}$ *is a stationary control policy with respect to* $L(\mathbf{U})$.

Proof. From the DDP method, it is easy to see that $\mathbf{U}|_k \in \Omega(\mathbf{U}|_0)$ for all k's. Therefore, from the standard assumptions, the two sequences $\{\mathbf{U}|_k\}$ and $\{\mathbf{X}|_k\}$ are bounded, where $\mathbf{X}|_k$ is the trajectory associated with $\mathbf{U}|_k$. Then from the standard assumptions and the DDP method, there exists an $M > 0$ such that $s_t \leq M$ for all t's in all iterations, where s_t is defined in (14). From the Theorem of [14] (p. 42), the result can be established. □

THEOREM 2.2 *Suppose that the standard assumptions hold, and g_t's and f_t's have continuous third-order derivatives in level set $\Omega(\mathbf{U}|_0)$ for any $\mathbf{U}|_0 \in I\!R^{Nm}$. Let $\{\mathbf{U}|_k\}$ be the sequence generated by the DDP method, and $\{\mathbf{U}|_k\}$ converge to $\mathbf{U}^*$ which is a solution of UDOCP, and $\mathbf{X}^*$ be the trajectory associated with $\mathbf{U}^*$. If $s_t \equiv 0$ for all t's and all k's, then the DDP method is locally q-quadratically convergent.*

Proof. From the DDP method, it is easy to see that $\mathbf{U}|_k \in \Omega(\mathbf{U}|_0)$ for all k's. Therefore, from the standard assumptions, the two sequences $\{\mathbf{U}|_k\}$ and $\{\mathbf{X}|_k\}$ are bounded, where $\mathbf{X}|_k$ is the trajectory associated with $\mathbf{U}|_k$. From the proof of Theorem 1 in [7], it can be easily seen that the closed bounded convex set D can be replaced by any closed bounded set D. Therefore, from Theorem 1 in [7], the result can be established. □

3. The SD method for UDOCP

The SD method for UDOCP proposed in this paper is an iterative method that combines the DP technique and the SD method for NLP. Similar to DDP, the SD method for UDOCP also consists of two sweeps in each iteration: the backward sweep and the forward sweep. These sweeps and the convergence analysis of the SD method are addressed in the following sub-sections.

3.1. THE BACKWARD SWEEP

Same as the DDP method, the goal in the backward sweep is to find control laws backwardly by solving a sequence of sub-problems. However, instead of using Newton's method at each stage in the DDP method, the SD method is adopted. Hence, the negative gradient of each stagewise objective function is taken as the search direction. The detail of the procedure is described as follows.

Let $\Delta\mathbf{u}_t \in I\!R^m$ and $\Delta\mathbf{x}_t \in I\!R^n$ be the perturbations at stage t about the nominal control and state, respectively, that is,

$$\Delta\mathbf{u}_t = \mathbf{u}_t - \bar{\mathbf{u}}_t, \quad t = 1, 2, \ldots, N, \tag{27}$$

$$\Delta\mathbf{x}_t = \mathbf{x}_t - \bar{\mathbf{x}}_t, \quad t = 1, 2, \ldots, N+1. \tag{28}$$

Let $R_{t+1}(\mathbf{x}_{t+1}) \in I\!R$ be the linear approximation to the cumulative loss from stage $t+1$ to stage $N+1$, that is,

$$R_{t+1}(\mathbf{x}_{t+1}) = r_{t+1} + \sigma_{t+1}^T \Delta\mathbf{x}_{t+1}, \tag{29}$$

where $r_{t+1} \in I\!R$ and $\sigma_{t+1} \in I\!R^n$. Let $V_t(\mathbf{x}_t, \mathbf{u}_t) \in I\!R$ be the cumulative value at stage t, that is,

$$V_t(\mathbf{x}_t, \mathbf{u}_t) = g_t(\mathbf{x}_t, \mathbf{u}_t) + R_{t+1}\big(f_t(\mathbf{x}_t, \mathbf{u}_t)\big). \tag{30}$$

Then, the linear approximation to $V_t(\mathbf{x}_t, \mathbf{u}_t)$ can be expressed as follows:

$$\mathrm{LP}(V_t)(\mathbf{x}_t, \mathbf{u}_t) = v_t + \alpha_t^T \Delta\mathbf{x}_t + \beta_t^T \Delta\mathbf{u}_t, \tag{31}$$

where

$$v_t = g_t + r_{t+1} \in I\!R, \tag{32}$$

$$\alpha_t = \nabla_{\mathbf{x}_t} g_t + (\mathrm{J}_{\mathbf{x}_t} f_t)^T \sigma_{t+1} \in I\!R^n, \tag{33}$$

$$\beta_t = \nabla_{\mathbf{u}_t} g_t + (\mathrm{J}_{\mathbf{u}_t} f_t)^T \sigma_{t+1} \in I\!R^m. \tag{34}$$

From the DP decomposition, the sub-problem at stage t is:

$$\min_{\mathbf{u}_t} \quad V_t(\mathbf{x}_t, \mathbf{u}_t). \tag{35}$$

Hence, to solve the sub-problem by the SD method, the search direction is taken as $\frac{-\beta_t}{\|\beta_t\|_2}$. (Notice that $\beta_t \neq 0$, otherwise $\bar{\mathbf{u}}_t$ is the solution to the sub-problem.) Thus the control law can be written as:

$$\mathbf{u}_t = \bar{\mathbf{u}}_t - \lambda \left(\frac{\beta_t}{\|\beta_t\|_2} \right), \tag{36}$$

where λ ($0 < \bar{\lambda} \leq \lambda \leq 1$) is the line search parameter of the SD method; $\bar{\lambda}$ is a small fixed parameter. Therefore, the linear approximation to the cumulative loss from stage t to stage $N+1$ can be obtained by substituting (36) into (31). This gives

$$R_t(\mathbf{x}_t) = r_t + \sigma_t^T \Delta\mathbf{x}_t, \tag{37}$$

where

$$r_t = v_t - \lambda\|\beta_t\|_2 \in I\!R, \tag{38}$$

$$\sigma_t = \alpha_t \in I\!R^n. \tag{39}$$

Hence, assuming $R_{N+2}(\mathbf{x}_{N+2}) = 0$ and repeating the above steps from stage $N+1$ to stage 1 backwardly, N control laws can be found. In summary, the backward sweep of the SD method can be written as follows:

3.1.1. *The backward sweep*

1. Initialize the following variables,

$$\theta_{N+1} = 0, \tag{40}$$
$$\sigma_{N+1} = \nabla_{\mathbf{x}_{N+1}} g_{N+1}. \tag{41}$$

2. For stage $t = N, N-1, \ldots, 1$, compute the followings,

$$\alpha_t = \nabla_{\mathbf{x}_t} g_t + (\mathrm{J}_{\mathbf{x}_t} f_t)^T \sigma_{t+1}, \tag{42}$$
$$\beta_t = \nabla_{\mathbf{u}_t} g_t + (\mathrm{J}_{\mathbf{u}_t} f_t)^T \sigma_{t+1}, \tag{43}$$
$$\theta_t = \theta_{t+1} + \|\beta_t\|_2, \tag{44}$$
$$\sigma_t = \alpha_t. \tag{45}$$

3.2. THE FORWARD SWEEP WITH A LINE SEARCH SCHEME

To ensure the global convergen ce, some line search scheme is necessary. To develop an inexact line search rule for the SD method based on the Goldstein and Armijo Conditions, the following lemmas and theorems are important. Notice that all the functions, derivatives, vectors and matrices in the following are evaluated at a fixed control policy and the corresponding trajectory.

LEMMA 3.1

$$\nabla_{\mathbf{x}_t} L = \begin{cases} \nabla_{\mathbf{x}_t} g_t + (\mathrm{J}_{\mathbf{x}_t} f_t)^T (\nabla_{\mathbf{x}_{t+1}} L), & t = 1, 2, \ldots, N, \\ \nabla_{\mathbf{x}_{N+1}} g_{N+1}, & t = N+1. \end{cases} \tag{46}$$

Proof. For $t = 1, 2, \ldots, N$, it is straightforward to see

$$\begin{aligned} \nabla_{\mathbf{x}_t} L &= \sum_{i=t}^{N+1} \nabla_{\mathbf{x}_t} g_i \\ &= \nabla_{\mathbf{x}_t} g_t + \sum_{i=t+1}^{N+1} \left(\frac{\partial\, \mathbf{x}_{t+1}}{\partial\, \mathbf{x}_t} \right)^T \left(\frac{\partial\, g_i}{\partial\, \mathbf{x}_{t+1}} \right)^T \\ &= \nabla_{\mathbf{x}_t} g_t + (\mathrm{J}_{\mathbf{x}_t} f_t)^T (\nabla_{\mathbf{x}_{t+1}} L). \end{aligned}$$

Therefore (46) can be obtained. □

LEMMA 3.2 *For* $t = 1, 2, \ldots, N$,

$$\nabla_{\mathbf{u}_t} L = \nabla_{\mathbf{u}_t} g_t + (\mathrm{J}_{\mathbf{u}_t} f_t)^T (\nabla_{\mathbf{x}_{t+1}} L). \tag{47}$$

Proof. For $t = 1, 2, \ldots, N$, it is straightforward to see

$$\begin{aligned}\nabla_{\mathbf{u}_t} L &= \sum_{i=t}^{N} \nabla_{\mathbf{u}_t} g_i \\ &= \nabla_{\mathbf{u}_t} g_t + \sum_{i=t+1}^{N} \left(\frac{\partial\, \mathbf{x}_{t+1}}{\partial\, \mathbf{u}_t}\right)^T \left(\frac{\partial\, g_i}{\partial\, \mathbf{x}_{t+1}}\right)^T \\ &= \nabla_{\mathbf{u}_t} g_t + (\mathrm{J}_{\mathbf{u}_t} f_t)^T (\nabla_{\mathbf{x}_{t+1}} L).\end{aligned}$$

Therefore (47) can be obtained. □

LEMMA 3.3 *Let* $\theta_t = -\frac{\mathrm{d}\, r_t}{\mathrm{d}\, \lambda}$, $t = 1, 2, \ldots, N$, *where* r_t *is obtained by* (38), *then*

$$\theta_t = \sum_{i=t}^{N} \|\beta_i\|_2 \quad \textit{and} \quad \theta_1 \geq \theta_2 \geq \cdots \geq \theta_N \geq 0. \tag{48}$$

Proof. From (38) and (32), we have

$$\begin{aligned} r_t &= (g_t - \lambda \|\beta_t\|_2) + r_{t+1} \\ &= \sum_{i=t}^{N} g_i - \lambda \sum_{i=t}^{N} \|\beta_i\|_2 + r_{N+1} \\ &= \sum_{i=t}^{N+1} g_i - \lambda \sum_{i=t}^{N} \|\beta_i\|_2. \end{aligned} \tag{49}$$

Thus,

$$\frac{\mathrm{d}\, r_t}{\mathrm{d}\, \lambda} = -\sum_{i=t}^{N} \|\beta_i\|_2. \tag{50}$$

This proves the results. □

LEMMA 3.4

$$\nabla_{\mathrm{U}} L = 0 \iff \beta_t = 0,\ t = 1, 2, \ldots, N \iff \theta_1 = 0. \tag{51}$$

Proof. First we prove

$$\sigma_t = \nabla_{\mathbf{x}_t} L \quad \text{for } t = 1, \cdots, N+1 \tag{52}$$

by induction backwardly in t. From (41) and Lemma 3.1, we have

$$\sigma_{N+1} = \nabla_{\mathbf{x}_{N+1}} g_{N+1} = \nabla_{\mathbf{x}_{N+1}} L. \tag{53}$$

Now, we assume that

$$\sigma_{t+1} = \nabla_{\mathbf{x}_{t+1}} L. \tag{54}$$

Then by Lemma 3.2 and (43),

$$\begin{aligned}\nabla_{\mathbf{u}_t} L &= \nabla_{\mathbf{u}_t} g_t + (\mathrm{J}_{\mathbf{u}_t} f_t)^T (\nabla_{\mathbf{x}_{t+1}} L) \\ &= \nabla_{\mathbf{u}_t} g_t + (\mathrm{J}_{\mathbf{u}_t} f_t)^T \sigma_{t+1} \\ &= \beta_t. \end{aligned} \tag{55}$$

Moreover, by (45), (42), and Lemma 3.1,

$$\begin{aligned}\sigma_t &= \alpha_t \\ &= \nabla_{\mathbf{x}_t} g_t + (\mathrm{J}_{\mathbf{x}_t} f_t)^T \sigma_{t+1} \\ &= \nabla_{\mathbf{x}_t} g_t + (\mathrm{J}_{\mathbf{x}_t} f_t)^T (\nabla_{\mathbf{x}_{t+1}} L) \\ &= \nabla_{\mathbf{x}_t} L. \end{aligned} \tag{56}$$

Thus, by the induction principle, (52) is true. Therefore, (52) and (55) imply that the first part of the theorem holds.

Now, the remaining part of the theorem can be shown easily. From Lemma 3.3,

$$\theta_1 = \sum_{t=1}^{N} \|\beta_t\|_2. \tag{57}$$

Therefore,

$$\begin{aligned}\theta_1 = 0 &\Longleftrightarrow \|\beta_t\|_2 = 0, t = 1, 2, \ldots, N \\ &\Longleftrightarrow \beta_t = 0, t = 1, 2, \ldots, N. \quad \square \end{aligned}$$

From the results of Lemma 3.3 and Lemma 3.4, the inexact line search rule based on the Goldstein and Armijo conditions for the SD method can be written as follows:

$$L(\mathbf{U}) - L(\bar{\mathbf{U}}) \leq -\lambda \cdot \kappa \cdot \theta_1, \tag{58}$$

where $\kappa \in (0, 0.01)$ is a small fixed parameters. Notice that, (58) is the right-hand extreme of the conventional Goldstein and Armijo conditions only. The left-hand extreme of the conventional Goldstein and Armijo conditions is not necessary because the use of a backtracking strategy can avoid excessively small steps. In conclusion, the forward sweep of the globally convergent SD method can be written as follows:

3.2.1. *The forward sweep with a line search scheme:*

1. Select three small positive scalars $\bar{\theta}$, $\kappa \leq 0.01$ and $\bar{\lambda} \leq 0.01$.
2. Initialize the line search parameter $\lambda = 1$.
3. Repeat the following inexact line search procedure until the control policy for this iteration is found.

(a) Forward sweep:
(i) Set $\mathbf{x}_1(\lambda) \equiv \bar{\mathbf{x}}_1$.

(ii) For stage $t = 1, 2, \ldots, N$, compute:

$$\mathbf{u}_t(\lambda) = \begin{cases} \bar{\mathbf{u}}_t - \lambda \frac{\beta_t}{\|\beta_t\|_2}, & \text{if } \beta_t \neq 0, \\ \bar{\mathbf{u}}_t, & \text{if } \beta_t = 0, \end{cases} \tag{59}$$

$$\mathbf{x}_{t+1}(\lambda) = f_t\big(\mathbf{x}_t(\lambda), \mathbf{u}_t(\lambda)\big). \tag{60}$$

(b) Compute the loss $L\big(\mathbf{U}(\lambda)\big)$.
(c) Check the reduction in loss:

If the right-hand extreme of the Goldstein and Armijo conditions is satisfied, that is,

$$L\big(\mathbf{U}(\lambda)\big) - L\big(\bar{\mathbf{U}}\big) \leq -\lambda \cdot \kappa \cdot \theta_1, \tag{61}$$

then the control policy for this iteration is found. So, let

$$\mathbf{U} = \mathbf{U}(\lambda), \tag{62}$$

$$\mathbf{X} = \mathbf{X}(\lambda), \tag{63}$$

and stop the inexact line search procedure. Furthermore, if

$$\theta_1 \leq \bar{\theta} \cdot \max\{1, |L(\mathbf{U})|\}, \tag{64}$$

then $\mathbf{U}$ is the optimal control policy.
Otherwise, the inexact line search procedure should be continued by cutting the step size, such as, setting $\lambda = \lambda/2$. However, if the new step size is too small, that is

$$\lambda < \bar{\lambda}, \tag{65}$$

then the inexact l ine search procedure fails to reduce the total loss while satisfying (61).

3.3. CONVERGENCE ANALYSIS

Since the SD method is a stagewise NLP procedure, so the theorem of global convergence of descent methods in NLP can be applied (Theorem 2.5.1 in [4]). The following theorem is a revised version for the SD method.

THEOREM 3.1 *Suppose that the standard assumptions hold. Let* $\{\mathbf{U}|_k\}$ *be the sequence generated by the SD method for UDOCP, where* k *is the iteration index. Then the SD method either stops in finitely many iterations with* $\{\nabla_{\mathbf{U}} L|_k\} = 0$ *for some* k *or generates an infinite sequence such that* $\{\nabla_{\mathbf{U}} L|_k\} \to 0$.

Proof. From the SD method, it is easy to see that $\mathbf{U}|_k \in \Omega(\mathbf{U}|_0)$ for all k's. Therefore, from the standard assumptions, the two sequences $\{\mathbf{U}|_k\}$ and $\{\mathbf{X}|_k\}$ are bounded, where $\mathbf{X}|_k$ is the trajectory associated with $\mathbf{U}|_k$.

Now we verify that $\{\mathbf{U}|_k\}$ satisfies the Goldstein and Armijo conditions (58). Equations (55), (57), (59), and the Taylor expansion of $L(\mathbf{U})$ indicate

$$L(\mathbf{U}|_{k+1}) - L(\mathbf{U}|_k) = -\lambda\theta_1 + o(\|\mathbf{U}|_{k+1} - \mathbf{U}|_k\|). \tag{66}$$

Since $\kappa \in (0, 0.01)$, (59) indicates that by choosing a proper λ, (58) can be always satisfied.

In addition, the standard assumptions imply that $\nabla_{\mathbf{U}} L|_k$ is uniformly continuous in $\Omega(\mathbf{U}|_0)$. Then from Theorem 2.5.1 in [4], the results of the theorem hold. □

4. A combined method for UDOCP

In this section, a new method that combines the DDP method with a shift scheme and the SD method is proposed to solve the general UDOCP. The algorithm and its convergence analysis are addressed in the following sub-sections.

4.1. THE ALGORITHM

The new method consists of two parts: the DDP method with a shift scheme and the SD method. In each part, two sweeps are involved: the backward sweep and the forward sweep. The algorithm of the new method is listed as follows. The key strategy of the method is to adopt the DDP method with a shift scheme whenever possible.

1. The DDP method with a shift scheme:
   (a) The backward sweep:
   Perform the backward sweep as stated in Section 2.1 except when the shift s_t must be greater than a pre-selected non-negative scalar $\hat{s}$ to guarantee the positive definiteness of all $\mathbf{C}_t$'s. Whenever this occurs, the iteration is restarted by using the SD Method.
   (b) The forward sweep:
   Perform the forward sweep as stated in Section 2.2.
2. The SD method:
   (a) The backward sweep:
   Perform the backward sweep as stated in Section 3.1.
   (b) The forward sweep:
   Perform the forward sweep as stated in Section 3.2.

4.2. CONVERGENCE ANALYSIS

To prove the global convergence of the proposed method, the following lemmas for the DDP method are required.

LEMMA 4.1 *For θ_t obtained in* (17)*, we have*

$$\theta_1 \geq \theta_2 \geq \cdots \geq \theta_N \geq 0. \tag{67}$$

Proof. From (17) and (15),

$$\theta_t = \sum_{i=t}^{N} \beta_i^T \mathbf{C}_i^{-1} \beta_i, \quad t = 1, 2, \ldots, N. \tag{68}$$

Since $\mathbf{C}_t,\ t = 1, 2, \ldots, N$, are positive definite, the result follows directly. □

LEMMA 4.2

$$\nabla_{\mathbf{U}} L = 0 \iff \beta_t = 0,\ t = 1, 2, \ldots, N \iff \theta_1 = 0. \tag{69}$$

Proof. Following exactly the same steps as discussed in the proof of Lemma 3.4, this lemma can be proved easily. □

The following theorems provide the global convergence and locally q-quadratic convergence of the proposed method described in Section 4.1.

THEOREM 4.1 *Suppose that the standard assumptions hold. Let $\{\mathbf{U}|_k\}$ be the sequence generated by the method described in Section 4.1 for UDOCP, where k is the iteration index. Then the method described in Section 4.1 either stops in finitely many iterations or generates an infinite sequence such that $\{\nabla_{\mathbf{U}} L|_k\} \to 0$.*

Proof. From the method described in Section 4.1, it is easy to see that $\mathbf{U}|_k \in \Omega(\mathbf{U}|_0)$ for all k's. Therefore, from the standard assumptions, the two sequences $\{\mathbf{U}|_k\}$ and $\{\mathbf{X}|_k\}$ are bounded, where $\mathbf{X}|_k$ is the trajectory associated with $\mathbf{U}|_k$. Therefore, $\nabla_{\mathbf{U}} L|_k$ is uniformly continuous in $\Omega(\mathbf{U}|_0)$. In addition, the positive definiteness of $\mathbf{C}_t$ matrices and the definition of positive scalar $\hat{s}$ guarantee that the angle between $\mathbf{U}|_{k+1} - \mathbf{U}|_k$ and $-\nabla_{\mathbf{U}} L(\mathbf{U}|_k)$ is uniformly bounded away from orthogonality.

Mayne in [10] (equation (26)) has achieved the following result

$$L(\mathbf{U}|_{k+1}) - L(\mathbf{U}|_k) = -\lambda(1 - \frac{\lambda}{2})\theta_1 + o(\lambda^2). \tag{70}$$

This ensures that (22), or the Goldstein and Armijo conditions, holds for DDP. On the other hand, (66) guarantees that (58), or the Goldstein and Armijo conditions, holds for the SD method. Therefore, we can say that the sequence $\{\mathbf{U}|_k\}$ satisfies the Goldstein and Armijo conditions.

The above discussions indicate that the conditions of Theorem 2.5.1 in [4] are all satisfied. Therefore, using Theorem 2.5.1 in [4], our results hold. □

THEOREM 4.2 *Suppose that the standard assumptions hold, and g_t's and f_t's have continuous third-order derivatives in level set $\Omega(\mathbf{U}|_0)$ for any $\mathbf{U}|_0 \in I\!R^{Nm}$. Let $\{\mathbf{U}|_k\}$ be the sequence generated by the method described in Section 4.1, and $\{\mathbf{U}|_k\}$ converge to $\mathbf{U}^*$ which is a solution of UDOCP, and $\mathbf{X}^*$ be the trajectory associated with $\mathbf{U}^*$. If $s_t \equiv 0$ for all t's and all k's in a neighborhood of $\mathbf{U}^*$, then the method described in Section 4.1 is locally q-quadratically convergent.*

Proof. From the principle of the combined method, we know that the DDP method with a shift scheme will be adopted whenever possible. In addition, the assumption that $\{\mathbf{U}|_k\}$ converge to $\mathbf{U}^*$ and $s_t \equiv 0$ for all t's and all k's in a neighborhood of $\mathbf{U}^*$ ensures that the DDP method with a shift scheme should be used in that neighborhood. Thus by Theorem 2.2, the proposed combined method is locally q-quadratically convergent. □

5. Numerical experiment

In this section, the proposed method is examined on eight test problems available in the literature. These test problems are provided in the Appendix. The objectives of the experiment are to show the robustness and efficiency of the proposed method.

To implement the tests, the parameter of the Goldstein and Armijo conditions, κ, is set to 1e-8; the tolerance of line search, $\bar{\lambda}$, is set to 1e-10; and the parameter of the stopping criterion, $\bar{\theta}$, is set to 1e-12.

For each test problem, 144 tests are carried out with different combinations of three parameters:

1. The number of decision times N: Four choices of the time stages are investigated, $N + 1 = 10$, 30, 50, and 100.
2. The starting point $\mathbf{U}|_0$: In order to have unbiased results, nine starting points are selected, $[\mathbf{U}|_0]_i = 0, \pm 0.2, \pm 0.4, \pm 1$ and ± 10, for $i = 1, 2, \ldots, Nm$.
3. The maximum shift allowed $\hat{s}$: To show the robustness and efficiency of the proposed method, $\hat{s}$ = 100, 1000, 10000, and $\hat{s} = \infty$ are selected. Notice that the case of $\hat{s} = \infty$ is just the conventional DDP method with a shift scheme.

The numerical results of the first three test problems with $N + 1 = 100$ are displayed in Tables I–III, respectively. Since the numerical results of the remaining five test problems share the similar behaviors, they have been omitted. In each table, the followings are reported:

Maximum shift:	$\max(s)$
Total number of iterations	
(Number of iterations using the SD Method):	$k(k^{\mathrm{SD}})$
Final θ_1:	$\theta_1\|_k$
Final Euclidean norm of the gradient of the total loss:	$\|\nabla_{\mathbf{U}} L\|_k\|_2$

6. Concluding remarks

In this paper, a new method that combines the DDP method with a shift scheme and the SD method is proposed to solve the general UDOCP. Based on our analysis and discussion in earlier sections, we can see that this new method en joys the following advantages.

First, the proposed method is robust. In Section 4.1, the proposed method is shown to have global convergence under the standard assumptions. From the numerical experiments reported in Section 5, all cases can converge to local minimizers rapidly. These results confirm the global convergence property. Furthermore, the proposed method can solve a problem when the DDP method with a shift scheme failed (see Table III). This shows the robustness and attractiveness of the proposed method.

In addition, the proposed method is extremely efficient. In Section 4.1, the proposed method is shown to be locally q-quadratically convergent under some assumptions. This property is also verified in our numerical experiment in Section 5. Besides the proposed method could save up to 99.9% in the number of iterations in the experiment (Table I). These results indicate the efficiency of the proposed method.

Since the Newton method for UDOCP shares the same computational structure as the DDP method [13], it is expected that the Newton method can be also combined with the SD method to form a new globally convergent and efficient method for solving UDOCP.

7. Acknowledgements

This research was partially supported by the Research Grants Council of Hong Kong under Grant CUHK4392/99E and Grant FRG/99-00/II-23 of Hong Kong Baptist University.

References

1. Bertsekas, D. P. (1982), Projected Newton methods for optimization problems with simple constraints, *SIAM Journal of Control and Optimization* 20, 221–246.
2. Dennis, J. E. and Schnabel, R. B. (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ.
3. Di Pillo, G., Grippo, L. and Lampariello, F. (1983), A class of structured quasi-Newton algorithms for optimal control problems, *IFAC Applications of Nonlinear Programming to Optimization and Control*, Edited by H. E. Rauch, International Federation of Automatic Control, Pergamon Press, NY, 101–107.
4. Fletcher, R. (1987), *Practical Methods of Optimization*, Second Edition, John Wiley and Sons, Chichester.
5. Gill, P. E., Murray, W. and Wright, M. H. (1981), *Practical Optimization*, Academic Press, London.
6. Jacobson, D. H. and Mayne, D. Q. (1970), *Differential Dynamic Programming*, American Elsevier, New York.

7. Liao, L. Z. and Shoemaker, C. A. (1991), Convergence in unconstrained discrete-time differential dynamic programming, *IEEE Transactions on Automatic Control* 36, 692–706.
8. Liao, L. Z. and Shoemaker, C. A. (1992), Comparison of differential dynamic programming and Newton's method for discrete-time optimal control Problems, Technical Report CTC92TR97, Cornell University.
9. Luenberger, D. G. (1989), *Linear and Nonlinear Programming*, Second Edition, Addison-Wesley, Reading, MA.
10. Mayne, D. Q. (1966), A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems, *International Journal of Control* 3:1, 85–95.
11. Murray, D. and Yakowitz, S. (1981), The application of optimal control methodology to nonlinear programming problems, *Mathematical Programming* 21, 331–347.
12. Murray, D. M. and Yakowitz, S. J. (1984), Differential dynamic programming and Newton's method for discrete optimal control problems, *Journal of Optimization Theory and Applications* 43:3, 395–414.
13. Pantoja, J. F. A. De O. (1988), Differential dynamic programming and Newton's method, *International Journal of Control* 47:5, 1539–1553.
14. Yakowitz, S. and Rutherford, B. (1984), Computational aspects of discrete-time optimal control, *Applied Mathematics and Computation* 15, 29–45.

Appendix

A. Testing problems

Problem 1: [8]

$$\min \quad \sum_{t=1}^{N} \left\{ \sum_{i=1}^{n} ([\mathbf{x}_t]_i + 0.25)^4 + \sum_{j=1}^{m} ([\mathbf{u}_t]_j + 0.5)^4 \right\} + \sum_{i=1}^{n} ([\mathbf{x}_{N+1}]_i + 0.25)^4$$

where

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mu \cdot (\mathbf{x}_t^T \mathbf{C}\mathbf{u}_t) \cdot \mathbf{e}, \quad t = 1, 2, \ldots, N,$$

$$\mathbf{x}_1 = [0, \ldots, 0]^T \in I\!R^n,$$

$$\mathbf{A} \in I\!R^{n\times n}, \quad [\mathbf{A}]_{i,k} = \begin{cases} 0.5 & \text{if } k = i, \\ 0.25 & \text{if } k = i+1, \\ -0.25 & \text{if } k = i-1, \end{cases} \quad i, k = 1, 2, \ldots, n,$$

$$\mathbf{B} \in I\!R^{n\times m}, \quad [\mathbf{B}]_{i,j} = \frac{i-j}{n+m}, \quad i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, m,$$

$$\mathbf{C} \in I\!R^{n\times m}, \quad [\mathbf{C}]_{i,j} = \frac{i+j}{n+m}, \quad i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, m,$$

$$\mathbf{e} = [1, \cdots, 1]^T \in I\!R^n,$$

$$\mu = 1, \quad n = 4, \quad m = 2.$$

Problem 2: [8]

$$\min \quad \sum_{t=1}^{N} \|\mathbf{x}_t\|_2^2 \left[\sin^2 \left(\frac{\|\mathbf{u}_t\|_2^2}{m} \right) + 1 \right] + \|\mathbf{x}_{N+1}\|_2^2$$

where

$$[\mathbf{x}_{t+1}]_i = \sin([\mathbf{x}_t]_i) + [\mathbf{K} \cdot W(\mathbf{u}_t)]_i, \quad i = 1, 2, \ldots, n, \ t = 1, 2, \ldots, N,$$
$$[\mathbf{x}_1]_i = \tfrac{i}{(2n)}, \quad i = 1, 2, \ldots, n,$$
$$\mathbf{K} \in I\!R^{n \times m}, \quad [\mathbf{K}]_{i,j} = \tfrac{(i+j)}{(2n)}, \quad i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, m,$$
$$W : I\!R^m \mapsto I\!R^m, \quad [W(\mathbf{u}_t)]_j = \sin([\mathbf{u}_t]_j), \quad j = 1, 2, \ldots, m,$$
$$n = 4, \quad m = 2.$$

Problem 3: [14]

$$\min \quad \sum_{t=1}^{N} \exp\left(\|\mathbf{x}_t\|_2^2\right) \left[\sin^2 \left(\frac{\|\mathbf{u}_t\|_2^2}{m} \right) + 1 \right] + \exp\left(\|\mathbf{x}_{N+1}\|_2^2\right)$$

where

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{K} \cdot W(\mathbf{u}_t), \quad t = 1, 2, \ldots, N,$$
$$[\mathbf{x}_1]_i = \tfrac{i}{20}, \quad i = 1, 2, \ldots, n,$$
$$\mathbf{K} \in I\!R^{n \times m}, \quad [\mathbf{K}]_{i,j} = \tfrac{(i+j)}{100}, \quad i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, m,$$
$$W : I\!R^m \mapsto I\!R^m, \quad [W(\mathbf{u}_t)]_j = \sin([\mathbf{u}_t]_j), \quad j = 1, 2, \ldots, m,$$
$$n = 5, \quad m = 5.$$

Problem 4: [1]

$$\min \quad \frac{k}{2} \sum_{t=1}^{N} (\mathbf{x}_t^T \mathbf{Q} \mathbf{x}_t + r u_t^2) + \frac{k}{2} \mathbf{x}_{N+1}^T \mathbf{Q} \mathbf{x}_{N+1}$$

where

$$\mathbf{x}_{t+1} = \begin{bmatrix} rr1 & k \\ -k & 1 \end{bmatrix} \mathbf{x}_t + \begin{bmatrix} r0 \\ k \end{bmatrix} u_t, \quad t = 1, 2, \ldots, N,$$
$$\mathbf{x}_1 = \begin{bmatrix} r15 \\ 5 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} rr2 & 0 \\ 0 & 1 \end{bmatrix}, \quad r = 6, \quad k = \tfrac{1}{(N+1)},$$
$$n = 2, \quad m = 1.$$

Problem 5: [3]

$$\min \quad \frac{5k}{2}\|\mathbf{x}_1\|_2^2 + 5ku_1^2 + 5k\sum_{t=2}^{N}\left(\|\mathbf{x}_t\|_2^2 + u_t^2\right) + \frac{5k}{2}\|\mathbf{x}_{N+1}\|_2^2$$

where

$$\mathbf{x}_{t+1} = \mathbf{x}_t + 5k\begin{bmatrix} c\left[1-([\mathbf{x}_t]_2)^2\right][\mathbf{x}_t]_1 - [\mathbf{x}_t]_2 + \mathbf{u}_t \\ [\mathbf{x}_t]_1 \end{bmatrix}, \quad t = 1, 2, \ldots, N,$$
$$\mathbf{x}_1 = [0, 1]^T, \quad k = \tfrac{1}{(N+1)},$$
$$n = 2, \quad m = 1.$$

Problem 6: [3]

$$\min \quad k\sum_{t=1}^{N}(x_t^2 + u_t^2)$$

where

$$x_{t+1} = x_t + k\cdot(x_t^2 - u_t), \quad t = 1, 2, \ldots, N-1,$$
$$x_1 = 1, \quad k = \tfrac{1}{(N+1)},$$
$$n = 1, \quad m = 1.$$

Problem 7: [11]

$$\min \quad \frac{1}{2}\sum_{t=1}^{N}\left\{\left[x_t + \exp(u_t)\right]^2 + u_t^2\right\}$$

where

$$x_{t+1} = x_t + \exp(u_t), \quad t = 1, 2, \ldots, N-1,$$
$$x_1 = 0,$$
$$n = 1, \quad m = 1.$$

Problem 8: [7]

$$\min \quad \sum_{t=1}^{N}\sin^2(x_t^2 + u_t^2) + \sin^2(x_{N+1}^2)$$

where

$$x_{t+1} = x_t^2 + x_t u_t, \quad t = 1, 2, \ldots, N,$$
$$x_1 = \tfrac{1}{2},$$
$$n = 1, \quad m = 1.$$

Table I. Numerical results of Problem 1 with N+1=100*

$[\mathbf{U}\|_0]_i, \forall i$	$\hat{s}$	max(s)	$k(k^{\mathrm{SD}})$	$\theta_1\|_k$	$\|\nabla_{\mathbf{U}} L\|_k\|_2$
0	∞	0	5 (0)	9e-15	8e-08
	100	0	5 (0)	9e-15	8e-08
	1000	0	5 (0)	9e-15	8e-08
	10000	0	5 (0)	9e-15	8e-08
0.2	∞	1e71	526 (0)	1e-13	3e-07
	100	0	5 (1)	5e-12	2e-06
	1000	0	5 (1)	5e-12	2e-06
	10000	0	5 (1)	5e-12	2e-06
−0.2	∞	0	4 (0)	4e-20	1e-10
	100	0	4 (0)	4e-20	1e-10
	1000	0	4 (0)	4e-20	1e-10
	10000	0	4 (0)	4e-20	1e-10
0.4	∞	1e163	937 (0)	1e-12	8e-07
	100	0	8 (2)	5e-17	6e-09
	1000	0	8 (2)	5e-17	6e-09
	10000	0	8 (2)	5e-17	6e-09
−0.4	∞	1e87	187 (0)	2e-17	4e-09
	100	0	6 (1)	2e-13	4e-07
	1000	0	6 (1)	2e-13	4e-0 7
	10000	0	6 (1)	2e-13	4e-07
1	∞	le301	†	8e-03	2e+16
	100	0	9 (3)	4e-17	6e-09
	1000	0	9 (3)	4e-17	6e-09
	10000	0	9 (3)	4e-17	6e-09
−1	∞	1e207	†	4e-01	1e+66
	100	1	10 (2)	3e-16	2e-08
	1000	1	10 (2)	3e-16	2e-08
	10000	1	10 (2)	3e-16	2e-08

*Computer overflow was observed for $[\mathbf{U}|_0]_i = \pm 10,\ i = 1, \cdots, n$.
† The stopping criterion was not reached after 10000 iterations.

Table II. Numerical results of Problem 2 with N+1=100

$[\mathbf{U}\|_0]_i, \forall i$	$\hat{s}$	$\max(s)$	$k(k^{\mathrm{SD}})$	$\theta_1\|_k$	$\|\nabla_{\mathbf{U}} L\|_k\|_2$
0	∞	1	7 (0)	2e-19	1e-09
	100	1	7 (0)	2e-19	1e-09
	1000	1	7 (0)	2e-19	1e-09
	10000	1	7 (0)	2e-19	1e-09
0.2	∞	10000	25 (0)	2e-14	2e-07
	100	100	11 (3)	1e-15	2e-08
	1000	1000	18 (2)	9e-17	3e-08
	10000	10000	25 (0)	2e-14	2e-07
-0.2	∞	10000	21 (0)	2e-15	2e-07
	100	100	73 (3)	1e-21	1e-10
	1000	1000	18 (1)	5e-13	2e-06
	10000	10000	21 (0)	2e-15	2e-07
0.4	∞	1000	30 (0)	5e-17	1e-08
	100	100	19 (2)	1e-17	4e-09
	1000	1000	30 (0)	5e-17	1e-08
	10000	1000	30 (0)	5e-17	1e-08
-0.4	∞	1000	239 (0)	3e-18	5e-09
	100	100	27 (2)	2e-15	7e-09
	1000	1000	239 (0)	3e-18	5e-09
	10000	1000	239 (0)	3e-18	5e-09
1	∞	1000	27 (0)	1e-27	5e-13
	100	100	30 (2)	1e-21	4e-10
	1000	1000	27 (0)	1e-27	5e-13
	10000	1000	27 (0)	1e-27	5e-13
-1	∞	1e12	687 (0)	2e-18	2e-09
	100	100	197 (154)	2e-18	9e-10
	1000	1000	416 (1)	6e-15	2e-07
	10000	10000	324 (1)	2e-15	1e-07
10	∞	1e12	2925 (0)	2e-13	4e-05
	100	100	764 (321)	1e-16	1e-08
	1000	1000	2255 (5)	1e-18	4e-10
	10000	10000	2699 (7)	6e-13	3e-07
-10	∞	1e9	2661 (0)	8e-18	1e-09
	100	100	1241 (29)	5e-16	3e-08
	1000	1000	1668 (7)	2e-18	2e-09
	10000	10000	2446 (3)	3e-16	1e-08

Table III. Numerical results of Problem 3 with N+1=100*

$[\mathbf{U}\|_0]_i, \forall i$	$\hat{s}$	max(s)	$k(k^{\mathrm{SD}})$	$\theta_1\|_k$	$\|\nabla_{\mathbf{U}} L\|_k\|_2$
0	∞	1	517 (0)	1e-10	1e-05
	100	1	517 (0)	1e-10	1e-05
	1000	1	517 (0)	1e-10	1e-05
	10000	1	517 (0)	1e-10	1e-05
0.2	∞	1e86	206 (0)	4e-11	7e-06
	100	10	13 (1)	4e-11	2e-08
	1000	10	13 (1)	4e-11	2e-08
	10000	10	13 (1)	4e-11	2e-08
−0.2	∞	1e78	2647 (0)	1e-10	1e-05
	100	100	17 (1)	3e-11	2e-08
	1000	100	17 (1)	3e-11	2e-08
	10000	100	17 (1)	3e-11	2e-08
−0.4	∞	Computer overflow while initialization			
	100	1	15 (3)	4e-11	7e-09
	1000	1	15 (3)	4e-11	7e-09
	10000	1	15 (3)	4e-11	7e-09

*Computer overflow was observed for $[\mathbf{U}|_0]_i = 0.4, \pm 1, \pm 10,\ i = 1, \cdots, n$.