Journal of the Operations Research
Society of Japan
Vol.21, No.3, Sept., 1978

DIFFERENTIAL DYNAMIC PROGRAMMING FOR SOLVING NONLINEAR PROGRAMMING PROBLEMS

Katsuhisa Ohno

Kyoto University

(Received August 29, 1977; Revised March 27, 1978)

Abstract Dynamic programming is one of the methods which utilize special structures of large-scale mathematical programming problems. Conventional dynamic programming, however, can hardly solve mathematical programming problems with many constraints. This paper proposes differential dynamic programming algorithms for solving large-scale nonlinear programming problems with many constraints and proves their local convergence. The present algorithms, based upon Kuhn-Tucker conditions for subproblems decomposed by dynamic programming, are composed of iterative methods for solving systems of nonlinear equations. It is shown that the convergence of the present algorithms with Newton's method is R-quadratic. Three numerical examples including the Rosen-Suzuki test problem show the efficiency of the present algorithms.

1. Introduction

Large-scale mathematical programming problems, as is well known, have special structures. Several decomposition and partitioning procedures for solving them [12] have been developed by utilizing their special structures. Dynamic programming also utilizes a similar structure of large-scale mathematical programming problems, but it admits more flexible structure than the decomposition and partitioning procedures do [16, 17, 19]. It, however, is hardly possible to solve mathematical programming problems with many constraints by conventional dynamic programming, even if they have the required structure. This is because each constraint yields one state variable and because conventional dynamic programming must compute optimal values of subproblems for every possible lattice points of state variables and must reserve them in high-speed memories. Although some state variable reduction methods have been developed [1], the difficulty of dimensionality never seems to disappear.

Jacobson and Mayne [10] have invented differential dynamic programming

(D.D.P.) for solving discrete and continuous time optimal control problems. Gershwin and Jacobson [5], Havira and Lewis [7] and Mayne [13] have discussed further D.D.P. for constrained optimal control problems. A method analogous to D.D.P. has been invented by Dyer and McReynolds [3]. The above authors, however, have not proved the convergence of their D.D.P. algorithms. Recently, Mayne and Polak [14, 24] have proposed first-order algorithms of the D.D.P. type for solving continuous time optimal control problems and have proved the convergence of their algorithms. It should be noted, however, that their algorithms are not based upon principle of optimality but based upon maximum principle, and are quite different from the first-order D.D.P. algorithms mentioned in [3, 10]. Ohno [20, 21] has devised a new D.D.P. algorithm for solving discrete time optimal control problems with constraints on both control and state variables, and has proved its local convergence.

As shown in the above, D.D.P. has been applied to optimal control problems. In a previous paper [22], a D.D.P. algorithm for solving separable programs has been devised and its local convergence has been proved. The main purposes of this paper are to propose D.D.P. algorithms for solving large-scale nonlinear programming problems including separable programs and to prove their local convergence. In Section 2 it is shown that under some conditions, nonlinear programming problems can be decomposed into subproblems by dynamic programming. Section 3 contains Kuhn-Tucker conditions for each subproblem and a basic lemma. A D.D.P. algorithm for solving large-scale nonlinear programming problems is devised in Section 4 and a combination of the D.D.P. algorithm with Newton's method is discussed in Section 5. A modified version of the D.D.P. algorithm is described in Section 6. Numerical examples are given in Section 7 and convergence proofs of the D.D.P. algorithms are given in Section 8.

2. Decomposition by Dynamic Programming

Let $x_n (n=1,2,\dots,N)$ be k_n-dimensional column vector. Consider the following nonlinear programming problem with angular structure [12]:

$$\text{(P)} \quad \text{minimize} \quad f(x_1,x_2,\dots,x_N)$$
$$\text{subject to } g^j(x_1,x_2,\dots,x_N)\le 0 \quad (j=1,\dots,m),$$
$$h_n^j(x_n)\le 0 \quad (j=1,\dots,m_n,\ n=1,\dots,N).$$

If equality constraints on $(x_1,x_2,\dots,x_N)$ or x_n are imposed on (P), the following analysis is valid with obvious changes. Define m and m_n-dimensional vector valued functions g and h_n as $(g^1,\dots,g^m)^T$ and $(h_n^1,\dots,h_n^{m_n})^T$, respectively, where T denotes the transposition. Then the feasible region of (P) denoted by X is represented as

$$X=\{(x_1,\dots,x_N);\ g(x_1,\dots,x_N)\le 0 \text{ and } h_n(x_n)\le 0 \quad (n=1,\dots,N)\}.$$

Since Problem (P) is too general to be decomposed by dynamic programming, it is assumed that [16]:

(C_1) There exist functions ξ_N: $R^{k_N}\to R^1$ and $\xi_n: R^{k_n}\times R^1\to R^1$ $(n=1,\dots,N-1)$ such that

$$f_N(x_N) = \xi_N(x_N),$$
$$f_n(x_n,\dots,x_N) = \xi_n(x_n, f_{n+1}(x_{n+1},\dots,x_N)) \quad (n=1,\dots,N-1)$$

and $f(x_1,\dots,x_N) = f_1(x_1,\dots,x_N)$,

where R^{k_n} denotes the k_n-dimentional Euclidean space and $\xi_n(\cdot,y)$ $(n=1,\dots,N-1)$ are monotone nondecreasing functions of y;

(C_2) There exist functions σ_1: $R^{k_1}\to R^m$ and σ_n: $R^m\times R^{k_n}\to R^m$ $(n=2,\dots,N)$ such that

$$g_1(x_1) = \sigma_1(x_1),$$
$$g_n(x_1,\dots,x_n) = \sigma_n(g_{n-1}(x_1,\dots,x_{n-1}),x_n) \quad (n=2,\dots,N)$$

and $g(x_1,\dots,x_N) = g_N(x_1,\dots,x_N)$.

It is clear that separable programs satisfy the above conditions. Moreover, almost all large-scale mathematical programming problems which have been discussed by many researchers [9, 12] also satisfy these conditions.

Now let us introduce m-dimentional state variables s_n $(n=0,1,\dots,N)$ as

(2.1) $$s_0 = 0 \quad\text{and}\quad s_n = g_n(x_1,\dots,x_n) \quad (n=1,\dots,N).$$

Then Condition (C_2) leads to the following difference equations:

(2.2) $$s_n = \sigma_n(s_{n-1},x_n) \quad (n=1,2,\dots,N),$$

where $\sigma_1(0,x_1) \equiv \sigma_1(x_1)$.

Denote by S_n $(n=1,\dots,N)$ the reachable set of s_n; that is,

$$S_n = \{s_n\in R^m;\ s_n = g_n(x_1,\dots,x_n),\ (x_1,\dots,x_N)\in X\}.$$

Clearly for any $n=1,\dots,N$,

(2.3) $$X = \bigcup_{s_{n-1}\in S_{n-1}} [\{(x_1,\dots,x_{n-1});\ s_{n-1}=g_{n-1}(x_1,\dots,x_{n-1}),\ h_i(x_i)\le 0\ (i=1,\dots,n-1)\}\times\{(x_n,\dots,x_N);\ s_{n-1}=g_{n-1}(x_1,\dots,x_{n-1}),\ g(x_1,\dots,x_N)\le 0,\ h_i(x_i)\le 0\ (i=n,\dots,N)\}].$$

Therefore Condition (C_1) implies that

$$(P)=\min\{\xi_1(x_1,\xi_2(\ldots,\xi_{n-1}(x_{n-1},f_n(x_n,\ldots,x_N))\ldots))\,|\,(x_1,\ldots,x_N)\in X\}$$
$$=\min_{s_{n-1}\in S_{n-1}}\{\xi_1(x_1,\xi_2(\ldots,\xi_{n-1}(x_{n-1},\min\{f_n\,|\,s_{n-1}=g_{n-1},g\le 0,$$
$$h_i\le 0\ (i=n,\ldots,N)\})\ldots))\,|\,s_{n-1}=g_{n-1},h_i\le 0\ (i=1,\ldots,n-1)\}.$$

This suggests that the following subproblem (P_n) should be dealt with:

$$(P_n)\qquad F_n(s_{n-1}) = \min\{f_n(x_n,\ldots,x_N)\,|\,s_{n-1}=g_{n-1}(x_1,\ldots,x_{n-1}),$$
$$g(x_1,\ldots,x_N)\le 0,\ h_i(x_i)\le 0\quad (i=n,\ldots,N)\},$$

where $s_{n-1}\in R^m$ and it is assumed that $F_n(s_{n-1})=\infty$ for s_{n-1} such that the feasible region of (P_n) is empty. In addition, suppose that $\xi_n(x_n,\infty)=\infty$ $(n=1,\ldots,N-1)$.

Theorem 1. Suppose that Conditions (C_1) and (C_2) are satisfied. Then for $n=1,\ldots,N-1$,

$$(2.4)\qquad F_n(s_{n-1}) = \min\{\xi_n(x_n,F_{n+1}(\sigma_n(s_{n-1},x_n)))\,|\,h_n(x_n)\le 0\}$$

and

$$(2.5)\qquad F_N(s_{N-1}) = \min\{\xi_N(x_N)\,|\,\sigma_N(s_{N-1},x_N)\le 0,\ h_N(x_N)\le 0\}.$$

Proof: Let us redefine S_n $(n=1,\ldots,N-1)$ as

$$S_n = \{s_n\in R_m;\ \text{there exists a } (x_{n+1},\ldots,x_N) \text{ such that } s_n=g_n,\ g\le 0,$$
$$h_i\le 0\ (i=n+1,\ldots,N)\}.$$

In a way similar to (2.3), for $s_{n-1}\in S_{n-1}$,

$$\{(x_n,\ldots,x_N);\ s_{n-1}=g_{n-1},\ g\le 0,\ h_i\le 0\ (i=n,\ldots,N)\}$$
$$=\bigcup_{s_n\in S_n}[\{x_n;\ \sigma_n(s_{n-1},x_n)=s_n,\ h_n(x_n)\le 0\}\times\{(x_{n+1},\ldots,x_N);\ s_n=g_n,$$
$$g\le 0,\ h_i(x_i)\le 0\ (i=n+1,\ldots,N)\}].$$

From this and Condition (C_1) it follows that for $s_{n-1}\in S_{n-1}$

$$F_n(s_{n-1}) = \min_{s_n\in S_n}\{\xi_n(x_n,\min\{f_{n+1}\,|\,s_n=g_n,\ g\le 0,\ h_i\le 0\ (i=n+1,\ldots,N)\})\,|$$
$$\sigma_n(s_{n-1},x_n)=s_n,\ h_n\le 0\}$$
$$= \min\{\xi_n(x_n,F_{n+1}(\sigma_n(s_{n-1},x_n)))\,|\,\sigma_n(s_{n-1},x_n)\in S_n,\ h_n\le 0\}$$
$$= \min\{\xi_n(x_n,F_{n+1}(\sigma_n(s_{n-1},x_n)))\,|\,h_n(x_n)\le 0\}.$$

Clearly, (2.4) holds for $s_{n-1}\notin S_{n-1}$, and (P_N) is reduced to (2.5).

Since $F_1(0)$ is identical with the optimal value of (P), Theorem 1 implies that every optimal solution of (P) can be obtained by solving first Subproblem (2.5) and solving (2.4) recursively for $n=N-1,\ldots,1$. That is, Problem (P) with $\sum_{n=1}^{N}k_n$-dimentional variable and $m+\sum_{n=1}^{N}m_n$ constraints has been decomposed into N subproblems with each k_n-dimensional variable and $m+m_N$ or m_n constraints.

Remark 1. The above decomposition of (P) is different from that in [16]. In [16], it is assumed that (C_1) and, instead of (C_2), (C_2') are satisfied:

(C_2') There exist functions $\sigma_N: R^{k_N} \to R^m$ and $\sigma_n: R^{k_n} \times R^m \to R^m$ $(n=1,\dots,N-1)$ such that

$$g_N(x_N) = \sigma_N(x_N)$$

$$g_n(x_n,\dots,x_N) = \sigma_n(x_n, g_{n+1}(x_{n+1},\dots,x_N)) \quad (n=1,\dots,N-1)$$

and $g(x_1,\dots,x_N) = g_1(x_1,\dots,x_N)$,

where for $n=1,\dots,N-1$ and $s=(s^1,\dots,s^m)\in R^m$, $\sigma_n(x_n,s)=(\sigma_n^1(x_n,s^1),\dots,\sigma_n^m(x_n,s^m))^T$ and $\sigma_n^j(x_n,s^j)$ $(j=1,\dots,m)$ are nondecreasing functions of s^j. Moreover, instead of (P_n), define for $n=1,\dots,N$,

(P_n') $\quad F_n(s_n) = \min\{f_n(x_n,\dots,x_N) \,|\, g_n(x_n,\dots,x_N)\le s_n,\ h_i(x_i)\le 0 \quad (i=n,\dots,N)\}$.

Then the following recurrence relations hold for $n=1,\dots,N-1$:

$$(2.6) \qquad F_n(s_n) = \min\{\xi_n(x_n, F_{n+1}(\sigma_n^{-1}(x_n,s_n))) \,|\, h_n(x_n)\le 0,\ x_n\in V_n\},$$

where $\sigma_n^{-1}(x_n,s_n)=\max\{s_{n+1}\in R^m;\ \sigma_n(x_n,s_{n+1})\le s_n\}$ and $V_n=\{x_n;$ there exists $\sigma_n^{-1}(x_n, s_n)$ for given $s_n\}$. Since (2.6) includes the function σ_n^{-1} and the set V_n, it is not easy to discuss (2.6) theoretically.

As noted above, (P) can be solved by using (2.5) and (2.4) recursively. However, it is almost impossible to solve (P) with $m\ge 3$ by using (2.5) and (2.4). This is because both the storage of $F_n(s_{n-1})$ for suitable lattice points of s_{n-1} and the comparisons of values $\xi_n(x_n, F_{n+1}(\sigma_n(s_{n-1},x_n)))$ at all x_n satisfying $h_n(x_n)\le 0$ for each lattice point of s_{n-1} are required. Thus an iterative method based on (2.5) and (2.4), which is called a D.D.P., will be developed in the following sections.

3. Kuhn-Tucker Conditions

Define the Lagrangian functions L_n $(n=1,\dots,N)$ for subproblems given by (2.4) and (2.5) as: for $n=1,\dots,N-1$,

$$(3.1) \qquad L_n(x_n,\lambda_n,s_{n-1}) = \xi_n(x_n, F_{n+1}(\sigma_n(s_{n-1},x_n))) + \lambda_n^T h_n(x_n)$$

and

$$(3.2) \qquad L_N(x_N,\lambda_N,\mu,s_{N-1}) = \xi_N(x_N) + \lambda_N^T h_N(x_N) + \mu^T \sigma_N(s_{N-1},x_N),$$

where λ_n and μ are m_n-dimensional and m-dimensional nonnegative Lagrange multipliers. To begin with, suppose that

(C_3) For each $n=1,\dots,N$, the function ξ_n, component functions σ_n^j $(j=1,\dots,m)$ of σ_n and component functions h_n^j $(j=1,\dots,m_n)$ of h_n are all twice differentiable functions and all their second derivatives are uniformly continuous.

For scalar functions, say, ξ_n denote by $\nabla_x\xi_n$ and $\nabla_x^2\xi_n$ the gradient row vector and the Hessian matrix of ξ_n with respect to x_n, respectively, and for vector functions, say, σ_n denote by $\nabla_x\sigma_n$ and $\nabla_x^2\sigma_n$ the Jacobian matrix and the second Frechet-derivative of σ_n with respect to x_n, respectively. That is, $\nabla_x\xi_n=(\partial\xi_n/\partial x_n^1,\ldots,\partial\xi_n/\partial x_n^{k_n})$, $\nabla_x^2\xi_n=(\partial^2\xi_n/\partial x_n^i\partial x_n^j)$, $\nabla_x\sigma_n=(\partial\sigma_n^i/\partial x_n^j)$ and for any m-dimensional vector z, $z^T\nabla_x^2\sigma_n=\sum_{j=1}^m z^j\nabla_x^2\sigma_n^j$. Note that gradient vectors are taken as row vectors in relation to Jacobian matrices.

Suppose that Problem (P) has an optimal solution $\{x_n^*;\ n=1,\ldots,N\}$. Then the optimal trajectory $\{s_n^*;\ n=1,\ldots,N\}$ corresponding to $\{x_n^*\}$ can be determined by (2.2). Moreover each subproblem (P_n) with $s_{n-1}=s_{n-1}^*$ has also the optimal solution $\{x_i^*;\ i=n,\ldots,N\}$, and hence the optimal value $F_n(s_{n-1}^*)$ is attained at $x_n=x_n^*$ in (2.4) and (2.5). Let h_n^{j*}, σ_N^{j*}, ∇h_n^{j*} and so on denote $h_n^j(x_n^*)$, $\sigma_N^j(s_{N-1}^*, x_N^*)$, $\nabla h_n^j(x_n^*)$ and so on, and put for n=1,...,N,

$$I_n^* = \{j;\ h_n^{j*}=0,\ j=1,\ldots,m_n\}$$

and

$$I^* = \{j;\ \sigma_N^{j*}=0,\ j=1,\ldots,m\}.$$

Suppose that

(C_4) For n=1,...,N-1, gradient vectors $\{\nabla h_n^{j*};\ j\in I_n^*\}$ are linearly independent, and $\{\nabla h_N^{j*};\ j\in I_N^*\}$ and $\{\nabla_x\sigma_N^{j*};\ j\in I^*\}$ are also linearly independent.

This condition implies that the second-order constraint qualification is satisfied for each subproblem, if F_{n+1} is twice continously differentiable (differentiability of F_{n+1} will be proved in Lemma 1). Consequently, it follows from the second-order necessary conditions [4, p.25] for x_n^* to be an optimal solution of (P_n) with $s_{n-1}=s_{n-1}^*$ that: For each n=1,...,N-1, there exists a Lagrange multiplier λ_n^* such that

(3.3) $$\nabla_x L_n(x_n^*,\lambda^*,s_{n-1}^*) = \nabla_x\xi_n^* + \frac{\partial}{\partial y}\xi_n^*\nabla F_{n+1}^*\nabla_x\sigma_n^* + (\lambda_n^*)^T\nabla h_n^* = 0,$$

(3.4) $$\mathrm{Diag}(\lambda_n^*)h_n^* = 0,$$

(3.5) $$h_n(x_n^*) \le 0,$$

(3.6) $$\lambda_n^* \ge 0,$$

and such that for every vector z satisfying $\nabla h_n^{j*}z=0$ for all $j\in I_n^*$,

(3.7) $$z^T\nabla_x^2 L_n^* z \ge 0,$$

where $\mathrm{Diag}(\lambda_n)$ denotes the diagonal matrix with the j-th diagonal element λ_n^j and

(3.8) $$\nabla_x^2 L_n = \nabla_x^2 \xi_n + 2\left(\frac{\partial}{\partial y}\nabla_x \xi_n\right)^T \nabla F_{n+1}\nabla_x \sigma_n + \frac{\partial}{\partial y}\xi_n\{\nabla F_{n+1}\nabla_x^2\sigma_n + \nabla_x\sigma_n^T\nabla^2 F_{n+1}\nabla_x\sigma_n\}$$
$$+\nabla_x\sigma_n^T\nabla F_{n+1}^T\frac{\partial^2}{\partial y^2}\xi_n\nabla F_{n+1}\nabla_x\sigma_n + \lambda_n^T\nabla^2 h_n;$$

For n=N, there exist Lagrange multipliers λ_N^* and μ^* such that

(3.9) $$\nabla_x L_N(x_N^*, \lambda_N^*, \mu^*, s_{N-1}^*) = \nabla\xi_N^* + (\lambda_N^*)^T\nabla h_N^* + (\mu^*)^T\nabla_x\sigma_N^* = 0,$$

(3.10) $$\mathrm{Diag}\,(\lambda_N^*)h_N^* = 0,\ \mathrm{Diag}\,(\mu^*)\sigma_N^* = 0,$$

(3.11) $$h_N(x_N^*) \le 0, \qquad \sigma_N(s_{N-1}^*, x_N^*) \le 0$$

(3.12) $$\lambda_N^* \ge 0, \qquad \mu^* \ge 0$$

and such that for every vector z satisfying $\nabla h_N^{j*}z=0$ for all $j\in I_N^*$ and $\nabla_x\sigma_N^{j*}z=0$ for all $j\in I^*$,

(3.13) $$z^T\nabla_x^2 L_N^* z \ge 0,$$

where

(3.14) $$\nabla_x^2 L_N = \nabla^2\xi_N + \lambda_N^T\nabla^2 h_N + \mu^T\nabla_x^2\sigma_N.$$

Put for n=1,...,N-1, $X_n=(x_n^T,\lambda_n^T)^T$ and $X_N=(x_N^T,\lambda_N^T,\mu^T)^T$ and define for n=1,..., N-1,

(3.15) $$T_n(X_n,s_{n-1}) = (\nabla_x L_n(x_n,s_{n-1}), h_n(x_n)^T\mathrm{Diag}(\lambda_n))^T$$

and

(3.16) $$T_N(X_N,s_{N-1}) = (\nabla_x L_N(X_N,s_{N-1}), h_N(x_N)^T\mathrm{Diag}(\lambda_N), \sigma_N(s_{N-1},x_N)^T\mathrm{Diag}(\mu))^T.$$

It should be noted that for arbitrarily fixed s_{n-1}, $T_n(X_n,s_{n-1})=0$ is a system of (k_n+m_n) equations for the same number of unknowns and that $T_N(X_N,s_{N-1})=0$ with fixed s_{N-1} is a system of (k_N+m_N+m) equations for the same number of unknowns. Therefore if X_n^* (n=1,...,N) is an isolated solution of $T_n(X_n,s_{n-1}^*)=0$, that is, if there exists a neighbourhood of X_n^* which contains no other solutions of $T_n(X_n,s_{n-1}^*)=0$, then X_n^* satisfying the second-order necessary conditions can be obtained by solving $T_n(X_n,s_{n-1}^*)=0$ in the neighbourhood without taking into account inequalities (3.5) and (3.6) or (3.11) and (3.12). From inverse function theorem [23, p.125] it follows that if the Jacobian matrix of T_n with respect to X_n is nonsingular at X_n^*, then X_n^* is an isolated solution of $T_n(X_n,s_{n-1}^*)=0$. The Jacobian matrix of T_n, denoted by J_n, is: for n=1,...,N-1,

(3.17) $$J_n(X_n,s_{n-1}) = \begin{pmatrix} \nabla_x^2 L_n & \nabla h_n^T \\ \mathrm{Diag}(\lambda_n)\nabla h_n & \mathrm{Diag}(h_n) \end{pmatrix}$$

and

(3.18) $$J_N(X_N,s_{N-1}) = \begin{pmatrix} \nabla_x^2 L_N & \nabla h_N^T & \nabla_x\sigma_N^T \\ \mathrm{Diag}(\lambda_N)\nabla h_N & \mathrm{Diag}(h_N) & 0 \\ \mathrm{Diag}(\mu)\nabla_x\sigma_N & 0 & \mathrm{Diag}(\sigma_N) \end{pmatrix}.$$

It is clear that if $\lambda_n^{j*}=0$ for some n and $j\in I_n^*$ or $\mu^{j*}=0$ for some $j\in I^*$, then some J_n^* are singular. Consequently it is assumed that

(C_5) For $n=1,\dots,N$, $\lambda_n^{j*}>0$ for all $j\in I_n^*$ and $\mu^{j*}>0$ for all $j\in I^*$.

Denote by H_n^* ($n=1,\dots,N-1$) the matrix whose rows are ∇h_n^{j*} ($j\in I_n^*$) and by H_N^* the matrix whose rows are ∇h_N^{j*} ($j\in I_N^*$) and $\nabla_x\sigma_N^{j*}$ ($j\in I^*$). Moreover denote the kernel of H by N(H), that is, $N(H)=\{z;\ Hz=0\}$. The last assumption is [8]:

(C_6) $N(H_n^*)\cap N(\nabla_x^2 L_n^*)=\{0\}$ for $n=1,\dots,N$.

Now denote by

$$X_n^*(s_{n-1}) = (x_n^*(s_{n-1})^T,\ \lambda_n^*(s_{n-1})^T)^T \quad \text{for } n<N,$$
$$= (x_N^*(s_{N-1})^T,\ \lambda_N^*(s_{N-1})^T,\ \mu^*(s_{N-1})^T)^T \quad \text{for } n=N$$

a solution of $T_n(X_n,s_{n-1})=0$ for fixed s_{n-1} and by $K_n(X_n,s_{n-1})$ the Jacobian matrix of T_n with respect to s_{n-1}. The Jacobian matrix K_n is given by

$$(3.19)\qquad K_n(X_n,s_{n-1}) = \begin{pmatrix} \nabla_{xs}^2 L_n(X_n,s_{n-1}) \\ A_n(X_n,s_{n-1}) \end{pmatrix},$$

where for $n=1,\dots,N-1$,

$$(3.20)\qquad \nabla_{xs}^2 L_n = \nabla_x\sigma_n^T\{\frac{\partial}{\partial y}\xi_n\nabla^2 F_{n+1}+\nabla F_{n+1}^T\frac{\partial^2}{\partial y^2}\xi_n\nabla F_{n+1}\}\nabla_s\sigma_n$$
$$+\frac{\partial}{\partial y}\xi_n\nabla F_{n+1}\nabla_{xs}^2\sigma_n+(\frac{\partial}{\partial y}\nabla\xi_n)^T\nabla F_{n+1}\nabla_s\sigma_n,$$

$$(3.21)\qquad A_n = 0,$$

and

$$(3.22)\qquad \nabla_{xs}^2 L_N = \mu^T\sigma_{xs\ N}^2,$$

$$(3.23)\qquad A_N = \begin{pmatrix} 0 \\ \mathrm{Diag}(\mu)\nabla_s\sigma_N \end{pmatrix}.$$

Then the following lemma holds.

Lemma 1. Suppose that Conditions (C_1) through (C_6) are satisfied. Then for $n=1,\dots,N$, $J_n(X_n^*,s_{n-1}^*)$ is nonsingular. Moreover, $F_n(s_{n-1})$ is twice continuously differentiable in a neighbourhood O_s^{n-1} of s_{n-1}^*, and for $n=1,\dots,N-1$,

$$(3.24)\qquad F_n(s_{n-1}) = \xi_n(x_n^*(s_{n-1}),F_{n+1}(\sigma_n(s_{n-1},x_n^*(s_{n-1})))),$$

$$(3.25)\qquad \nabla F_n(s_{n-1}) = \frac{\partial}{\partial y}\xi_n(x_n^*(s_{n-1}),F_{n+1}(\sigma_n(s_{n-1},x_n^*(s_{n-1}))))\nabla F_{n+1}(\sigma_n(s_{n-1},$$
$$x_n^*(s_{n-1})))\nabla_s\sigma_n(s_{n-1},x_n^*(s_{n-1})),$$

$$(3.26)\qquad \nabla^2 F_n(s_{n-1}) = \nabla x_n^*(s_{n-1})^T\{(\frac{\partial}{\partial y}\nabla\xi_n)^T\nabla F_{n+1}+\nabla_x\sigma_n^T\nabla F_{n+1}^T\frac{\partial^2}{\partial y^2}\xi_n\nabla F_{n+1}$$
$$+\frac{\partial}{\partial y}\xi_n\nabla_x\sigma_n^T\nabla^2 F_{n+1}\}\nabla_s\sigma_n+\nabla_s\sigma_n^T\{\nabla F_{n+1}^T\frac{\partial^2}{\partial y^2}\xi_n\nabla F_{n+1}$$

$$+ \frac{\partial}{\partial y} \xi_n \nabla^2 F_{n+1} \} \nabla_s \sigma_n + \frac{\partial}{\partial y} \xi_n \nabla F_{n+1} \{ \nabla_s^2 \sigma_n + \nabla_{sx}^2 \sigma_n \nabla x_n^*(s_{n-1}) \}$$

and

$$F_N(s_{N-1}) = \xi_N(x_N^*(s_{N-1})), \tag{3.27}$$

$$\nabla F_N(s_{N-1}) = \mu^*(s_{N-1})^T \nabla_s \sigma_N(s_{N-1}, x_N^*(s_{N-1})), \tag{3.28}$$

$$\nabla^2 F_N(s_{N-1}) = \nabla \mu^*(s_{N-1})^T \nabla_s \sigma_N + \mu^*(s_{N-1})^T \nabla_s^2 \sigma_N + \mu^*(s_{N-1})^T \nabla_{sx}^2 \sigma_N \nabla x_N^*(s_{N-1}), \tag{3.29}$$

where for $n=1,\ldots,N$, $X_n^*(s_{n-1})$ belongs to a neighbourhood O_X^n of X_n^* and

$$\nabla X_n^*(s_{n-1}) = -J_n^{-1}(X_n^*(s_{n-1}), s_{n-1}) K_n(X_n^*(s_{n-1}), s_{n-1}). \tag{3.30}$$

Proof: Since under Condition (C_6), (3.13) becomes

$$z^T \nabla_x^2 L_N^* z > 0 \tag{3.31}$$

for every nonzero vector $z \in N(H_N^*)$, the nonsingularity of J_N^* can be proved in a way similar to [4, p.80-81]. Consequently, implicit function theorem [23, p. 128] implies that there exist open neighbourhoods O_X of X_N^* and O_s of s_{N-1}^* such that for any $s_{N-1} \in c\ell O_s$, $T_N(X_N, s_{N-1})=0$ has a unique solution $X_N^*(s_{N-1}) \in c\ell O_X$ and for any $s_{N-1} \in O_s$, (3.30) holds for $n=N$, where $c\ell O_s$ means the closure of O_s. Since (3.9) through (3.12) and (3.31) are the second-order sufficient conditions [4, p.30] for $x_N^* = x_N^*(s_{N-1}^*)$ to be an isolated local optimal solution of $F_N(s_{N-1}^*)$, there exist open neighbourhoods $O_X^N \subset O_X$ and $O_s^{N-1} \subset O_s$ such that $x_N^*(s_{N-1}) \in O_X^N$ is an isolated optimal solution of $F_N(s_{N-1})$ for $s_{N-1} \in O_s^{N-1}$. It is clear that $F_N(s_{N-1}) = \xi_N(x_N^*(s_{N-1}))$ and $T_N(X_N^*(s_{N-1}), s_{N-1})=0$ for $s_{N-1} \in c\ell O_s^{N-1}$. Consequently,

$$F_N(s_{N-1}) = L_N(X_N^*(s_{N-1}), s_{N-1})$$

and hence $\nabla F_N(s_{N-1}) = \nabla_X L_N(X_N^*(s_{N-1}), s_{N-1}) \nabla X_N^*(s_{N-1}) + \nabla_s L_N(X_N^*(s_{N-1}), s_{N-1})$

$$= \mu^*(s_{N-1})^T \nabla_s \sigma_N(s_{N-1}, x_N^*(s_{N-1})).$$

By (3.30) with $n=N$, this implies that F_N is twice continuously differentiable and $\nabla^2 F_N$ is given by (3.29). Therefore $\nabla_x L_{N-1}$ and $\nabla_x^2 L_{N-1}$ are well-defined, and the lemma for $n=1,\ldots,N-1$ can be proved in the same way as in the above.

4. Differential Dynamic Programming

Denote any iteration procedure for solving the system of the nonlinear equations $T_n(X_n, s_{n-1})=0$ for fixed s_{n-1} by

$$X_n^{k+1} = U_n(X_n^k, s_{n-1}), \tag{4.1}$$

where $k=0,1,\ldots$. Since by Lemma 1, $J_n^{-1}(X_n, s_{n-1})$ exists for $X_n \in O_X^n$ and $s_{n-1} \in O_s^{n-1}$, for example, Newton's method is described as

(4.2) $$U_n(X_n^k, s_{n-1}) = X_n^k - J_n^{-1}(X_n^k, s_{n-1}) T_n(X_n^k, s_{n-1}).$$

Let an initial guess $\{X_n^0 \in O_X^n;\ n=1,\dots,N\}$ be given. Then the initial trajectory $\{s_n^0;\ n=0,\dots,N\}$ corresponding to $\{x_n^0\}$ is determined by (2.2) with $s_0^0=0$. As noted in the preceding section, if $s_{n-1}^0 \in O_s^{n-1}$, then an optimal solution of (P_n) with $s_{n-1}=s_{n-1}^0$ can be obtained by solving $T_n(X_n, s_{n-1}^0)=0$ in O_X^n. Since the iteration procedure U_n usually generates a sequence $\{X_n^*;\ k=1,2,\dots\}$ converging to the solution of $T_n(X_n, s_{n-1}^0)=0$ which is nearest to the initial point X_n^0, $X_n^1 = U_n(X_n^0, s_{n-1}^0)$ will come nearer to the optimal solution of $F_n(s_{n-1}^0)$. In particular, since s_0^0 is always fixed to the origin, X_1^1 and s_1^1 given by

$$X_1^1 = U_1(X_1^0, s_0^0) \quad \text{and} \quad s_1^1 = \sigma_1(s_0^0, x_1^1)$$

will come nearer to X_1^* and s_1^*. This suggests the following conceptual algorithm: Compute X_n^{k+1} by $X_n^{k+1} = U_n(X_n^k, s_{n-1}^k)$ for $n=N,\dots,1$ and determine s_n^{k+1} by (2.2) with $s_0^k=0$ for $n=1,\dots,N-1$. However, it should be noted that T_n, J_n and K_n $(n=1,\dots,N-2)$ which may be used in U_n contain unknown values $F_{n+1}(s_n^k)$, $\nabla F_{n+1}(s_n^k)$ and $\nabla^2 F_{n+1}(s_n^k)$. Therefore it is essential to obtain their approximate values which guarantee that $\{X_n^*\}$ is *a point of attraction* of the following D.D.P. algorithm, that is, there exist open neighbourhoods $O_n \subset O_X^n$ $(n=1,\dots,N)$ such that for any $X_n^0 \in O_n$, X_n^k $(k=1,2,\dots)$ generated by the algorithm remain in O_n and converge to X_n^* [23, p.299]. Since exact values $F_{n+1}(s_n^k)$, $\nabla F_{n+1}(s_n^k)$ and $\nabla^2 F_{n+1}(s_n^k)$ are given in Lemma 1, such approximate values can be obtained by approximating suitably (3.24) through (3.29). Denote by $\tilde{F}_{n+1}^k$, $\nabla\tilde{F}_{n+1}^k$ and $\nabla^2\tilde{F}_{n+1}^k$ the approximate values of $F_{n+1}(s_n^k)$, $\nabla F_{n+1}(s_n^k)$ and $\nabla^2 F_{n+1}(s_n^k)$, respectively, and denote by $\tilde{T}_n$, $\tilde{J}_n$, $\tilde{K}_n$ and $\tilde{U}_n$ $(n=1,\dots,N-1)$ T_n, J_n, K_n and U_n with F_{n+1}, ∇F_{n+1} and $\nabla^2 F_{n+1}$ substituted by their approximate values $\tilde{F}_{n+1}^k$, $\nabla\tilde{F}_{n+1}^k$ and $\nabla^2\tilde{F}_{n+1}^k$, respectively. For example, when for $n=1,\dots,N-1$, U_n represents Newton's method (4.2), $\tilde{U}_n$ is described as

(4.3) $$\tilde{U}_n(X_n^k, s_{n-1}) = X_n^k - \tilde{J}_n^{-1}(X_n^k, s_{n-1}) \tilde{T}_n(X_n^k, s_{n-1}).$$

Note that as shown by (3.9), (3.14), (3.16), (3.18), (3.19), (3.22) and (3.23), T_N, J_N and K_N include no unknown functions so that U_N also does.

D.D.P. algorithm: Let $\{X_n^0;\ n=1,\dots,N\}$ and $\{s_n^0;\ n=0,\dots,N-1\}$ be given. Set $k=0$.

Step 1: Compute $\tilde{X}_N^{k+1}$, $\tilde{F}_N^k$, $\nabla\tilde{F}_N^k$ and $\nabla^2\tilde{F}_N^k$ by

(4.4) $$\tilde{X}_N^{k+1} = U_N(X_N^k, s_{N-1}^k),$$

(4.5) $$\tilde{F}_N^k = \xi_N(\tilde{x}_N^{k+1}),$$

(4.6) $$\nabla\tilde{F}_N^k = (\tilde{\mu}^{k+1})^T \nabla_s \sigma_N(s_{N-1}^k, \tilde{x}_N^{k+1})$$

and

(4.7) $$\nabla^2\tilde{F}_N^k = -[J_N^{-1}(X_N^k,s_{N-1}^k)K_N(X_N^k,s_{N-1}^k)]_\mu \nabla_s\sigma_N(s_{N-1}^k,x_N^k)+(\mu^k)^T\{\nabla_s^2\sigma_N(s_{N-1}^k,x_N^k) -\nabla_{sx}^2\sigma_N(s_{N-1}^k,x_N^k)[J_N^{-1}K_N]_x\},$$

where $[J_N^{-1}K_N]_\mu$ and $[J_N^{-1}K_N]_x$ denote submatrices of $J_N^{-1}K_N$ corresponding to μ and x_N, respectively.

Step 2: For n=N-1,...,2, compute $\tilde{X}_n^{k+1}$, $\tilde{F}_n^k$, $\nabla\tilde{F}_n^k$ and $\nabla^2\tilde{F}_n^k$ by

(4.8) $$\tilde{X}_n^{k+1} = \tilde{U}_n(X_n^k,s_{n-1}^k),$$

(4.9) $$\tilde{F}_n^k = \xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k)+\frac{\partial}{\partial y}\xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k)\nabla\tilde{F}_{n+1}^k\nabla_x\sigma_n(s_{n-1}^k,x_n^k)(\tilde{x}_n^{k+1}-x_n^k),$$

(4.10) $$\nabla\tilde{F}_n^k = \frac{\partial}{\partial y}\xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k)\nabla\tilde{F}_{n+1}^k\nabla_s\sigma_n(s_{n-1}^k,\tilde{x}_n^{k+1})+(\tilde{x}_n^{k+1}-x_n^k)^T\nabla_x\sigma_n(s_{n-1}^k,x_n^k)^T \cdot(\nabla\tilde{F}_{n+1}^k)^T\frac{\partial^2}{\partial y^2}\xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k)\nabla\tilde{F}_{n+1}^k\nabla_s\sigma_n(s_{n-1}^k,\tilde{x}_n^{k+1}) + \frac{\partial}{\partial y}\xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k)(\tilde{x}_n^{k+1}-x_n^k)^T\nabla_x\sigma_n(s_{n-1}^k,x_n^k)^T\nabla^2\tilde{F}_{n+1}^k\nabla_s\sigma_n(s_{n-1}^k,\tilde{x}_n^{k+1})$$

and

(4.11) $$\nabla^2\tilde{F}_n^k = -[\tilde{J}_n^{-1}\tilde{K}_n(X_n^k,s_{n-1}^k)]_x^T\{(\frac{\partial}{\partial y}\nabla\xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k))^T\nabla\tilde{F}_{n+1}^k+\nabla_x\sigma_n(s_{n-1}^k,x_n^k)^T \cdot(\nabla\tilde{F}_{n+1}^k)^T\frac{\partial^2}{\partial y^2}\xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k)\nabla\tilde{F}_{n+1}^k+\frac{\partial}{\partial y}\xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k)\nabla_x\sigma_n(s_{n-1}^k,x_n^k)^T \cdot\nabla^2\tilde{F}_{n+1}^k\}\nabla_s\sigma_n(s_{n-1}^k,\tilde{x}_n^{k+1})+\nabla_s\sigma_n(s_{n-1}^k,\tilde{x}_n^{k+1})^T\{(\nabla\tilde{F}_{n+1}^k)^T\frac{\partial^2}{\partial y^2}\xi_n(\tilde{x}_n^{k+1}, \tilde{F}_{n+1}^k)\nabla\tilde{F}_{n+1}^k+\frac{\partial}{\partial y}\xi_n(\tilde{x}_n^{k+1},\tilde{F}_{n+1}^k)\nabla^2\tilde{F}_{n+1}^k\}\nabla_s\sigma_n(s_{n-1}^k,\tilde{x}_n^{k+1})+\frac{\partial}{\partial y}\xi_n(\tilde{x}_n^{k+1}, \tilde{F}_{n+1}^k)\nabla\tilde{F}_{n+1}^k\{\nabla_s^2\sigma_n(s_{n-1}^k,x_n^k)-\nabla_{sx}^2\sigma_n(s_{n-1}^k,x_n^k)[\tilde{J}_n^{-1}\tilde{K}_n(X_n^k,s_{n-1}^k)]_x\}.$$

Step 3: Compute X_1^{k+1} by

(4.12) $$X_1^{k+1} = \tilde{U}_1(X_1^k,s_0^k),\text{ where } s_0^k=0.$$

Step 4: For n=2,...,N-1, compute s_{n-1}^{k+1} and X_n^{k+1} by

(4.13) $$s_{n-1}^{k+1} = \sigma_{n-1}(s_{n-2}^{k+1},x_{n-1}^{k+1})$$

and

(4.14) $$X_n^{k+1} = \tilde{X}_n^{k+1}-[\tilde{J}_n^{-1}\tilde{K}_n(X_n^k,s_{n-1}^k)](s_{n-1}^{k+1}-s_{n-1}^k).$$

Step 5: Compute s_{N-1}^{k+1} and X_N^{k+1} by

(4.15) $$s_{N-1}^{k+1} = \sigma_{N-1}(s_{N-2}^{k+1},x_{N-1}^{k+1})$$

and

(4.16) $$X_N^{k+1} = \tilde{X}_N^{k+1}-[J_N^{-1}K_N(X_N^k,s_{N-1}^k)](s_{N-1}^{k+1}-s_{N-1}^k).$$

Set k=k+1 and go back to Step 1.

Remark 2. Note that $\tilde{X}_n^{k+1}$ in Steps 1 and 2 is not an improved estimate to a solution of $T_n(X_n,s_{n-1}^{k+1})=0$ but that to a solution of $T_n(X_n,s_{n-1}^k)=0$. Steps 4 and 5 compute the new state s_{n-1}^{k+1} and adjust $\tilde{X}_n^{k+1}$ for the old state s_{n-1}^{k} to X_n^{k+1} for s_{n-1}^{k+1}. Let $c_n(x_n)$ and $c_n'(x_n)$ be twice differentiable functions with uniformly continuous derivatives. When all ξ_n (n=1,...,N-1) are given by $\xi_n(x_n,y)=c_n(x_n)+y$, computations of $\tilde{F}_n^k$ in Steps 1 and 2 are unnecessary, because none of $\tilde{T}_n$, $\tilde{J}_n$, $\tilde{K}_n$, $\nabla\tilde{F}_{n-1}^k$ and $\nabla^2\tilde{F}_{n-1}^k$ include values of $\tilde{F}_n^k$. A discrete time optimal control problem is one of the most important problems with such ξ_n. Moreover, a separable program is composed of such ξ_n and σ_n given by $\sigma_n(s_{n-1},x_n)=s_{n-1}+c_n'(x_n)$. Therefore the D.D.P. algorithm for solving separable programs becomes much simpler than the present D.D.P. algorithm [22].

In the following, $\|\cdot\|$ denotes ℓ_1 norm or the corresponding matrix norm, i.e., for matrix $A=(a_j^i)$, $\|A\|=\max_j \sum_i |a_j^i|$. Put $\delta_n^k=\|X_n^k-X_n^*(s_{n-1}^k)\|$ and $\delta^k=(\delta_1^k,\ldots,\delta_N^k)^T$.

Theorem 2: Suppose that Conditions (C_1) through (C_6) are satisfied. It is assumed that the iteration procedures U_n (n=1,...,N) satisfy the following conditions:

(C_7) For n=1,...,N, there exist nonnegative numbers a_n and p such that for $s_{n-1}^k\in O_s^{n-1}$ and $X_n^k\in O_X^n$, $\|U_n(X_n^k,s_{n-1}^k)-X_n^*(s_{n-1}^k)\|\le a_n(\delta_n^k)^{1+p}$, where if p=0, then $a_n<1$;

(C_8) For n=1,...,N-1, there exist positive numbers b_{n1}, b_{n2} and b_{n3} such that for $s_{n-1}^k\in O_s^{n-1}$ and $X_n^k\in O_X^n$,

$$\|U_n(X_n^k,s_{n-1}^k)-\tilde{U}_n(X_n^k,s_{n-1}^k)\| \le b_{n1}\|F_{n+1}(s_n^k)-\tilde{F}_{n+1}^k\| +b_{n2}\|\nabla F_{n+1}(s_n^k)-\nabla\tilde{F}_{n+1}^k\| +b_{n3}\delta_n^k\|\nabla^2F_{n+1}(s_n^k)-\nabla^2\tilde{F}_{n+1}^k\| .$$

Then the optimal solution $\{X_n^*\}$ of (P) is a point of attraction of the D.D.P. algorithm. Moreover its convergence is *R-superlinear* or *R-linear* [23, p.291], according as the constant p in (C_7) is positive or zero.

The proof is given in Section 8. In [22] a similar result is shown for the D.D.P. algorithm for solving separable programs. In [22], however, its convergence rate is not shown explicitly and the constant corresponding to p in (C_7) is assumed positive. Since p in (C_7) may be zero, almost all iteration methods for solving a system of nonlinear equations satisfy Condition (C_7). In fact, Newton's method, discrete Newton's method, some modifications of Newton's method, secant method [23], quasi-Newton methods [2] and Newton-Moser type method [6] satisfy Condition (C_7) with p>0. In addition, parallel-chord method, simplified Newton method and successive overrelaxation method [23] satisfy (C_7) with p=0. Consequently, all these methods can be used as U_n in the D.D.P. algorithm, if they satisfy Condition (C_8).

5. Combination with Newton's method

One of the methods which are used popularly in solving a system of non-linear equations is Newton's method given by (4.2). This section deals with the D.D.P. algorithm with U_n and $\tilde{U}_n$ given by (4.2) and (4.3). The following lemma is necessary in proving that Newton's method satisfies Conditions (C_7) and (C_8) in Theorem 2.

Lemma 2. Suppose that Conditions (C_1) through (C_6) are satisfied. Then for n=1,...,N-1 and integer k, there exist nonnegative numbers α^n_{ij} (i,j=1,2,3) such that for $X^k_n \epsilon O^n_X$ and $s^k_{n-1} \epsilon O^{n-1}_s$

$$\| T_n(X^k_n, s^k_{n-1}) - \tilde{T}_n(X^k_n, s^k_{n-1}) \| \le \alpha^n_{11} \| F_{n+1}(s^k_n) - \tilde{F}^k_{n+1} \| + \alpha^n_{12} \| \nabla F_{n+1}(s^k_n) - \nabla \tilde{F}^k_{n+1} \|$$

$$\| J_n(X^k_n, s^k_{n-1}) - \tilde{J}_n(X^k_n, s^k_{n-1}) \| \le \alpha^n_{21} \| F_{n+1}(s^k_n) - \tilde{F}^k_{n+1} \| + \alpha^n_{22} \| \nabla F_{n+1}(s^k_n) - \nabla \tilde{F}^k_{n+1} \| + \alpha^n_{23} \| \nabla^2 F_{n+1}(s^k_n) - \nabla^2 \tilde{F}^k_{n+1} \| ,$$

and
$$\| K_n(X^k_n, s^k_{n-1}) - \tilde{K}_n(X^k_n, s^k_{n-1}) \| \le \alpha^n_{31} \| F_{n+1}(s^k_n) - \tilde{F}^k_{n+1} \| + \alpha^n_{32} \| \nabla F_{n+1}(s^k_n) - \nabla \tilde{F}^k_{n+1} \| + \alpha^n_{33} \| \nabla^2 F_{n+1}(s^k_n) - \nabla^2 \tilde{F}^k_{n+1} \| .$$

Proof: From (3.15), (3.17), and (3.19) it follows that

$$\| T_n - \tilde{T}_n \| = \| \nabla_x L_n - \nabla_x \tilde{L}_n \| , \qquad \| J_n - \tilde{J}_n \| = \| \nabla^2_x L_n - \nabla^2_x \tilde{L}_n \|$$

and
$$\| K_n - \tilde{K}_n \| = \| \nabla^2_{xs} L_n - \nabla^2_{xs} \tilde{L}_n \| .$$

Therefore Condition (C_3) and (3.3), (3.8), (3.20) imply that the lemma holds.

Corollary 1. Suppose that Conditions (C_1) through (C_6) are satisfied. Then the optimal solution $\{X^*_n\}$ of (P) is a point of attraction of the D.D.P. algorithm with Newton's method. The convergence of this algorithm is *R-quadratic* [23, p.291].

Proof: Since Condition (C_3) implies that J_n is Lipschitz-continuous in X_n, Newton's method satisfies Condition (C_7) with p=1 [23, p.312]. From (4.2) and (4.3) it follows that

$$\| U_n(X^k_n, s^k_{n-1}) - \tilde{U}_n(X^k_n, s^k_{n-1}) \| \le \| \tilde{J}^{-1}_n \| (\| T_n - \tilde{T}_n \| + \| T_n \| \, \| J^{-1}_n \| \, \| J_n - \tilde{J}_n \|).$$

Therefore Lemma 2 implies that Newton's method satisfies Condition (C_8), because

$$\| T_n(X^k_n, s^k_{n-1}) \| = \| T_n(X^k_n, s^k_{n-1}) - T_n(X^*_n(s^k_{n-1}), s^k_{n-1}) \|$$

and the boundedness of $\| \tilde{J}^{-1}_n \|$ is shown in Lemma 3 in Section 8. The R-quadratic convergence can be proved in much the same way as in Section 8.

The same argument as in the above proof can apply to the D.D.P. algorithm with any other iteration method U_n represented by T_n, J_n and K_n. Thus the optimal solution $\{X_n^*\}$ of (P) will be a point of attraction of the D.D.P. algorithm with any iteration method noted in the preceding section.

Let us discuss the operation count, i.e., the number of multiplications and divisions, per one iteration of the D.D.P. algorithm with Newton's method. Denote for n=1,...,N-1, $\tilde J_n^{-1}\tilde T_n$ and $\tilde J_n^{-1}\tilde K_n$ by ΔX_n and ∇X_n, respectively, and $J_N^{-1}T_N$ and $J_N^{-1}K_N$ by ΔX_N and ∇X_N, respectively. Put for n=1,...,N-1, $M_n=k_n+m_n$ and N_n $m+m_n$, and $M_N=k_N+m+m_N$ and $N_N=m+m_N$. Then ΔX_n and ∇X_n can be obtained by solving the following matrix equation:

$$\tilde J_n(\Delta X_n \nabla X_n) = (\tilde T_n \tilde K_n). \tag{5.1}$$

Since by (3.15) through (3.19), $\tilde J_n$, $\tilde T_n$ and $\tilde K_n$ are $M_n\times M_n$, $M_n\times 1$ and $M_n\times m$ matrices, the operation count for solving this matrix equation by using Gaussian elimination is $[M_n^3+3(m+1)M_n^2-M_n]/3$. In addition, the operation counts for constructing $\tilde J_n$, $\tilde T_n$ and $\tilde K_n$ (n=1,...,N-1) are $k_n[2k_nN_n+(m+1)^2+3k_n+1]$, $(k_n+1)N_n$, $m[(k_n+1)(2m+k_n)+m+2]$, respectively, and those for constructing J_N, T_N and K_N are $k_N(k_N+1)N_N$, $(k_N+1)N_N$ and $m^2(k_N+1)$, respectively. Since the operation counts for constructing $\tilde F_N^k$, $\nabla\tilde F_N^k$ and $\nabla^2\tilde F_N^k$ by (4.5) through (4.7) are 0, m^2 and $m^2(m+2k_N)$, respectively, the operation count in Step 1 is $[M_N^3+3(m+1)M_N^2-M_N]/3+(k_N+1)^2N_N+m^2(m+3k_N+2)$. Similarly, since the operation counts for constructing $\tilde F_n^k$, $\nabla\tilde F_n^k$ and $\nabla^2\tilde F_n^k$ by (4.9) through (4.11) are $m(k_n+1)+k_n$, $m(4m+2k_n+3)+k_n+1$ and $2m[2m^2+(m+1)(2k_n+1)]+k_n$, respectively, the operation count in Step 2 is $\sum_{n=2}^{N-1}\{[M_n^3+3(m+1)M_n^2-M_n]/3+(2k_n^2+k_n+1)N_n+m[2(m+1)(2m+2k_n+1)+3m(k_n+1)+k_n^2+6k_n+6]+3k_n^2+5k_n+1\}$. The operation count in Step 3 is $(M_1^3+3M_1^2-M_1)/3+(2k_1^2+k_1+1)N_1+k_1(m+1)^2+k_1(3k_1+1)$ and that in Steps 4 and 5 is $m\sum_{n=2}^{N} M_n$. Therefore the operation count per one iteration of the D.D.P. algorithm with Newton's method is:

$$\sum_{n=1}^{N}\{[M_n^3+3(m+1)M_n^2+(3m-1)M_n]/3+(2k_n^2+k_n+1)N_n\} + (m+3)\sum_{n=2}^{N-1}k_n^2$$
$$+ (7m^2+10m+5)\sum_{n=2}^{N-1}k_n + N(4m^3+9m^2+8m) - (k_N^2-k_N)N_N - mM_1(M_1+1)$$
$$- m(7m^2+16m+16) + 3m^2k_N + (m^2+2m+1)k_1 + 3k_1^2 + k_1.$$

Neglecting the order of $\sum_{n=1}^{N}(m+M_n)M_n$ and $\sum_{n=1}^{N}k_nN_n$, the operation count is of the order of

$$\sum_{n=1}^{N}\{M_n^3/3+mM_n^2+2k_n^2N_n+mk_n(7m+k_n)\}. \tag{5.2}$$

The D.D.P. algorithm with Newton's method requires the core memory which stores values of $\tilde X_n^{k+1}$, $\nabla X_n=[\tilde J_n^{-1}\tilde K_n]$, s_n^k and s_n^{k+1} for all n in Steps 4 and 5 and those of X_n^k, $\tilde T_n$, $\tilde J_n$, $\tilde K_n$, $\tilde F_n^k$, $\tilde F_{n+1}^k$, $\nabla\tilde F_n^k$, $\nabla\tilde F_{n+1}^k$, $\nabla^2\tilde F_n^k$ and $\nabla^2\tilde F_{n+1}^k$ for each n in Steps 1 and 2. Therefore the magnitude of the core memory required for the D.D.P. algorithm with Newton's method is:

$$\sum_{n=1}^{N}(mM_n+k_n) + 2\{m^2+(N+1)m+1\} + \max_n\{M_n^2+(m+2)M_n\}. \tag{5.3}$$

Summing up the results obtained above yields the following corollary.

Corollary 2. The operation count per one iteration of the D.D.P. algorithm with Newton's method is of the order of (5.2) and the magnitude of the core memory required for the algorithm is given by (5.3).

This corollary shows that both the operation count and the magnitude of the required core memory for the D.D.P. algorithm with Newton's method grow only linearly with N. This desirable property is one of the well-known desirable properties of dynamic programming. Since the D.D.P. algorithm is based upon the decomposition by dynamic programming, it inherits almost all desirable properties of dynamic programming.

When M_n is large for some n, Newton's method (4.2) is not a good practical method for solving the system of nonlinear equations. This is because solving (5.1) consumes much time. For such n, Newton's method had better be replaced by quasi-Newton method [2]. Since $M_N=k_N+m+m_N$ is usually larger than other $M_n=k_n+m_n$, let N be such an n. In quasi-Newton method approximate matrices of $(-J_N^{-1})$ are successively computed without calculating J_N. Denote by G_N^k an approximate matrix of $(-J_N^{-1}(X_N^k,s_{N-1}^k))$ and put $G_N^0=-J_N^{-1}(X_N^0,s_{N-1}^0)$. Then for k=0,1, ..., Step 1 is modified as follows:

Step 1-1: Compute M_N dimensional vectors y_N^k and $\tilde{X}_N^{k+1}$ by

$$y_N^k = G_N^kT_N(X_N^k,s_{N-1}^k)$$

and

$$\tilde{X}_N^{k+1} = X_N^k + y_N^k .$$

Step 1-2: Compute M_N dimensional vector z_N^k by

$$z_N^k = T_N(\tilde{X}_N^{k+1},s_{N-1}^k) - T_N(X_N^k,s_{N-1}^k).$$

Step 1-3: Compute G_N^{k+1} by

$$G_N^{k+1} = G_N^k - (G_N^kz_N^k+y_N^k)(y_N^k)^TG_N^k/(y_N^k)^TG_N^kz_N^k .$$

Step 1-4: Compute $\tilde{F}_N^k$ and $\nabla\tilde{F}_N^k$ by (4.5) and (4.6), respectively, and $\nabla^2\tilde{F}_N^k$ by (4.7) with $(-J_N^{-1})$ replaced by G_N^{k+1}.

The corresponding modification in Step 5 is obvious. If for n<N, M_n is large, then the corresponding step in Step 2 had better be modified in a way similar to the above modification of Step 1.

6. Modified Differential Dynamic Programming

In the D.D.P. algorithm discussed in the previous sections, equalities (3.3), (3.4), (3.9) and (3.10) in the Kuhn-Tucker conditions have played a major role, but inequalities (3.5), (3.6), (3.11) and (3.12) in the conditions have been intentionally neglected, because these inequalities are unnecessary for the local convergence of the D.D.P. algorithm. Thus, if the initial guess $\{X_n^0\}$ is far from the optimal solution of (P), then the D.D.P. algorithm has a tendency to generate a sequence $\{X_n^k\}$ converging to the unconstrained optimal solution or an optimal solution of (P) with some neglected constraints. This tendency will be corrected by taking into consideration those inequalities. A natural way to do so is to restrict X_n^{k+1} $(n=1,\dots,N)$ so that they satisfy those inequalities.

For given positive numbers ε_n $(n=1,\dots,N)$ and ε, put

$$I_n^k = \{j;\ h_n^j(x_n^k)\le\varepsilon_n,\ j=1,\dots,m_n\} \qquad (n=1,\dots,N,\ k=0,1,\dots)$$

and
$$I^k = \{j;\ \sigma_N^j(s_{N-1}^k, x_N^k)\le\varepsilon,\ j=1,\dots,m\} \qquad (k=0,1,\dots).$$

Moreover for given positive number $r_n<1$, define a simple modification of iteration procedure $U_n(X_n^k, s_{n-1}^k)$ as

$$V_n(X_n^k, s_{n-1}^k;\ \ell)=\{1-(r_n)^{\ell}\}X_n^k+(r_n)^{\ell}U_n(X_n^k, s_{n-1}^k) \qquad (n=1,\dots,N,\ k=0,1,\dots), \tag{6.1}$$

where ℓ is an appropriately chosen nonnegative number. For example, when $U_n(X_n^k, s_{n-1}^k)$ represents Newton's method (4.2), the corresponding modification $V_n(X_n^k, s_{n-1}^k;\ \ell)$ becomes

$$V_n(X_n^k, s_{n-1}^k;\ \ell) = X_n^k - (r_n)^{\ell}J_n^{-1}(X_n^k, s_{n-1}^k)T_n(X_n^k, s_{n-1}^k). \tag{6.2}$$

Let us denote by $\tilde V_n(X_n^k, s_{n-1}^k;\ \ell)$ the above modification of $\tilde U_n(X_n^k, s_{n-1}^k)$. A modification of the D.D.P. algorithm is made by using $\tilde V_n$ $(n=1,\dots,N-1)$ and V_N instead of $\tilde U_n$ $(n=1,\dots,N-1)$ and U_N.

Modified D.D.P. algorithm: Let $\{X_n^0;\ n=1,\dots,N\}$ and $\{s_n^0;\ n=0,\dots,N-1\}$ be given. Set k=0.

Step 1: Compute $\tilde X_N^{k+1}$ by

$$\tilde X_N^{k+1} = V_N(X_N^k, s_{N-1}^k;\ \ell), \tag{6.3}$$

where for given positive numbers ε_N' and ε', ℓ is the smallest nonnegative integer such that $h_N^j(\tilde x_N^{k+1})\le\varepsilon_N$ for $j\in I_N^k$, $\sigma_N^j(s_{N-1}^k, \tilde x_N^{k+1})\le\varepsilon$ for $j\in I^k$, $\tilde\lambda_N^{k+1\ j}\ge-\varepsilon_N'$ for j satisfying $\lambda_N^{kj}\ge-\varepsilon_N'$ and $\tilde\mu^{k+1\ j}\ge-\varepsilon'$ for j satisfying $\mu^{kj}\ge-\varepsilon'$ are all satisfied. Then compute $\tilde F_N^k$, $\nabla\tilde F_N^k$ and $\nabla^2\tilde F_N^k$ by (4.5), (4.6) and (4.7), respectively.

Step 2: Set n=N-1. Compute $\tilde X_n^{k+1}$ by

(6.4) $\tilde{X}_n^{k+1} = \tilde{V}_n(X_n^k, s_{n-1}^k;\ \ell)$,

where for given positive number ε_n', ℓ is the smallest nonnegative integer such that $h_n^j(\tilde{x}_n^{k+1}) \leq \varepsilon_n$ for $j \in I_n^k$ and $\tilde{\lambda}_n^{k+1\ j} \geq -\varepsilon_n'$ for j such that $\lambda_n^{kj} \geq -\varepsilon_n'$ are all satisfied. Then compute $\tilde{F}_n^k$, $\nabla\tilde{F}_n^k$ and $\nabla^2\tilde{F}_n^k$ by (4.9), (4.10) and (4.11), respectively. Set n=n-1 and repeat this step until n=2.

Step 3: Compute X_1^{k+1} by

(6.5) $X_1^{k+1} = \tilde{V}_1(X_1^k, s_0^k;\ \ell)$,

where for given positive number ε_1', ℓ is the smallest nonnegative integer such that $h_1^j(x_1^{k+1}) \leq \varepsilon_1$ for $j \in I_1^k$ and $\lambda_1^{k+1\ j} \geq -\varepsilon_1'$ for j such that $\lambda_1^{kj} \geq -\varepsilon_1'$ are all satisfied.

Step 4: For n=2,...,N-1, compute s_{n-1}^{k+1} by (4.13) and X_n^{k+1} by

(6.6) $X_n^{k+1} = \tilde{X}_n^{k+1} - r^{\ell}[\tilde{J}_n^{-1}\tilde{K}_n(X_n^k, s_{n-1}^k)](s_{n-1}^{k+1} - s_{n-1}^k)$,

where r is a given positive number less than one and ℓ is the smallest nonnegative integer such that $h_n^j(x_n^{k+1}) \leq \varepsilon_n$ for $j \in I_n^k$ and $\lambda_n^{k+1\ j} \geq -\varepsilon_n'$ for j such that $\lambda_n^{kj} \geq -\varepsilon_n'$ are all satisfied.

Step 5: Compute s_{N-1}^{k+1} by (4.15) and X_N^{k+1} by

(6.7) $X_N^{k+1} = \tilde{X}_N^{k+1} - r^{\ell}[J_N^{-1}K_N(X_N^k, s_{N-1}^k)](s_{N-1}^{k+1} - s_{N-1}^k)$,

where ℓ is the smallest nonnegative integer such that $h_N^j(x_N^{k+1}) \leq \varepsilon_N$ for $j \in I_N^k$, $\sigma_N^j(s_{N-1}^{k+1}, x_N^{k+1}) \leq \varepsilon$ for $j \in I^k$, $\lambda_N^{k+1\ j} \geq -\varepsilon_N'$ for j such that $\lambda_N^{kj} \geq -\varepsilon_N'$ and $\mu^{k+1\ j} \geq -\varepsilon'$ for j such that $\mu^{kj} \geq -\varepsilon'$ are all satisfied.

The following theorem states that points $\{X_n^k\}$ generated by the modified D.D.P. algorithm converges locally to the optimal solution $\{X_n^*\}$ and that its rate of convergence is the same as that of the D.D.P. algorithm.

Theorem 3. Suppose that all conditions in Theorem 2 are satisfied. Then the optimal solution $\{X_n^*\}$ of (P) is a point of attraction of the modified D.D.P. algorithm. Moreover its convergence is R-superlinear or R-linear according as the constant p in (C_7) is positive or zero.

The proof is given in Section 8. As shown in the proof, the modified D.D.P. algorithm has larger convergence domain than the D.D.P. algorithm. In much the same way as in the proof of Corollary 1, the following corollary can be proved.

Corollary 3. Suppose that Conditions (C_1) through (C_6) are satisfied. Then $\{X_n^*\}$ is a point of attraction of the modified D.D.P. algorithm with Newton's method and its convergence is R-quadratic.

7. Numerical Examples

As shown in Theorems 2 and 3, the optimal solution of (P) is a point of attraction of both the D.D.P. algorithm and the modified D.D.P. algorithm under Conditions (C_1) through (C_8). Various types of nonlinear programming problems satisfy Conditions (C_1) through (C_3). For example, objective functions f composed of the following functions $\xi(x,y)$ satisfy Conditions (C_1) and (C_3):

(7.1) $\xi(x,y) = c(x)+a(x)b(y),$

(7.2) $\xi(x,y) = c(x)+\{d(x)\}^{e(y)},$

(7.3) $\xi(x,y) = e(c(x)+a(x)b(y)),$

where all functions a, b, c, d, e are twice differentiable functions with uniformly continuous second derivatives, and a is nonnegative (nonpositive) valued, b is nondecreasing (nonincreasing), d is nonnegative valued and e is nondecreasing functions. As noted in Remark 2, separable programs and discrete time optimal control problems satisfy (C_1) through (C_3). Moreover, many large-scale nonlinear programming problems satisfy these conditions [9, 12]. Conditions (C_4) and (C_5) are regular conditions imposed frequently on nonlinear programming problems, and Condition (C_6) is satisfied, if $\nabla_x^2 L_n^*$ (n=1,...,N) are positive definite matrices. In the following, three examples are solved by using the D.D.P. algorithm or the modified D.D.P. algorithm with Newton's method. Thus Corollaries 1 and 3 show that Conditions (C_7) and (C_8) are satisfied.

Example 1. Minimize

$$\exp(x_1^2)+\exp(x_2^2+x_3^2)$$

subject to $x_1^2+x_1-4x_2+3 \le 0$.

The optimal solution (x_1^*, x_2^*, x_3^*, μ^*) is (-0.17264, 0.67227, 0.16807, 0.34577) and the optimal value is 2.64665. Clearly the objective function is decomposed by using $\xi_1(x,y)=\exp(x_1^2)+y$, $\xi_2(x_2,y)=\exp(x_2^2)y$ and $\xi_3(x_3)=\exp(x_3^2)$. Numerical computations with the termination criterion $\max_n \|x_n^k - x_n^{k+1}\| < 10^{-5}$ were carried out on the FACOM M-190 computer of Data Processing Center, Kyoto University. The optimal solution with over six-place accuracy was obtained. The results are shown in Table 1. The first column shows the initial values $(x_1^0, x_2^0, x_3^0, \mu^0)$. The symbol † means that the D.D.P. algorithm with Newton's method starting from the initial value gave the unconstrained optimal solution and hence the modified D.D.P. algorithm with Newton's method was used by setting ε=0.01, ε' =0.1 and r=0.5. The second and third columns describe the numbers of iterations and the computation times of the D.D.P. algorithm or the modified D.D.P. algorithm with Newton's method, respectively. Time is measured in milliseconds.

Table 1. Computational Results for Example 1.

Initial value $(x_1^0, x_2^0, x_3^0, \mu^0)$	Iteration	Time (m.s.)	s_3^0
(-1, 1, 1, 0.5)	6	19	-2
(0.5,0.5,0.5,0.5)	14	47	1.25
(1, 1, 1, 1)	7	24	0
(1.5,1.5,1.5,1.5)	8	26	-0.75
(2, 2, 2, 2)†	13	41	-1
(3, 3, 3, 3)	27	98	0

The fourth column indicates the values of s_3^0, i.e., the values of the constraint at the initial values.

Example 2. Minimize

$$x_1^2-5x_1+x_2^2-5x_2+2x_3^2-21x_3+x_4^2+7x_4$$

subject to $x_1^2+ x_1+x_2^2- x_2+ x_3^2+ x_3+x_4^2- x_4- 8 \le 0$

$$x_1^2- x_1+2x_2^2 + x_3^2 +2x_4^2-x_4-10 \le 0$$

and $2x_1^2+2x_1+ x_2^2-x_2+ x_3^2 -x_4- 5 \le 0.$

This is the well-known Rosen-Suzuki Test Problem [25]. The optimal solution $(x_1^*,x_2^*,x_3^*,x_4^*,\mu_1^*,\mu_2^*,\mu_3^*)$ is (0,1,2,-1,1,0,2) and the optimal value is -44. It is clear that all conditions except (C_4) are satisfied for the above problem. However, Condition (C_4) requires that the decomposition of the above problem should be three stages. Therefore, the objective function must be decomposed by $\xi_1(x_1,y)=x_1^2-5x_1+y$, $\xi_2(x_2,y)=x_2^2-5x_2+y$ and $\xi_3(x_3,x_4)=2x_3^2-21x_3+x_4^2+7x_4$. The numerical results are shown in Table 2. All details are the same as in Table 1. Table 2 shows that the D.D.P. algorithm or the modified D.D.P. algorithm with Newton's method can solve rather quickly the Rosen-Suzuki Test Problem.

Table2. Computational Results for Example 2.

Initial value $(x_1^0,x_2^0,x_3^0,x_4^0,\mu_1^0,\mu_2^0,\mu_3^0)$	Iteration	Time (m.s.)	s_3^0
(0,0,0,0,1,1,1)†	12	68	(-8,-10,-5)
(0,1,0,1,1,1,1)	9	51	(-8,-7,-6)
(1,1,1,1,1,1,1)†	14	78	(-4,-6,-1)
(-1,-1,-1,-1,1,1,1)†	18	103	(-4,-2,-1)
(1,-1,1,-1,1,1,1)	9	52	(0,-4,3)

Example 3. Minimize

$$-\prod_{n=1}^{30}\{1-(1-r_n)^{x_n}\}$$

subject to $\sum_{n=1}^{30} a_{mn}x_n \le b_m$ (m=1,2,3),

where the values of constants r_n, a_{mn} and b_m are given in Table 3. This is a relaxed version of an optimal redundancy allocation problem. Nakagawa, Nakajima and Hattori [18] have solved the above problem with integral constraints on x_n and different values of b_m. The optimal value of the above problem is -0.95473 and its optimal solution is shown in Table 3. As noted in Example 2, Condition (C_4) requires that the objective function should be decomposed by using $\xi_n(x_n,y)=\{1-(1-r_n)^{x_n}\}y$ (n=1,2,...,27) and $\xi_{28}(x_{28},x_{29},x_{30}) = -\prod_{n=28}^{30}\{1-(1-r_n)^{x_n}\}$. Table 4 shows the computational results. All the details

Table 3. Constants and the Optimal Solution of Example 3.

n	1	2	3	4	5	6	7	8	9	10
	11	12	13	14	15	16	17	18	19	20
	21	22	23	24	25	26	27	28	29	30
r_n	.90	.75	.65	.80	.85	.93	.78	.66	.78	.91
	.79	.77	.67	.79	.67	.94	.73	.79	.68	.98
	.90	.86	.95	.92	.83	.97	.89	.99	.88	.98
a_{1n}	5	4	9	7	7	5	6	9	4	5
	6	7	9	8	6	4	3	9	7	4
	9	8	6	3	4	5	7	6	8	7
a_{2n}	8	9	6	7	8	8	9	6	7	8
	9	7	6	5	7	8	4	9	3	9
	5	3	4	5	2	6	1	10	7	6
a_{3n}	2	4	10	1	5	5	4	8	8	10
	7	3	1	2	4	12	6	5	4	3
	5	9	2	5	7	8	6	3	12	5
x_n^*	3.031	4.491	5.333	4.211	3.407	2.571	4.146	5.335	4.041	2.654
	3.899	4.393	5.813	4.345	5.500	2.291	4.851	3.959	5.642	1.879
	2.958	3.286	2.539	2.785	3.771	1.995	3.183	1.609	2.904	1.869

$(b_1,b_2,b_3) = (700, 680, 585)$

$(\mu_1^*,\mu_2^*,\mu_3^*) = (5.173\times10^{-5}, 1.760\times10^{-4}, 2.366\times10^{-4})$

Table 4. Computational Results for Example 3.

Initial Value $(x_1^0,\ldots,x_{30}^0,\mu_1^0,\mu_2^0,\mu_3^0)$	Iteration	Time (m.s.)	s_{28}^0
a	7	477	(-40,-43.5,-30.5)
b	10	673	(-104,-107,-94)
(1,...,1,0.1,0.3,0.4)	13	873	(-513,-488,-419)

a=(2.5, 4, 5, 4, 3, 2.5, 4, 5, 4, 2.5, 4, 4, 5.5, 4, 5, 2, 4.5, 3.5, 5.5, 1.5, 3, 3, 2.5, 3, 4, 1.5, 3, 1.5, 3, 2, 1, 1, 1)

b=(3, 4, 5, 4, 3, 2, 4, 5, 4, 2, 3, 4, 5, 4, 5, 2, 4, 3, 5, 1, 2, 3, 2, 2, 3, 1, 3, 1, 2, 1, 0.1, 0.3, 0.4) are the same as in Table 1.

The above three examples show that the D.D.P. algorithm or the modified D.D.P. algorithm with Newton's method has both very rapid convergence property and rather large convergence domain. Since Corollary 2 implies that the computation time of the D.D.P. algorithm with Newton's method grows only linearly with N, the above examples suggest that the D.D.P. algorithm or the modified D.D.P. algorithm with Newton's method will solve large-scale nonlinear programming problems with several hundred variables within few minutes.

8. Convergence Proofs

This section deals with the proofs of Theorems 2 and 3. Equation (2.2) and Condition (C_3) imply that for n=1,...,N,

$$(8.1)\qquad \begin{aligned} \| s_n^k - s_n^* \| &\le \| \sigma_n(s_{n-1}^k, x_n^k) - \sigma_n(s_{n-1}^k, x_n^*(s_{n-1}^k)) \| + \| \sigma_n(s_{n-1}^k, x_n^*(s_{n-1}^k)) \\ &\quad - \sigma_n(s_{n-1}^k, x_n^*) \| + \| \sigma_n(s_{n-1}^k, x_n^*) - \sigma_n(s_{n-1}^*, x_n^*) \| \\ &\le \beta_{n1}(\delta_n^k + \| X_n^*(s_{n-1}^k) - X_n^*(s_{n-1}^*) \|) + \beta_{n2} \| s_{n-1}^k - s_{n-1}^* \| , \end{aligned}$$

where β_{n1} and β_{n2} are Lipschitz-constants of σ_n. Since by Lemma 1,

$$\| X_n^*(s_{n-1}^k) - X_n^*(s_{n-1}^*) \| \le \| J_n^{-1} \| \, \| K_n \| \, \| s_{n-1}^k - s_{n-1}^* \| \qquad \text{for } s_{n-1}^k \in 0_s^{n-1},$$

(8.1) implies that for n=1,...,N, there exist nonnegative numbers β_ℓ^n (ℓ=1,..., n) such that

$$(8.2)\qquad \| s_n^k - s_n^* \| \le \sum_{\ell=1}^{n} \beta_\ell^n \delta_\ell^k .$$

Consequently, for n=1,...,N,

$$(8.3)\qquad \begin{aligned} \| X_n^k - X_n^* \| &\le \delta_n^k + \| X_n^*(s_{n-1}^k) - X_n^*(s_{n-1}^*) \| \\ &\le \delta_n^k + \| J_n^{-1} \| \, \| K_n \| \sum_{\ell=1}^{n-1} \beta_\ell^{n-1} \delta_\ell^k , \end{aligned}$$

where for n=1, the summation over ℓ is assumed to be zero. Therefore, in order to prove that $\{X_n^*\}$ is a point of attraction of the D.D.P. algorithm or the modified D.D.P. algorithm, it suffices to prove that as $k\to\infty$, $\|\delta^k\| \to 0$ and that to any small number $\varepsilon_1 > 0$, there corresponds a number ε_2 such that $\| \delta^0 \| < \varepsilon_2$ implies $\| \delta^k \| < \varepsilon_1$ for all k; this is just uniform asymptotic stability of the origin of a system of difference equations for δ^k, if it exists.

To begin with, in order to prove Theorem 2, a system of difference equations for δ^k generated by the D.D.P. algorithm will be derived. From (4.8) and (4.14) it follows that for n=2,...,N-1,

$$(8.4)\qquad X_n^{k+1} = \tilde{U}_n(X_n^k, s_{n-1}^k) - [\tilde{J}_n^{-1} \tilde{K}_n(X_n^k, s_{n-1}^k)](s_{n-1}^{k+1} - s_{n-1}^k).$$

Consequently, since $\delta_n^{k+1} = \| X_n^{k+1} - X_n^*(s_{n-1}^{k+1}) \|$, for $n=2,\ldots,N-1$,

$$\begin{aligned}(8.5)\qquad \delta_n^{k+1} \le\; & \| U_n(X_n^k, s_{n-1}^k) - X_n^*(s_{n-1}^k) \| + \| \tilde{U}_n(X_n^k, s_{n-1}^k) - U_n(X_n^k, s_{n-1}^k) \| \\ & + \| -X_n^*(s_{n-1}^{k+1}) + X_n^*(s_{n-1}^k) - J_n^{-1} K_n(X_n^*(s_{n-1}^k), s_{n-1}^k)(s_{n-1}^{k+1} - s_{n-1}^k) \| \\ & + \| J_n^{-1} K_n(X_n^*(s_{n-1}^k), s_{n-1}^k) - \tilde{J}_n^{-1} \tilde{K}_n(X_n^k, s_{n-1}^k) \| \, \| s_{n-1}^{k+1} - s_{n-1}^k \| .\end{aligned}$$

Similarly, (4.4), (4.12) and (4.16) imply that

$$(8.6)\qquad \delta_1^{k+1} \le \| U_1(X_1^k, s_0^k) - X_1^*(s_0^*) \| + \| \tilde{U}_1(X_1^k, s_0^k) - U_1(X_1^k, s_0^k) \|$$

and

$$\begin{aligned}(8.7)\qquad \delta_N^{k+1} \le\; & \| U_N(X_N^k, s_{N-1}^k) - X_N^*(s_{N-1}^k) \| + \| -X_N^*(s_{N-1}^{k+1}) + X_N^*(s_{N-1}^k) \\ & - J_N^{-1} K_N(X_N^*(s_{N-1}^k), s_{N-1}^k)(s_{N-1}^{k+1} - s_{N-1}^k) \| \\ & + \| J_N^{-1} K_N(X_N^*(s_{N-1}^k), s_{N-1}^k) - J_N^{-1} K_N(X_N^k, s_{N-1}^k) \| \, \| s_{N-1}^{k+1} - s_{N-1}^k \| .\end{aligned}$$

Moreover, Condition (C_3) and Lemma 1 imply that for $n=2,\ldots,N$, there exist nonnegative numbers γ_1^n [23, p.73] and γ_2^n such that

$$(8.8)\qquad \| X_n^*(s_{n-1}^{k+1}) - X_n^*(s_{n-1}^k) + J_n^{-1} K_n (s_{n-1}^{k+1} - s_{n-1}^k) \| \le \gamma_1^n \| s_{n-1}^{k+1} - s_{n-1}^k \|^2 / 2,$$

$$(8.9)\qquad \| J_n^{-1} K_n(X_n^*(s_{n-1}^k), s_{n-1}^k) - J_n^{-1} K_n(X_n^k, s_{n-1}^k) \| \le \gamma_2^n \delta_n^k$$

and

$$\begin{aligned}(8.10)\qquad & \| J_n^{-1} K_n(X_n^*(s_{n-1}^k), s_{n-1}^k) - \tilde{J}_n^{-1} \tilde{K}_n(X_n^k, s_{n-1}^k) \| \le \gamma_2^n \delta_n^k + \| \tilde{J}_n^{-1} \| (\| K_n - \tilde{K}_n \| \\ & \qquad + \| J_n^{-1} \| \, \| K_n \| \, \| J_n - \tilde{J}_n \|).\end{aligned}$$

The following lemma is essential in evaluating the right-hand sides of (8.5) through (8.10).

Lemma 3. Suppose that Conditions (C_1) through (C_8) are satisfied. Then for $n=1,\ldots,N-1$, there exist nonnegative numbers $c_{\ell i}^{n+1}$, $d_{\ell i}^n$ ($\ell=n+1,\ldots,N$, $i=1,2,3$), η_n and e_ℓ^n ($\ell=1,\ldots,N$) such that for $X_n^k \in O_X^n$ and $s_{n-1}^k \in O_s^{n-1}$,

$$\text{(i)}\qquad \| F_{n+1}(s_n^k) - \tilde{F}_{n+1}^k \| \le \sum_{\ell=n+1}^{N} c_{\ell 1}^{n+1} (\delta_\ell^k)^{1+p},$$

$$\text{(ii)}\qquad \| \nabla F_{n+1}(s_n^k) - \nabla \tilde{F}_{n+1}^k \| \le \sum_{\ell=n+1}^{N} c_{\ell 2}^{n+1} (\delta_\ell^k)^{1+p},$$

$$\text{(iii)}\qquad \| \nabla^2 F_{n+1}(s_n^k) - \nabla^2 \tilde{F}_{n+1}^k \| \le \sum_{\ell=n+1}^{N} c_{\ell 3}^{n+1} \delta_\ell^k,$$

$$\text{(iv)}\qquad \| J_n(X_n^k, s_{n-1}^k) - \tilde{J}_n(X_n^k, s_{n-1}^k) \| \le \sum_{\ell=n+1}^{N} d_{\ell 1}^n \delta_\ell^k,$$

$$\text{(v)}\qquad \| K_n(X_n^k, s_{n-1}^k) - \tilde{K}_n(X_n^k, s_{n-1}^k) \| \le \sum_{\ell=n+1}^{N} d_{\ell 2}^n \delta_\ell^k,$$

$$\text{(vi)}\qquad \| U_n(X_n^k, s_{n-1}^k) - \tilde{U}_n(X_n^k, s_{n-1}^k) \| \le \sum_{\ell=n+1}^{N} d_{\ell 3}^n (\delta_\ell^k)^{1+p},$$

(vii) $\| \tilde{J}_n^{-1}(X_n^k, s_{n-1}^k) \| \leq \eta_n$,

(viii) $\| s_n^{k+1} - s_n^k \| \leq \sum_{\ell=1}^{N} e_\ell^n \delta_\ell^k$.

Proof: Since by (3.27), (4.5) and Condition (C_3), there exists a nonnegative number ζ_1 such that

$$\| F_N(s_{N-1}^k) - \tilde{F}_N^k \| = \| \xi_N(x_N^*(s_{N-1}^k)) - \xi_N(\tilde{x}_N^{k+1}) \| \leq \zeta_1 \| X_N^*(s_{N-1}^k) - \tilde{X}_N^{k+1} \|,$$

(4.4) and Condition (C_7) imply that (i) holds for n=N-1. Similarly, from (3.28), (3.29), (4.6) and (4.7) it follows that (ii) and (iii) hold for n=N-1. Therefore Condition (C_8) and Lemma 2 imply that (iv) through (vi) hold for n=N-1, because without loss of generality, $(\delta_{N-1}^k)^2 \leq (\delta_{N-1}^k)^{1+p} \leq \delta_{N-1}^k$. Now suppose that (i) through (vi) hold for n. Equations (3.24) and (4.9) and Condition (C_3) imply that for nonnegative numbers ζ_2, ζ_3, ζ_4 and ζ_5,

$$\begin{aligned} \| F_n(s_{n-1}^k) - \tilde{F}_n^k \| &= \| \xi_n(x_n^*(s_{n-1}^k), F_{n+1}(\sigma_n(s_{n-1}^k, x_n^*(s_{n-1}^k)))) - \xi_n(\tilde{x}_n^{k+1}, \tilde{F}_{n+1}^k) \\ &\quad - \frac{\partial}{\partial y} \xi_n(\tilde{x}_n^{k+1}, \tilde{F}_{n+1}^k) \nabla_x \sigma_n(s_{n-1}^k, x_n^k)(\tilde{x}_n^{k+1} - x_n^k) \| \\ &\leq (\zeta_2 + \| \frac{\partial}{\partial y} \xi_n \| \, \| \nabla \tilde{F}_{n+1}^k \| \, \| \nabla_x \sigma_n \|) \| X_n^*(s_{n-1}^k) - \tilde{X}_n^{k+1} \| \\ &\quad + \zeta_3 (\delta_n^k)^2/2 + (\zeta_4 + \zeta_5 \| \nabla F_{n+1} \| \| \nabla_x \sigma_n \| \delta_n^k) \| F_{n+1}(s_n^k) - \tilde{F}_{n+1}^k \| \\ &\quad + \| \frac{\partial}{\partial y} \xi_n \| \, \| \nabla_x \sigma_n \| \delta_n^k \| \nabla F_{n+1}(s_n^k) - \nabla \tilde{F}_{n+1}^k \|. \end{aligned}$$

Since $\delta_n^k (\delta_\ell^k)^{1+p} \leq (\delta_\ell^k)^{1+p}$, the above inequality, Condition (C_7) and the assumption that (i) and (ii) hold for n prove (i) for n-1. In a similar way, (ii) and (iii) can be proved for n-1. Consequently, Condition (C_8) and Lemma 2 imply that (iv) through (vi) hold for n-1. The proofs of (i) through (vi) are concluded. From perturbation lemma [23, p.45] it follows that if $\| J_n^{-1}(X_n^k, s_{n-1}^k) \| \leq \zeta_6$, $\| J_n(X_n^k, s_{n-1}^k) - \tilde{J}_n(X_n^k, s_{n-1}^k) \| \leq \zeta_7$ and $\zeta_6 \zeta_7 < 1$, then $\| \tilde{J}_n^{-1}(X_n^k, s_{n-1}^k) \| \leq \zeta_6/(1-\zeta_6\zeta_7)$. Hence (iv) leads to (vii). Since $s_1^{k+1} = \sigma_1(0, x_1^{k+1})$ and $X_1^{k+1} = \tilde{U}_1(X_1^k, 0)$, for the Lipschitz constant β_{11},

$$\begin{aligned} \| s_1^{k+1} - s_1^k \| \leq \beta_{11} \| \tilde{U}_1(X_1^k, 0) - X_1^k \| \leq \beta_{11} \{ \| X_1^* - X_1^k \| + \| U_1(X_1^k, 0) - X_1^* \| \\ + \| \tilde{U}_1(X_1^k, 0) - U_1(X_1^k, 0) \| \}. \end{aligned}$$

Consequently, Condition (C_7) and (vi) of Lemma 3 imply that

$$\| s_1^{k+1} - s_1^k \| \leq \beta_{11} \{ (1 + a_1 (\delta_1^k)^p) \delta_1^k + \sum_{\ell=2}^{N} d_{\ell 3}^1 (\delta^k)^{1+p} \}.$$

This proves (viii) for n=1. Suppose that (viii) holds for n. Then (8.4) implies that for Lipschitz constants β_{n+11} and β_{n+12},

$$\begin{aligned} \| s_{n+1}^{k+1} - s_{n+1}^k \| \leq \beta_{n+11} \{ \| X_{n+1}^*(s_n^k) - X_{n+1}^k \| + \| U_{n+1}(X_{n+1}^k, s_n^k) - X_{n+1}^*(s_n^k) \| \\ + \| \tilde{U}_{n+1}(X_{n+1}^k, s_n^k) - U_{n+1}(X_{n+1}^k, s_n^k) \| \} + \{ \beta_{n+12} \end{aligned}$$

$$+\ \beta_{n+11}\|\tilde{J}_{n+1}^{-1}\tilde{K}_{n+1}\|\}\|s_n^{k+1}-s_n^k\|\ .$$

Therefore Condition (C_7) and (vi) and (vii) of Lemma 3 prove (viii) for n+1. The proof of Lemma 3 is concluded.

By using Condition (C_7) and (iv) through (vii) of Lemma 3, combination of (8.5) with (8.8) and (8.10) yields for n=2,...,N-1,

$$\text{(8.11)}\qquad \delta_n^{k+1} \le a_n(\delta_n^k)^{1+p}+\sum_{\ell=n+1}^{N} d_{\ell 3}^n(\delta_\ell^k)^{1+p}+\{\gamma_2^n\delta_n^k+\eta_n\sum_{\ell=n+1}^{N}(d_{\ell 2}^n +\|J_n^{-1}\|\,\|K_n\|\,d_{\ell 1}^n)\delta_\ell^k\}\|s_{n-1}^{k+1}-s_{n-1}^k\|+\gamma_1^n\|s_{n-1}^{k+1}-s_{n-1}^k\|^2/2.$$

Since $\delta_n^k\delta_\ell^k\le\{(\delta_n^k)^2+(\delta_\ell^k)^2\}/2$, (8.11) and (viii) of Lemma 3 imply that for n=2,..., N-1, there exist nonnegative numbers q_ℓ^n and r_ℓ^n such that

$$\text{(8.12)}\qquad \delta_n^{k+1} \le \sum_{\ell=n}^{N} q_\ell^n(\delta_\ell^k)^{1+p}+\sum_{\ell=1}^{N} r_\ell^n(\delta_\ell^k)^2,$$

where $q_n^n=a_n$ and $q_\ell^n=d_{\ell 3}^n$ ($\ell>n$). In a similar way, (8.6) and (8.7) imply that (8.12) holds for n=1 and n=N, where $r_\ell^1=0$ ($\ell=1,...,N$). Consequently, by using N×N matrices $Q(\delta)=(q_\ell^n(\delta_\ell)^p)$ and $R(\delta)=(r_\ell^n\delta_\ell)$ with $q_\ell^n=0$ ($\ell<n$) and $r_\ell^1=0$, (8.12) can be rewritten as

$$\text{(8.13)}\qquad \delta^{k+1} \le Q(\delta^k)\delta^k+R(\delta^k)\delta^k.$$

Clearly, in order to prove uniform asymptotic stability of the origin of δ^k, it sufficies to prove uniform asymptotic stability of the origin of the following system of difference equations:

$$\text{(8.14)}\qquad \delta^{k+1} = Q(\delta^k)\delta^k+R(\delta^k)\delta^k.$$

When the constant p in (C_7) is positive, $Q(\delta)$ and $R(\delta)$ are continuous in δ and $Q(0)=R(0)=0$. Therefore $\|Q(\delta)+R(\delta)\|<1$ for δ belonging to an appropriate neighbourhood of the origin. This implies that the origin is uniformly asymptotically stable [11]. Setting p=0 in (8.14) yields

$$\text{(8.15)}\qquad \delta^{k+1} = Q\delta^k+R(\delta^k)\delta^k,$$

where $q_n^n=a_n$, $q_\ell^n=d_{\ell 3}^n$ ($\ell>n$) and $q_\ell^n=0$ ($\ell<n$). Thus Condition (C_7) implies that all eigenvalues of Q are less than one in absolute value. Moreover $R(\delta)$ is continuous in δ and $R(0)=0$. Therefore the origin is uniformly asymptotically stable [11]. Thus it has been proved that the optimal solution $\{X_n^*\}$ of (P) is a point of attraction of the D.D.P. algorithm.

Since (8.3) implies that

$$\sum_{n=1}^{N}\|X_n^k-X_n^*\| \le \rho_1\sum_{n=1}^{N}\delta_n^k = \rho_1\|\delta^k\|\ ,$$

in order to show the convergence rate of the D.D.P. algorithm it suffices to show the convergence rate of $\|\delta^k\|$, where

$$\rho_1 = \max_n\{1+\sum_{\ell=n+1}^{N} \| J_\ell^{-1}\| \, \| K_\ell\| \, \beta_n^{\ell-1}\}.$$

Denote the ℓ_∞-norm by $\|\cdot\|_\infty$, that is, $\|\delta^k\|_\infty = \max_n \delta_n^k$. From (8.13) it follows that

$$\|\delta^k\| \le \rho_2 \|\delta^{k-1}\|_\infty^p \|\delta^{k-1}\|,$$

where $\rho_2 = \max_\ell\{\sum_{n=1}^{N}(q_\ell^n + r_\ell^n)\}$. Consequently, in the case of $p>0$,

$$\lim \sup_{k\to\infty} \|\delta^k\|^{1/k} \le \rho_2 \|\delta^0\| \lim \sup_{k\to\infty} (\prod_{j=1}^{k-1} \|\delta^j\|_\infty)^{p/k} = 0$$

because $\|\delta^k\|_\infty \to 0$ as $k\to\infty$. This proves that when $p>0$, the convergence of the D.D.P. algorithm is R-superlinear. Similarly it can be proved that when $p=0$, the convergence is R-linear.

Let us proceed to the proof of Theorem 3. From Condition (C_7) and (6.1) it follows that

$$\| V_n(X_n^k, s_{n-1}^k;\ \ell) - X_n^*(s_{n-1}^k)\| \le \{1-(1-a_n(\delta_n^k)^p)(r_n)^\ell\}\delta_n^k.$$

This implies that V_n satisfies Condition (C_7) with $p=0$ in Theorem 2. Since V_n satisfies also Condition (C_8) in Theorem 2, in much the same way as in the proof of Theorem 2 it can be proved that the optimal solution $\{X_n^*\}$ of (P) is a point of attraction of the modified D.D.P. algorithm. Moreover, since there exists positive integer k_0 such that for all $k\ge k_0$, $V_n(X_n^k, s_{n-1}^k, \ell) = U_n(X_n^k, s_{n-1}^k)$ $(n=1,\dots,N)$ and the integers ℓ in Steps 4 and 5 are always zero, the rate of convergence of the modified D.D.P. is the same as that of the D.D.P. algorithm. The proof of Theorem 3 is concluded.

9. Conclusion

In this paper, the D.D.P. algorithm and the modified D.D.P. algorithm for solving large-scale nonlinear programming problem (P) have been proposed and their local convergence has been proved. Moreover, it is shown that the rates of convergence of the present algorithms with Newton's method are R-quadratic. Numerical examples show the efficiency of the present algorithms. Since the present algorithms are based upon Kuhn-Tucker conditions for subproblems (P_n) decomposed by dynamic programming, they inherit desirable properties of dynamic programming. In particular, both the operation count and the magnitude of the required core memory for the D.D.P. algorithm with Newton's method grow only linearly with the number of variables. Thus the numerical examples suggest that the present algorithms with Newton's method will solve large-scale nonlinear programming problems with several hundred variables within few minutes.

Throughout this paper, it is assumed that (P) satisfies Condition (C_2).

If (P) satisfies Condition (C_2') and the functions σ_n^{-1} in (2.6) are twice continuously differentiable, then the results obtained in this paper remain valid with some modification.

In order to start the present algorithms, it is necessary to obtain an initial guess $\{X_n^0\}$ belonging to an neighbourhood of the optimal solution $\{X_n^*\}$. The conventional dynamic programming with coarse grid will provide a good initial guess $\{x_n^0\}$. Then a good initial guess $\{\lambda_n^0\}$ and μ^0 will be obtained by the method in [15]. This approach, however, will take much time. Therefore it is hoped to investigate a global stabilization of the present algorithms.

Finally, it should be noted that unless the matrices J_n given by (3.17) and (3.18) are singular, the present algorithms can be performed even if the initial guess $\{X_n^0\}$ is far from the optimal solution and/or infeasible. In this case, however, the present algorithms generate sequences of $\{X_n^k\}$ which converge to some points $\{X_n'\}$ or diverge unboundedly, where $\{X_n'\}$ may be an local optimal solution of (P) in which some constraints are dropped. If $\{X_n'\}$ satisfies all inequalities (3.5) through (3.7) and (3.11) through (3.13), then it is an local optimal solution of (P). Moreover if (P) is a convex program, then it is the optimal solution of (P). Since the convergence of the present algorithms is very rapid and their convergence domains are quite large, the present algorithms can solve rather easily large-scale nonlinear programming problems by adjusting the initial values $\{X_n^0\}$.

Acknowledgement

The author would like to express his appreciation to Professor H. Mine for valuable discussions and to the refrees for their helpful comments. The author also is indebted to Mr. K. Mano for his help in computing Example 3.

References

[1] Bellman, R. and Dreyfus, S.: *Applied Dynamic Programming*, Princeton Univ. Press, Princeton, N. J., 1962.

[2] Broyden, C. G.: A Class of Methods for solving Nonlinear Simultaneous Equations, *Math. Computation*, 19, 577-593, (1965).

[3] Dyer, P. and McReynolds, S. R.: *The Computation and Theory of Optimal Control*, Academic Press, New York, 1970.

[4] Fiacco, A. V. and McCormick, G. P.: *Nonlinear Programming: SUMT*, John Wiley, New York, 1968.

[5] Gershwin, S. B. and Jacobson, D. H.: A Discrete-Time Differential Dynamic Programming Algorithm with Application to Optimal Orbit Transfer, *AIAA Journal* 8, 1616-1626 (1970).

[6] Hald, O. H.: On a Newton-Moser Type Method, *Numer. Math.* 23, 411-426 (1975).

[7] Havira, R. M. and Lewis, J. B.: Computation of Quantized Controls Using Differential Dynamic Programming, *IEEE* AC-17, 191-196 (1972).

[8] Hearon, J. Z.: On the Singularity of a Certain Bordered Matrix, *SIAM J. Appl. Math.* 15, 1413-1421 (1967).

[9] Himmelblau, D. M.: *Decomposition of Large-Scale Problems*, North-Holland, Amsterdam, 1973.

[10] Jacobson, D. H. and Mayne, D. Q.: *Differential Dynamic Programming*, American Elsevier, New York, 1970.

[11] Kalman, R. E. and Bertram, J. E.: Control System Analysis and Design via the "Second Method" of Lyapunov, II. Discrete-Time System, *Trans. of ASME, J. Basic Eng.* 82, 394-400 (1960).

[12] Lasdon, L. S.: *Optimization Theory for Large Systems*, Macmillan, New York, 1970.

[13] Mayne, D. Q.: Differential Dynamic Programming - A Unified Approach to the Optimization of Dynamic Systems, in *Control and Dynamic Systems*, Vol. 10 ed. by C. T. Leondes, Academic Press, New York, 1973.

[14] Mayne, D. Q. and Polak, E.: First-Order Strong Variation Algorithms for Optimal Control, *J. Optimization Theory Appl.* 16, 277-301 (1975).

[15] Miele, A. and Levy, A. V.: Modified Quasilinearization and Optimal Choice of the Multipliers, Part 1 - Mathematical Programming Problems, *J. Optimization Theory and Appl.* 6, 364-380 (1970).

[16] Mine, H. and Ohno, K.: Decomposition of Mathematical Programming Problems by Dynamic Programming and Its Application to Block-Diagonal Geometric Programs, *J. Math. Anal. Appl.* 32, 370-385 (1970).

[17] Mine, H., Ohno, K. and Fukushima, M.: Multilevel Decomposition of Nonlinear Programming Problems by Dynamic Programming, *J. Math. Anal. Appl.* 56, 7-27 (1976).

[18] Nakagawa, Y., Nakashima, K. and Hattori, Y.: Optimal Reliability Allocation by Branch-and-Bound Technique, To be published in *IEEE Trans. Reliability* R-27, April, 1978.

[19] Nemhauser, G. L.: *Introduction to Dynamic Programming*, John Wiley, New York, 1966.

[20] Ohno, K.: Differential Dynamic Programming for Solving Discrete-Time Optimal Control Problems with Constraints on State Variables, *Systems*

and Control 21, 454-461 (1977) (in Japanese).

[21] Ohno, K.: A New Approach to Differential Dynamic Programming for Discrete Time Systems, *IEEE Trans. Automatic Control* AC-23, 37-47 (1978).

[22] Ohno, K.: Differential Dynamic Programming and Separable Programs, To be published in *J. Optimization Theory and Appl.* 24, 607-627 (1978).

[23] Ortega, J. M. and Rheinboldt, W. C.: *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[24] Polak, E. and Mayne, D. Q.: First-Order Strong Variation Algorithms for Optimal Control Problems with Terminal Inequality Constraints, *J. Optimization Theory Appl.* 16, 303-325 (1975).

[25] Rosen, J. B. and Suzuki, S.: Construction of Nonlinear Programming Test Problems, *Communications of the ACM* 8, 113 (1965).

Katsuhisa OHNO: Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Yoshida-Honmachi, Sakyo-Ku, Kyoto 606, Japan.