Automatica, Vol. 28, No. 6, pp. 1249–1254, 1992
Printed in Great Britain.

0005–1098/92 $5.00 + 0.00
Pergamon Press Ltd

Brief Paper

Complete Decomposition Algorithm for Nonconvex Separable Optimization Problems and Applications*†

SHIN-YEU LIN‡

Key Words—Decomposition; optimization and optimal control.

Abstract—In this paper, we present a complete decomposition algorithm for nonconvex separable optimization problems applied in the optimal control problems. This complete decomposition algorithm combines recursive quadratic programming with the dual method. When our algorithm is applied to discretized optimal control problems, a simple and parallel computation and a simple and regular data flow pattern between consecutive computational steps results. This paper also suggests an approach for developing a hardware implementation of our algorithm and gives an estimation of the execution time needed to solve a practical example.

1. *Introduction*

A NONCONVEX SEPARABLE *optimization problem* of the following form is considered in this paper:

$$\min_{y} \sum_{i=1}^{k} z_i(y_i),$$

subject to:

$$\sum_{i=1}^{k} h_i(y_i) = 0,$$

$$\sum_{i=1}^{k} c_i(y_i) \le 0, \tag{1}$$

where $y = (y_1, \ldots, y_k) \in \mathbb{R}^n$, $y_i \in \mathbb{R}^{n_i}$ and $\sum_{i=1}^{k} n_i = n$; and constraint functions $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^m$, $c_i : \mathbb{R}^{n_i} \to \mathbb{R}^q$ and objective functions $z_i : \mathbb{R}^{n_i} \to \mathbb{R}$, $i = 1, 2, \ldots, k$, are twice continuously differentiable in y_i.

The dual method is ideally suited for solving such large-scale separable optimization problem, because it has been shown (Luenburger, 1984) that the dual function of (1) can be decomposed into k independent smaller minimization subproblems. However, directly applying the dual method to (1) may fail in general since the *Hessian matrix* of the *Lagrangian* of (1) may not be positive definite. To cope with this difficulty, the augmented Lagrangian method can be used to convexify the Lagrangian; however, this destroys the separability of (1) due to the cross product terms in the added quadratic penalty function. In response to this problem, numerous techniques have been developed to find a proper Lagrangian. Among the leading methods, the approaches proposed by Watanabe *et al.* (1978) and Tatjewshi (1989) were both based on approximating the *augmented Lagrangian* to maintain the separability; while those of Bersekas (1979) and Tanikawa and Mukai (1985) used a convexification procedure, which is different from the augmented Lagrangian method, to ensure the positive definiteness of the Hessian matrix of the Lagrangian and preserve the separability. All the above techniques must, at least, solve k independent minimization subproblems in each iteration. Thus, an additional iterative optimization program is needed for these subproblems.

Compared to the above techniques, our approach is simpler because we do not seek a proper Lagrangian. Instead, we employ the recursive quadratic programming technique proposed by Han (1977) to successively solve the quadratic approximate problem of (1); the quadratic approximate problem is formulated to have a positive definite diagonal Hessian matrix. Thus, the separability of (1) will be preserved in the quadratic approximate problem, which can then be solved by the dual method. Furthermore, a complete decomposition property can be obtained because each decomposed quadratic subproblem can be solved analytically owing to the diagonal Hessian matrix. The construction of this positive definite diagonal Hessian matrix, say B, of the quadratic approximate problem is based on the diagonal terms of the matrix $\nabla^2 \sum_{i=1}^{k} z_i(y_i)$ such that $[\text{diag } B]_j = [\text{diag } \nabla^2 \sum_{i=1}^{k} z_i(y_i)]_j$ if it is positive; otherwise, $[\text{diag } B]_j = \gamma$, a small positive scalar. Thus in addition to the complete decomposition property obtained by using the dual method to solve each quadratic approximate problem, our technique will function like a *variable metric method* (Han, 1976) if $B = \nabla^2 \sum z_i(y_i)$ and the constraint functions are affine, or like the *gradient projection method*§ if $B = \gamma I$. Therefore, our technique is best suited to obtain the discrete solution of fixed final-time constrained nonlinear optimal control problems with quadratic performance index. In general, such problems are considered to be difficult, especially in the presence of control constraints. However, we show in this paper that our technique for such problems exhibits good convergence, even though an augmented Lagrangian method is incorporated to solve the problem with terminal state constraints. Furthermore, the complete decomposition property of our technique for solving such problems will not only result in simple and parallel computation but also a simple and regular data flow pattern between consecutive computational steps. This naturally leads to the use of data driven waveform VLSI array processors (Kung, 1988) to implement our technique. A detailed description of the implementation process is beyond the scope of this paper. However, a version of the hardware used to implement a general minimum-time control algorithm proposed by Lin (1992b) can be modified for the implementation of our technique.

At the end of this paper, we present a practical optimal control example among several examples which are tested by our technique. The convergence rate is slow as expected

* Received 24 December 1989; revised 11 April 1992; received in final form 28 April 1992. The original version of this paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor M. Jamshidi under the direction of Editor A. P. Sage.

† This work was supported by National Science Council under Grant NSC-78-0404-E-009-33.

‡ Department of Control Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

§ Mukai and Polak (1978) developed an efficient scheme with finite inner iterations.

because of the simplicity of our technique. However, based on current VLSI technology, the estimated execution time is fast.

2. *RQPD (recursive quadratic programming with dual method) algorithm for nonconvex separable optimization problems*

Preliminaries. The quadratic programming problem (2) describes an approximation of (1) at the point $y(l)$,

$$\min_{d} \sum_{i=1}^{k} \nabla_{y_i} z_i(y_i(l)) d_i + \tfrac{1}{2} d_i^T B_i(y_i(l)) d_i,$$

subject to:

$$\begin{cases} \sum_{i=1}^{k} h_i(y_i(l)) + \nabla_{y_i} h_i(y_i(l)) d_i = 0, \\ \sum_{i=1}^{k} c_i(y_i(l)) + \nabla_{y_i} c_i(y_i(l)) d_i] \le 0, \end{cases} \tag{2}$$

where $d_i \in \mathbb{R}^{n_i}$, $\nabla_{y_i} h_i(\cdot)$ is an $m \times n_i$ matrix, $\nabla_{y_i} c_i(\cdot)$ is a $q \times n_i$ matrix, $\nabla_{y_i} z_i(\cdot)$ is an n_i dimensional row vector, $B_i(y_i(l))$ is an $n_i \times n_i$ positive definite diagonal matrix such that its jth diagonal element

$$b_{ij} = \begin{cases} \left.\dfrac{\partial^2 z}{\partial y_{ij}^2}\right|_{y_i(l)}, & \text{if } \left.\dfrac{\partial^2 z}{\partial y_{ij}^2}\right|_{y_i(l)} > 0; \\ \gamma, & \text{otherwise,} \end{cases}$$

where $\gamma > 0$ is a scalar, y_{ij} is the jth element of the vector y_i, and $(\cdot)^T$ denotes the transpose of $(\cdot)$.

A recursive quadratic programming method for solving (1) proposed by Han (1977) is to solve (2) recursively with updating procedures by (3):

$$y_i(l+1) = y_i(l) + \tilde{\alpha} d_i^*, \quad i = 1, \ldots, k, \tag{3}$$

where d_i^*, which is the optimal solution of (2), is a descent direction of (1) at $y(l)$ in the sense of the *absolute-value penalty function* (Han, 1977); and the $\tilde{\alpha}$ is determined by the one-dimensional line search method to minimize the absolute-value penalty function of (1) along the direction $d^* = (d_1^* \cdots d_k^*)$.

Despite the search of the stepsize, the most cumbersome computational procedure of the recursive quadratic programming method is to solve (2) in each iteration l. Clearly, the dual method is ideally suited for solving (2) because it is separable and has a positive definite Hessian matrix.

To proceed with the dual method, the dual problem of (2) should be described first. It is shown below:

$$\max_{\lambda, \mu} \Phi(\lambda, \mu), \quad \mu \ge 0, \tag{4}$$

where the dual function $\Phi(\lambda, \mu)$ is a function of the Lagrange multiplier (λ, μ), $\lambda \in \mathbb{R}^m$, $\mu \in \mathbb{R}^q$, such that

$$\Phi(\lambda, \mu) = \min_{d} \left\{ \sum_{i=1}^{k} [\nabla_{y_i} z_i(y_i(l)) d_i + \tfrac{1}{2} d_i^T B_i(y_i(l)) d_i] + \left[\sum_{i=1}^{k} H_i(y_i(l), d_i)\right]^T \lambda + \left[\sum_{i=1}^{k} C_i(y_i(l), d_i)\right]^T \mu \right\}, \tag{5}$$

where $H_i(y_i(l), d_i) = h_i(y_i(l)) + \nabla_{y_i} h_i(y_i(l)) d_i$, $C_i(y_i(l), d_i) = c_i(y_i(l)) + \nabla_{y_i} c_i(y_i(l)) d_i$.

The dual method we employed to solve (2) uses the gradient ascent method to solve (4). Its iterative procedures are

$$\begin{aligned} \lambda^t(j+1) &= \lambda^t(j) + \bar{\beta} \nabla_{\lambda^t} \Phi(j), \quad t = 1, \ldots, m, \\ \mu^v(j+1) &= \mu^v(j) + \bar{\beta} \nabla_{\mu^v} \Phi(j), \quad v = 1, \ldots, q, \end{aligned} \tag{6}$$

where j denotes the iteration index, $\Phi(j)$ denotes $\Phi(\lambda, \mu)$ at the jth iteration, λ^t and μ^v are the tth and vth components of λ and μ, respectively, and the stepsize $\bar{\beta}$ is obtained by the one-dimensional line search method such that $\bar{\beta} = \arg\max_{\beta \in \mathcal{B}} \{\Phi(\lambda(j) + \beta \nabla_\lambda \Phi(j), \mu(j) + \beta \nabla_\mu \Phi(j))\}$, where $\mathcal{B} = \{\beta \in \mathbb{R} \mid \mu(j) + \beta \nabla_\mu \Phi(j) \ge 0\}$, and all components of $(\nabla_\lambda \Phi(j), \nabla_\mu \Phi(j))$ can be computed according to the following formula:

$$\begin{aligned} \nabla_{\lambda^t} \Phi(j) &= \sum_{i=1}^{k} H_i^t(y_i(l), \hat{d}_i), \\ \nabla_{\mu^v} \Phi(j) &= \sum_{i=1}^{k} C_i^v(y_i(l), \hat{d}_i), \end{aligned} \tag{7}$$

provided that the minimum solution $\hat{d}$ of (5) with $(\lambda, \mu) = (\lambda(j), \mu(j))$ is obtained. In (7), $H_i^t(y_i(l), \hat{d}_i)$ and $C_i^v(y_i(l), \hat{d}_i)$ denote the tth and vth component of $H_i(y_i(l), \hat{d}_i)$ and $C_i(y_i(l), \hat{d}_i)$, respectively.

For a given λ, due to the separability of (2), (5) can be decomposed into k independent minimization subproblems and each subproblem i can be solved analytically as follows:

$$\hat{d}_i = B_i(y_i(l))^{-1} [\nabla_{y_i} z_i(y_i(l))^T + \nabla_{y_i} h_i(y_i(l))^T \lambda(j) + \nabla_{y_i} c_i(y_i(l))^T \mu(j)]. \tag{8}$$

In addition to the parallelization of computing the k $\hat{d}_i$ vectors, there is also a trivial parallelization of computing the components of each $\hat{d}_i$, because $B_i(y_i(l))^{-1}$ is a diagonal matrix. Moreover, all the components of $\nabla_\lambda \Phi(j)$ and $\nabla_\mu \Phi(j)$ can also be computed in parallel according to (7). This indicates that the RQPD method has a *complete decomposition property*.

RQPD algorithm. Based on the above developments, we are ready to state the RQPD algorithm for nonconvex separable optimization problems:

Algorithm I.

Step 0. Set the values of γ, $y(0)$, and let $i = 0$. *Step* 1. Guess $\lambda(0)$, and $\mu(0)$ (≥ 0); and let $j = 0$. *Step* 2. Compute $\hat{d}$ in parallel by (8). *Step* 3. Compute $(\nabla_\lambda \Phi(j), \nabla_\mu \Phi(j))$ in parallel by (7). *Step* 4. If $\{|\nabla_\lambda \Phi(j)|_\infty, |\nabla_\mu \Phi(j)|_\infty\} < \epsilon_1$, go to Step 6; otherwise, go to Step 5. *Step* 5. Update $(\lambda(j+1), \mu(j+1))$ in parallel by (6), and return to Step 2. *Step* 6. If $|\hat{d}|_\infty < \epsilon_2$, stop, and output the solution $y(l)$; otherwise, update $y(l+1)$ in parallel by (3), and return to Step 1. (Note: ϵ_1 and ϵ_2 are preselected accuracies, and $|(\cdot)|_\infty$ denotes the infinity norm of $(\cdot)$.)

Convergence analysis. The convergence of the recursive quadratic programming method and the dual method have been analyzed by Han (1977) and Bazaraa and Shetty (1979), respectively. From their results, we easily obtain the following sufficient conditions to ensure the convergence of the RQPD algorithm:

Corollary 1. Assume that (i) there exist two positive scalars δ and η such that for each l, $\delta y_i^T y_i < y_i^T B_i(y_i(l)) y_i < \eta y_i^T y_i$, for any $y_i \in \mathbb{R}^{n_i}$, $i = 1, 2, \ldots, N$, (ii) there exists a unique solution of (2) for any given value of $y(l) \in \mathbb{R}^n$, and the corresponding Lagrange multiplier is bounded, and (iii) the constraint qualification of (2) that there exists a d' such that $\sum_{i=1}^{k} h_i(y_i(l)) + \nabla_{y_i} h_i(y_i(l)) d_i' = 0$, $\sum_{i=1}^{k} c_i(y_i(l)) + \nabla_{y_i} c_i(y_i(l)) d_i' < 0$ holds true for each l. Then any bounded sequence $\{y(l)\}$ generated by the RQPD algorithm will converge to a Kuhn–Tucker point of (1).

Remark. We adopt the version of the convergence theorem of the recursive quadratic programming method described by Luenburger (1984). It differs from the original theorem shown by Han (1977) by having two additional assumptions: (1) the sequence $\{y(l)\}$ generated should be bounded, and (2) the solution of (2) exists under any given value of $y(l) \in \mathbb{R}^n$.

3. *Application to optimal control problems*

The RQPD algorithm we proposed is especially suitable for fixed final-time, nonlinear, multivariable optimal control problems with quadratic performance index and control variable inequality constraints.

Mathematically, such optimal control problems can be expressed as

$$\min J(x, u) \Big[= x(t_f)^T M x(t_f) + \frac{1}{2} \int_{t_0}^{t_f} u^T(t) R(t) u(t) + x^T(t) Q(t) x(t)\, dt \Big], \tag{9a}$$

subject to:

$$\dot{x}(t) = f'(x(t), u(t), t); \quad x(t_0) = x_0, \tag{9b}$$

$$g(u(t)) \le 0, \tag{9c}$$

$$T(x(t_f)) = 0, \tag{9d}$$

where the vector of state variables $x \in \mathbb{R}^m$, the vector of control variables $u \in \mathbb{R}^p$, the functions $f': \mathbb{R}^{m+p} \to \mathbb{R}^m$, $g: \mathbb{R}^p \to \mathbb{R}^q$, $T: \mathbb{R}^m \to \mathbb{R}^m$ are twice continuously differentiable in $x(t)$, $u(t)$, and the matrices $R(t) \in \mathbb{R}^{p \times p}$ are positive definite, while $M \in \mathbb{R}^{m \times m}$, $Q(t) \in \mathbb{R}^{m \times m}$ are positive semidefinite matrices. The typical performance index for our problems might be minimum energy (e.g. $Q = 0$) or regulator and tracking problems, etc. Problem (9a)–(9d) and problem (9a)–(9c) are with and without terminal state constraints, respectively.

Discretization. The discrete solutions of (9a)–(9c) and (9a)–(9d) will be sought. Thus, we should discretize the continuous time optimal control problem first. Without loss of generality, we can assume $t_0 = 0$, using time interval $\Delta t = t_f/N$. The continuous problem (9a)–(9c) is then discretized as

$$\min \sum_{i=0}^{N} J(x_i, u_i), \tag{10a}$$

$$x_{i+1} - x_i = \frac{t_f}{N} f(x_i, u_i, i); \quad x_0 = x_0, \tag{10b}$$

$$g(u_i) \le 0, \quad i = 0, \ldots, N-1, \tag{10c}$$

where $(\cdot)_i$ denotes $(\cdot)(i\,\Delta t)$; $J(x_i, u_i) = \frac{1}{2}(u_i^T R_i u_i + x_i^T Q_i x_i)\,\Delta t$, for $0 \le i < N-1$; $J(x_N, u_N) = x_N^T M x_N$, and $u_N = 0$; the function $f: \mathbb{R}^{m+p} \to \mathbb{R}^m$ in (10b) results from discretizing (9b), and different orders of the Runge–Kutta method employed for discretization will result in different fs. Furthermore, the discrete form of (9d) is

$$T(x_N) = 0. \tag{10d}$$

Direct application of the RQPD algorithm to (10a)–(10c). Clearly, (10a)–(10c) is a nonconvex separable optimization problem. We can solve it directly by the RQPD algorithm. Let

$$E_i(l) = x_{i+1}(l) - x_i(l) - \frac{t_f}{N} f(x_i(l), u_i(l), i), \quad 0 \le i \le N-1,$$

$$f_{x_i}(l) \equiv \left.\frac{\partial f}{\partial x}\right|_{(x_i(l), u_i(l), i)}, \quad f_{u_i}(l) \equiv \left.\frac{\partial f}{\partial u}\right|_{(x_i(l), u_i(l), i)},$$

$$g_{u_i}(l) \equiv \left.\frac{\partial g}{\partial u}\right|_{u_i(l)},$$

and let r_i^s and $\bar{q}_i^\kappa$ be the sth and κth diagonal elements of R_i and Q_i, respectively. We also let $\bar{q}_N^\kappa$ denote the κth diagonal element of M. Define that

$$q_i^\kappa = \begin{cases} \bar{q}_i^\kappa, & \text{if } \bar{q}_i^\kappa > 0, \\ \gamma, & \text{otherwise,} \end{cases} \quad i = 1, \ldots, N;$$

then (11a)–(11c) are quadratic approximations of (10a)–(10c) at $(x_i(l), u_i(l))$, $i = 1, \ldots, N-1$:

$$\min [Mx_N(l)]^T dx_N + dx_N^T M dx_N + \frac{t_f}{N} \sum_{i=0}^{N-1} [Q_i x_i(l)]^T dx_i + [R_i u_i(l)]^T du_i + dx_i^T Q_i dx_i + du_i^T R_i du_i, \tag{11a}$$

subject to:

$$E_i(l) + dx_{i+1} - dx_i - \frac{t_f}{N} f_{x_i}^T(l) dx_i - \frac{t_f}{N} f_{u_i}^T(l) du_i = 0; \quad dx_0 = 0, \tag{11b}$$

$$g(u_i(l)) + g_{u_i}^T(l) du_i \le 0, \quad i = 0, \ldots, N-1. \tag{11c}$$

Let $\Psi(\lambda, \mu)$ denote the dual function of (11a)–(11c) and $\Psi(j)$ denote $\Psi(\lambda, \mu)$ at the jth iteration. We may now state the RQPD algorithm for solving (10a)–(10c).

The computing formula included in this algorithm can be easily derived based on the RQPD algorithm; readers who are interested in the details are referred to Lin (1992a).

Algorithm II:

Step 0. Set the values of γ and $(x(0), u(0))$, and let $l = 0$. *Step* 1. Set the values of $\lambda(0)$ and $\mu(0)$ (≥ 0), and let $j = 0$. *Step* 2. Compute the following coefficients in parallel: $E_i(l)$, $f_{x_i}(l)$, $f_{u_i}(l)$, $g(u_i(l))$, $R_i u_i(l)$ and $Q_i x_i(l)$, where $i = 0, 1, \ldots, N-1$, and $Mx_N(l)$. *Step* 3. Compute in parallel

$$\hat{dx}_i^\kappa(j) = \begin{cases} -\frac{1}{q_i^\kappa}([Q_i x_i(l)]^\kappa + \lambda_{i-1}^\kappa(j) - \lambda_i^\kappa(j) \\ \quad -\frac{t_f}{N} f_{x_i}^{\kappa T}(l)\lambda_i(j)), \quad \text{if } i = 1, \ldots, N-1; \\ -\frac{1}{q_N^\kappa}([Mx_N(l)]^\kappa + \lambda_{N-1}^\kappa(j)), \quad \text{if } i = N; \end{cases} \tag{12a}$$

$$\kappa = 1, \ldots, m,$$

$$\hat{du}_i^s(j) = -\frac{1}{r_i^s}([R_i u_i(l)]^s - \frac{t_f}{N} f_{u_i}^{sT}(l)\lambda_i(j) + g_{u_i}^{sT}(l)\mu_i(j)), \tag{12b}$$

$$s = 1, \ldots, p, \; i = 0, \ldots, N-1.$$

Step 4. Compute in parallel

$$\nabla_{\lambda_i^\kappa}\Psi(j) = E_i^\kappa(l) + \hat{dx}_{i+1}^\kappa(j) - \hat{dx}_i^\kappa(j) - \frac{t_f}{N}[f_{x_i}^{\kappa T}(l)\hat{dx}_i(j) + f_{u_i}^{\kappa T}(l)\hat{du}_i(j)], \quad \kappa = 1, \ldots, m, \tag{13a}$$

$$\nabla_{\mu_i^s}\Psi(j) = g^s(u_i(l)) + g_{u_i}^{sT}(l)\hat{du}_i(j), \tag{13b}$$

$$s = 1, \ldots, q, \quad i = 0, \ldots, N-1.$$

Step 5. If $\max_i \{|\nabla_{\lambda_i}\Psi(j)|_\infty, |\nabla_{\mu_i}\Psi(j)|_\infty\} < \varepsilon_1$, go to Step 7; otherwise, go to Step 6. *Step* 6. Update in parallel

$$\lambda_i^\kappa(j+1) = \lambda_i^\kappa(j) + \bar{\beta}\nabla_{\lambda_i^\kappa}\Psi(j),$$

$$\mu_i^s(j+1) = \mu_i^s(j) + \bar{\beta}\nabla_{\mu_i^s}\Psi(j),$$

$$i = 0, \ldots, N-1, \quad \kappa = 1, \ldots, m, \quad s = 1, \ldots, q,$$

and return to Step 3, where the stepsize $\bar{\beta}$ is similarly defined as in (6) by replacing Φ with Ψ. *Step* 7. If $\max_i (|\hat{dx}_i|_\infty, |\hat{du}_i|_\infty) < \varepsilon_2$, stop and output optimal control solution $u_i(l)$, $i = 0, \ldots, N-1$; otherwise, go to Step 8. *Step* 8. Update in parallel

$$x_{i+1}^k(l+1) = x_{i+1}^k(l) + \bar{\alpha}\hat{dx}_{i+1}^k,$$

$$u_i^s(l+1) = u_i^s(l) + \bar{\alpha}\hat{du}_i^s, \tag{15}$$

$$i = 0, \ldots, N-1, \quad k = 1, \ldots, m \quad \text{and} \quad s = 1, \ldots, p,$$

and set $l = l + 1$; then return to Step 1, where the stepsize $\bar{\alpha}$ is similarly defined as in (3), that is, to minimize the absolute-value penalty function of (11a)–(11c) along the direction $(\hat{dx}, \hat{du})$.

(Note: $(\cdot)^{(\diamond)}$ denotes the $\diamond$th element of the vector $(\cdot)$ or the $\diamond$th row of the matrix $(\cdot)$.)

Corollary 2. Suppose (i) there exist scalars δ and η such that $0 < \delta < \max_{i,s} r_i^s < \eta$, and $0 < \delta < \max_{i,\kappa} q_i^\kappa < \eta$, (ii) the bounded constraints on each component of u are included in the constraints set $g(u) \le 0$, (iii) there exists a nonempty subset $\{u \mid g'(u) < 0\} \subset \{u \mid g(u) \le 0\}$, and each g_i' is convex. Then if the sequence $\{(x(l), u(l))\}$ generated by Algorithm II is bounded, this sequence will converge to a Kuhn–Tucker point of (10a)–(10c).

Remark. For most practical control systems, the control variables are bounded.

The above corollary follows easily from Corollary 1 if we show that conditions (ii) and (iii) of Corollary 1 are satisfied. From (iii) of Corollary 2, we see that for any u, there exists a du such that $g(u) + g_u^T(u)du < 0$ because of the convexity of

g_i'. For this du, there always exists a dx such that

$$E_i(l) + dx_{i+1} - dx_i - \frac{t_f}{N} f_{x_i}^T(l) dx_i - \frac{t_f}{N} f_{u_i}^T(l) du_i = 0, \quad i = 0, \ldots, N-1. \quad (16)$$

Therefore, there must exist a feasible solution of (11a)–(11c), and the constraint qualification of (11a)–(11c) is also satisfied; thus, condition (iii) of Corollary 1 is satisfied. From (ii) of Corollary 2, the control is bounded, thus the optimal value of the objective function of (11a)–(11c) must be bounded below due to the continuous differentiability of f. Furthermore, the Hessian matrix of (11a)–(11c) is positive definite, so the optimal solution of (11a)–(11c) must be unique, and the Lagrange multiplier associated with the optimal solution must be finite. Thus, condition (ii) of Corollary 1 is satisfied.

Algorithm for solving (10*a*)–(10*d*). Because of the terminal constraints (10d), there may not exist a dx such that

$$E_i(l) + dx_{i+1} - dx_i - \frac{t_f}{N} f_{x_i}^T(l) dx_i - \frac{t_f}{N} f_{u_i}(l) du_i = 0, \quad i = 0, \ldots, N-1,$$

$$T(x_N(l)) + T_{x_N}^T(l) dx_N = 0, \quad (17)$$

hold for any du satisfying $g(u) + g_u^T(u) du \le 0$. This implies that condition (ii) of Corollary 1 may not be satisfied. To cope with such a difficulty, we employ an augmented Lagrangian method (Bersekas, 1982) to handle the terminal constraint (10d). The augmented Lagrangian method applied to (10a)–(10d) is to solve

$$\max_{\tau} \left\{ \min_{x,u} J(x, u) + cT(x_N)^2 + \tau T(x_N) \mid x_{i+1} - x_i - \frac{t_f}{N} f(x_i, u_i, i) = 0, \; g(u_i) \le 0, \; i = 0, \ldots, N-1 \right\},$$

iteratively with the updating procedure for τ by $\tau(k+1) = \tau(k) + cT(\hat{x}_N)$, where $\hat{x}_N$ is the optimal x_N of the constrained minimization problem within the bracket for a given $\tau(k)$, and the penalty coefficient c is a large positive scalar. The constrained minimization problem within the bracket has almost the same form as the optimal control problem (10a)–(10c), except that the terminal state penalty terms may not be quadratic. However, it can be solved by Algorithm II by simply replacing the $\bar{q}_N^{\kappa}$, which is equal to the κth diagonal term of M, with the κth diagonal term of $cT(x_N)^2 + \tau T(x_N)$ and replacing the $Mx_N(l)$ in (12a) with

$$\left. \frac{\partial [cT(x_N)^2 + \tau T(x_N)]}{\partial x_N} \right|_{x_N(l)}.$$

Thus, we may combine the augmented Lagrangian method with Algorithm II for solving (10a)–(10d) to formulate Algorithm III.

Algorithm III.

Step 0. Set a positive value $c(0)$ with moderate magnitude and a value $\tau(0)$, and let $k = 0$. *Step* 1. Use Algorithm II to solve

$$\min_{x,u} J(x, u) + c(k) T(x_N)^2 + \tau(k) T(x_N)$$

$$x_{i+1} - x_i - \frac{t_f}{N} f(x_i, u_i, i) = 0, \quad (18)$$

$$g(u_i) \le 0, \quad i = 0, \ldots, N-1.$$

Step 2. Update $\tau(k+1) = \tau(k) + c(k) T(\hat{x}_N)$, where $\hat{x}_N$ is obtained from Step 1. *Step* 3. If $|T(\hat{x}_N)|_\infty < \varepsilon_3$ (a preselected accuracy) stop, and the solution obtained from Step 1 is the optimal solution of (10a)–(10d); otherwise, set $k = k + 1$, increase $c(k)$ by some formula, and return to Step 1.

For the augmented Lagrangian method, in addition to all the sufficient conditions of Corollary 2, a second order sufficiency condition (Bersekas, 1982) of (10a)–(10d) stipulating that the Hessian matrix at the strict local minimum is positive definite on its tangent plane is needed to ensure the convergence of Algorithm III. This second order sufficiency condition is difficult to check in advance unless the problem under consideration is linear quadratic. For example, for a problem with quadratic performance index ($R > 0$, $Q \ge 0$, $M \ge 0$) and convex inequality constraint function g, equation (10b) shows that x_i, $i = 1, \ldots, N$ can be viewed as a function, say $\Gamma_i(u, x_0)$, of $u(i)$, $i = 0, \ldots, N-1$ and the initial condition x_0. Substituting the x terms in the performance index and the terminal constraint (10d) by the function $\Gamma_i(u, x_0)$, $i = 1, \ldots, N$, we see that the Hessian matrix of the problem is positive definite if T and f are affine. However, if either f or T is not affine, the second order sufficiency condition cannot be checked unless the optimal solution and the corresponding Lagrange multiplier are obtained.

Implementation and modifications. Obviously, the computations of (12a), (12b), (13a), and (13b) in Steps 3 and 4 of Algorithm II constitute most of the computations required in Algorithms II and III. These calculations can be carried out independently for each i, j. Furthermore, for each time interval i, only the data in the same time interval are needed to calculate $d\hat{u}_i$ and $\nabla_{u_i}\Phi$ in (12b) and (13b). However, the λ_{i-1} and $d\hat{x}_{i+1}$ from adjacent time intervals are required for calculating $d\hat{x}_i$ and $\nabla_{\lambda_i}\Phi$ in (12a) and (13a). For such a highly parallel computing algorithm with an extremely simple and regular data flow pattern, a sequential computer is not the optimal means of implementation. However, a parallel processing computing system is not suitable either, because too much communication overhead may degrade the computational performance. Thus, special purpose data driven waveform VLSI array processors such as those described by Kung (1988) should be the best choice for implementing our algorithms. Data driven waveform VLSI array processors can avoid global synchronization, thus greatly reducing time delay. A simple asynchronous handshaking device (Kung, 1988) may be used to acknowledge data propagation between processing elements, and this device will induce only negligible time delay. However, for the sake of regularity of the VLSI array processors, some modifications of our algorithms are needed. First, the stepsizes $\bar{\alpha}, \bar{\beta}$ are set as constants. Second, we replace the convergence check in Step 5 of Algorithm II by checking whether the number of iterations j exceeds a certain limit $j_{\max}$. The selection of this number $j_{\max}$ depends on the number of time intervals, N, and the order of the system. Similarly, when applying Algorithm III, we may replace the convergence check in Step 7 of Algorithm II by checking whether the number of iterations $l \ge l_{\max}$ which is a certain limit. Such type of modifications and successful tests are described by Lin (1991, 1992b). A detailed approach to the VLSI array processor design needed to realize our algorithm is not provided here; however, a similar version has been developed by Lin (1992b).

Remarks. (a) Consider a case where the simple inequality constraint is defined as $\underline{u} \le u \le \bar{u}$, where $\underline{u}$ and $\bar{u}$ are the lower and upper bounds, respectively, of control variable u. We can separate the inequality constraint $g(u) \le 0$ into a simple inequality constraint $\underline{u} \le u \le \bar{u}$ and non-simple inequality constraint $g''(u) \le 0$, and convert $g''(u) \le 0$ by the equality constraint $g''(u) + z = 0$ and simple inequality constraint $z \ge 0$. Thus, in the dual method, we can easily circumvent the need for the Lagrange multiplier μ for inequality constraints by projecting u and z into the set $\{(u, z) \mid \underline{u} \le u \le \bar{u}, z \ge 0\}$. Using such modifications, $\bar{\alpha}$ should be set as 1. Lin (1992b) has tested that cases with constant stepsize $\bar{\alpha}$ and $\bar{\beta}$ favor such modifications. (b) In order to ensure that the sequence $\{(x(l), u(l))\}$ generated by our algorithm is bounded, the increment $\bar{\alpha} d\hat{x}$ and $\bar{\alpha} d\hat{u}$ in (15) should be small. However remark (a) indicates that $\bar{\alpha} = 1$. Thus, in order to achieve the small increment, we may

increase the value of q_i^κ and r_i^s in (12a) and (12b) for each i, κ and s. This can be accomplished by multiplying a large positive scaling factor S to the performance index. It is easily verified that positively scaling the performance index will not affect the optimal solution.

Computation time analysis. As may be expected that our algorithms have a rather slow convergence rate because of the linear convergence rate of the dual method, the linear convergence rate, at the worst, of the employed recursive quadratic programming method, and the superlinear convergence rate of the augmented Lagrangian method. However, since we intend to implement the algorithms by VLSI array processors, it is more instructive to estimate the time needed for the complete solution process using current VLSI technology. In making this estimation, first we notice that in Algorithm II, for sufficiently large $j_{\max}$, the computation time needed for Steps 1, 2, 7 and 8 is much less than that needed for Steps 3–6. Thus, we may define a basic iteration of our algorithms (Algorithms II and III) as one sweep of Steps 3–6; total computation time would then be approximately equal to the total number of basic iterations needed to complete the solution process multiplied by the computation time needed for one basic iteration. Lin (1992b) has analyzed that compared to the computation time, the communication overhead due to data propagation is negligible. Let $\mathcal{T}(\cdot)$ denote the number of multiplications $\otimes$ and additions $\oplus$ required to complete the computations for $(\cdot)$. Under a data driven waveform VLSI array processor environment, computations of the same algorithmic step for different time intervals can be carried out simultaneously. Thus, the computation time spent on Step 3 of Algorithm II is $\max_{i,k,s}\{\mathcal{T}(\hat{dx}_i^k), \mathcal{T}(\hat{du}_i^s)\}$, which equals $(m+q+4)\otimes + (m+q+2)\oplus$; and the computation time spent on Step 4 of Algorithm II is $\max_{i,k,s}\{\mathcal{T}(\nabla_{\lambda_i^k}\Phi(j)), \mathcal{T}(\nabla_{\mu_i^s}\Phi(j))\}$, which equals $(m+p+2)\otimes + (m+p+2)\oplus$. The modified Step 5 of Algorithm II requires only one comparison operation to check whether $j \geq j_{\max}$, and the processing time for the comparison in hardware is usually less than one $\oplus$ (Lin, 1992b). The time spent on Step 6 of Algorithm II for updating the variables is one $(\otimes + \oplus)$. Therefore, the computation time spent for one basic iteration is approximately equal to $(2m+q+7)\otimes + (2m+p+q+6)\oplus$, and the total computation time of the whole solution process can be estimated as

(total number of basic iterations)

$$\times [(2m+q+7)\otimes + (2m+p+q+6)\oplus]. \quad (19)$$

Example. Van der Pol oscillator.

$$\min \int_0^5 \tfrac{1}{2}(x_1^2 + x_2^2 + u^2)\, dx,$$

$$\begin{aligned} \dot{x}_1 &= (1-x_2^2)x_1 - x_2 + u; \quad x_1(0) = 0, \\ \dot{x}_2 &= x_1; \quad x_2(0) = 1, \\ -0.5 &\leq u \leq +0.5; \quad t \in [0, 5], \\ x_1(5) &= 0, \quad x_2(5) = 0. \end{aligned} \quad (20)$$

This example without control variable and terminal state constraints has been tested by Nedeljkovic (1981). We apply the second order Runge–Kutta method to discretize the state equations in (20) and use the following parameters for Algorithm II: a performance index scaling factor $S = 100$, $N = 30$, $j_{\max} = 40$, $l_{\max} = 40$, $\bar{\alpha} = 1$, $\bar{\beta} = 0.1$. We designate the initial value of c in Algorithm III to be 5, and the increased formula for c by $ck!$, where k is the iteration index of Algorithm III. With the above parameters, we apply Algorithm III to solve (20); the solution of the optimal state trajectory and optimal control are sketched in Figs 1 and 2, respectively. As shown in Table 1, the Euclidean norm of the terminal state $(x_1(5), x_2(5))$ approaches 0 as c increases. The accuracies of the solution are fairly satisfactory with $\epsilon_1 = 0.06$, $\epsilon_2 = 0.001$.

We will not compare the total iterations of our algorithm with the tested methods described by Nedeljkovic (1981),

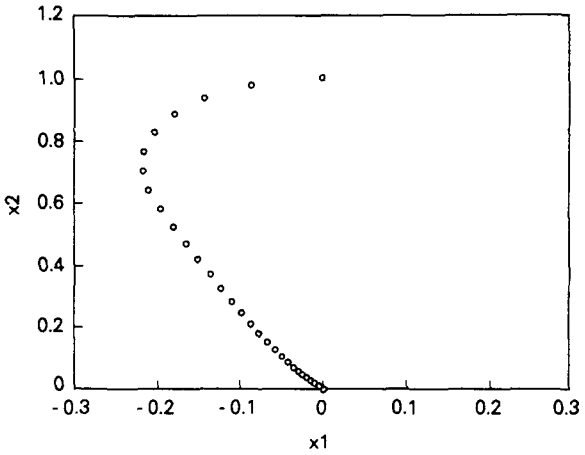

FIG. 1. The optimal state trajectory of the Van der Pol oscillator.

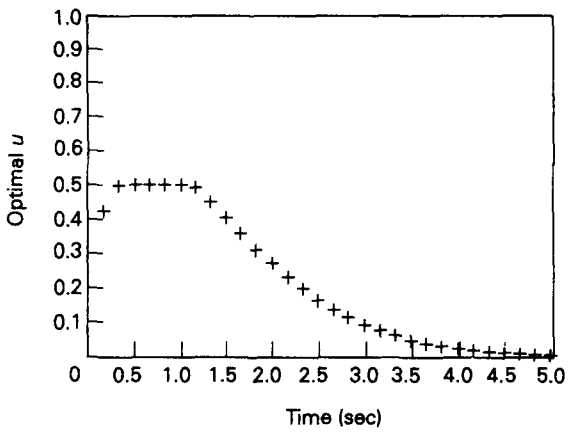

FIG. 2. The optimal control solution of the Van der Pol oscillator.

TABLE 1. THE VALUES OF THE TERMINAL STATES WITH RESPECT TO c

c	$x_1(5)$	$x_2(5)$
5	-7.3×10^{-3}	1.4×10^{-3}
10	-3.9×10^{-3}	2.2×10^{-3}
30	-1.9×10^{-3}	1.9×10^{-3}
120	-9.8×10^{-4}	9.2×10^{-4}
600	-5.2×10^{-4}	4.2×10^{-4}

because his methods deal with a simpler unconstrained case, and because our algorithm is basically designed to be implemented by hardware. Nevertheless, we have prepared a rough estimate of the execution time for our algorithm based on (19) and current VLSI technology. The total number of basic iterations used in this example is $5\times40\times40 = 8000$. The arithmetic operations needed for one basic iteration in this example are $(2\times2+1+7)\otimes + (2\times2+1+1+6) = 12\otimes + 12\oplus$. According to the technology developed by Sharma *et al.* (1989), the length of time needed to perform one 16×16-bit $\otimes$ and $\oplus$ are 6.75 nsec and $\leq$0.35 nsec, respectively. Thus, a rough estimation of the execution time for this example is less than 1 msec ($=10^{-3}$ sec).

4. *Conclusion*

This paper presents a complete decomposition algorithm for nonconvex separable optimization problems. The algorithm is especially suitable for solving constrained nonlinear fixed final-time optimal control problems with quadratic performance index. For such applications, implementation by VLSI array processors is recommended.

Although significant effort may be needed to realize such an implementation, considering its value for real-time processing control systems, this effort should be worthwhile.

References

Bazaraa, M. S. and C. M. Shetty (1979). *Nonlinear Programming*. John Wiley, New York.

Bersekas, D. P. (1979). Convexification procedures and decomposition methods for nonconvex optimization problems. *J. Opt. Theory Applic.*, **29**, 169–197.

Bersekas, D. P. (1982). *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York.

Han, S. P. (1976). Superlinearly convergent variable metric algorithms for general nonlinear programming problems. *Math. Prog.*, **11**, 263–282.

Han, S. P. (1977). A globally convergent method for nonlinear programming. *J. Opt. Theory Applic.*, **22**, 297–309.

Kung, S. Y. (1988). *VLSI Array Processors*. Prentice Hall, London.

Lin, S.-Y. (1991). A distributed state estimator for electric power systems. *IEEE PES* 1991 *Summer Meeting*, San Diego.

Lin, S.-Y. (1992a). A complete decomposition algorithm for large-scale nonconvex separable optimization problems and computations of optimal control. *NCTU Technical Report # CNTR-LIN-9201*.

Lin, S.-Y. (1992b). A hardware implementable two-level parallel computing algorithm for general minimum-time control. *IEEE Trans. Aut. Control*, **AC-37**, 589–603.

Luenberger, D. G. (1984). *Linear and Nonlinear Programming*, 2nd ed. Addison, MA.

Mukai, H. and E. Polak (1978). On the use of approximations in algorithms for optimization problems with equality and inequality constraints. *SIAM J. Numer. Anal.*, **15**, 674–693.

Nedeljkovic, N. B. (1981). New algorithms for unconstrained nonlinear optimal control problems. *IEEE Trans. Aut. Control*, **AC-26**, 868–884.

Sharma, R., A. Lopaz, J. Michejda, S. Hillenius, J. Andrews and A. Studwell (1989). A 6.75-ns 16×16-bit multiplier in single-level-metal CMOS technology. *IEEE J. Solid State Cric.*, **24**, 922–927.

Tanikawa, A. and H. Mukai (1985). A new technique for nonconvex primal-dual decomposition of a large-scale separable optimization problem. *IEEE Trans. Aut. Control*, **AC-30**, 133–143.

Tatjewski, P. (1989). New dual-type decomposition algorithm for nonconvex separable optimization problems. *Automatica*, **25**, 233–242.

Watanabe, N. Y. Nishimura and M. Matsubara (1978). Decomposition in large system optimization using the method of multipliers. *J. Opt. Theory Applic.*, **25**, 181–193.