INT. J. CONTROL, 1989, VOL. 49, NO. 2, 449–469

Quadratic incentive coordination for non-convex optimal control problems

Part 2. Parallel algorithms

MATT BROMBERG†, TSU-SHUAN CHANG† and PETER B. LUH‡

Quadratic incentive coordination is used to decompose optimal control problems into subproblems. A high-level problem is created to coordinate the subproblems. In the companion paper, it is shown that the high-level problem is equivalent to the original problem in the sense that there is a one-to-one correspondence between the local minima of the two problems, provided certain technical conditions are satisfied. This paper presents a gradient algorithm and Newton's algorithm for solving the high-level problem. The solution of the high-level problem then provides us with a solution for the original problem as well. It is shown that the gradient algorithm can be decoupled for use in parallel computation and that the algorithm converges. A numerical example is provided to demonstrate feasibility.

1. Introduction

As the scope and power of parallel processing machines increases, the desire for algorithms which utilize parallel processing techniques increases as well. Thus new methods for solving non-convex optimal control problems in a parallel processing environment are needed. It is desired to find methods which are sufficiently general to cover a wide class of problems. This paper addresses these issues by offering parallel algorithms which will solve a wide class of non-convex optimal control problems. It is a continuation of the companion paper which has addressed the more theoretical aspects of quadratic incentive control. The present work addresses numerical algorithms to support the theory.

In the companion paper (Bromberg *et al.* 1989), the following problem has been defined as the basic problem.

Basic problem: (P)

$$\min_{u} J(u), \quad J(u) = \sum_{j=0}^{M-1} g_j(x_j, u_j) \tag{1 a}$$

where

$$x_{j+1} = f_j(x_j, u_j), \quad j = 0, \ldots, M-1 \tag{1.1 b}$$

and x_0 is given.

In the present work numerical algorithms will be developed to solve this problem by decomposing it into several subproblems. The decomposition is achieved through

Received 4 November 1987. Revised 17 March 1988.

† Department of Electrical Engineering and Computer Science, University of California, Davis, CA 95616, U.S.A.

‡ Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06268, U.S.A.

0020-7179/89 $3.00

a method known as incentive coordination (Bromberg *et al.* 1989). A high-level problem has been defined by Bromberg *et al.* (1989). Its purpose is to coordinate the low-level subproblems by finding a set of parameters p_j^* which control the subproblems by changing the cost of an additive quadratic term. The p_j^* in turn are found as the solution of a high-level problem (P – *H*) defined below.

Incentive coordination: (P – H)

$$\min_{p} J^I(p), \quad J^I(p) = \sum_{j=0}^{M-1} g_j(x_j^*, u_j^*) \tag{1.2 a}$$

where

$$p \equiv (p_0, p_1, \ldots, p_{M-1}) \in P \tag{1.2 b}$$

and x_j^* and u_j^* are the optimal solutions for the low-level subproblems defined below.

Subproblem: (P – *j*)

$$\left.\begin{aligned} &\min_{u_j} J_j^I(u_j | x_j, p_j) \\ &J_j^I(u_j | x_j, p_j) \equiv g_j(x_j, u_j) + l_j(x_{j+1}) \end{aligned}\right\} \tag{1.3 a}$$

where

$$l_j(x_{j-1}) \equiv \tfrac{1}{2} Q x_{j-1}^t x_{j-1} + p_j x_{j-1} \tag{1.3 b}$$

and

$$x_{j-1} = f_j(x_j, u_j) \quad \text{and} \quad x_j \text{ is given} \tag{1.3 c}$$

The solution to (1.3 *a*) will be denoted by $u_j^* = u_j^*(x_j, p_j)$. If (1.3 *a*) are solved sequentially, x_j can be considered as a function of the previous p_i, $0 \leqslant i < j$. Therefore u_j can be written as a function of all of the p_j, and is expressed by $u_j = u_j^*(p)$. Another type of coordination which has been discussed is known as target coordination. Here the low-level problems are controlled by fixing their end states. It will be seen that this type of coordination is very useful in understanding the convergences of algorithms implementing quadratic incentive coordination.

Target coordination: (PT – *H*)

$$\left.\begin{aligned} &\min_{x} J^T(x) \\ &J^T(x) = \sum_{j=0}^{M-1} g_j(x_j^*, u_j) \equiv \sum_{j=0}^{M-1} g_j^*(x_j, x_{j+1}) \end{aligned}\right\} \tag{1.4}$$

where x_j^* and u_j^* are the optimal solutions for the low-level subproblems and $g_j^*(x_j, x_{j+1})$, denote the optimal cost of (PT – *j*) defined below.

Subproblem: (PT – *j*)

$$\min_{u_j} J_j^T(u_j | x_{j+1}) \tag{1.5 a}$$

where

$$J_j^I(u_j|x_j, x_{j+1}) = g_j(x_j, u_j) \tag{1.5 b}$$

$$= g_j(x_j, u_{Dj}(x_j, x_{j+1}, u_{I_j}), u_{I_j}) \equiv \hat{g}_j(x_j, x_{j+1}, u_{I_j}) \tag{1.5 c}$$

$$x_{j+1} = f_j(x_j, u_j) \quad \text{and} \quad \text{both } x_j \text{ and } x_{j+1} \text{ are given} \tag{1.5 d}$$

While it is desirable to obtain a global minimum for the problems above, in practice only local minima can be obtained. Therefore the minimizations above are assumed to be local minima obtained by some local minimum finder algorithm. This raises some theoretical difficulties which are addressed in the companion paper. Some of the assumptions, definitions and results are included for easy reference.

Definition 1: $r_j(p)$

Let $r_j(p)$ be a radius function chosen so that $J_j^I(u_j)$ has a global minimum at $u_j^*(p)$ in $B(u_j^*(p), r_j(p))$—the open ball with centre at $u_j^*(p)$ and radius $r_j(p)$. The ball in which this global minimum occurs will be referred to as $B_j(p)$.

Definition 2: $F(u)$ *conglomerate system dynamics*

Let $F(u)$ be the function which maps the controls u, to the conglomerate end states x, via the system dynamics. Thus $F(u)$ is defined recursively as follows:

$$F(u) \equiv (x_1, x_2, \ldots, x_M) \tag{1.6}$$

where $x_{j+1} = f_j(x_j, u_j)$ and x_0 is given.

Definition 3: $x^*(p)$ *incentive to state map*

$$x^*(p) \equiv F(u^*(p)) \tag{1.7}$$

Definition 4: C^n *solutions*

The basic problem has C^n solutions if and only if for any given $\bar{p}$ and $x \in F(U(\bar{p}))$,

$$g_j^*(x_j, x_{j+1}) = \min_{u_j \in S_j} g_j(x_j, u_j) \tag{1.8}$$

is C^n where S_j is defined by

$$S_j \equiv \{u_j : u_j \in U_j(\bar{p}) \text{ and } f_j(x_j, u_j) = x_{j+1}\} \tag{1.9}$$

Here $U_j(\bar{p})$ is an open subset of $B_j(p)$ containing $u_j^*(p)$, which is constructed in Lemma 2, which will be stated shortly, moreover, $U(\bar{p}) \equiv (U_0(\bar{p}), U_1(\bar{p}), \ldots, U_{M-1}(\bar{p}))$.

Assumption 1: *Smoothness*

Functions $g_j(x_j, u_j)$ and $f_j(x_j, u_j)$ are smooth in terms of x_j and u_j.

Assumption 2: *One-stage controllability*

For any given initial state x_j, one can find a u_j to drive x_{j+1} to any desired state. Also assume that the equation $f_j(x_j, u_j) = x_{j+1}$ can be solved implicitly for any given (x_j, x_{j+1}) to yield a smooth function for the dependent components of u_j. After a

possible reordering of the components of u_j, u_j can be written as

$$u_j \equiv (u_{D_j}(x_j, x_{j+1}, u_{I_j}), u_{I_j}) \tag{1.10 a}$$

and this yields

$$f_j(x_j, u_{D_j}(x_j, x_{j+1}, u_{I_j}), u_{I_j}) = x_{j+1} \tag{1.10 b}$$

Assumption 3: Compactness

Assume that $J(u)$ is compact below. One defines $J(u)$ as *compact below*, if and only if for any real number r, the set $\{u: J(u) \leqslant r\}$ is compact.

Assumption 4

The algorithm is called continuous if and only if $r_j(p)$ and $u_j^*(p)$ are continuous functions of p. The present work assumes the algorithm is continuous.

Based on these definitions and assumptions, the companion paper (Bromberg *et al.* 1989) has demonstrated the following useful results.

Lemma 1

If the algorithm is continuous as in Assumption 4, then for any p in (P), there exists an open neighbourhood Π_p such that p is in Π_p and open neighbourhoods $U_j(p)$ of $u_j^*(p)$ such that if $\tilde{p}$ is in Π_p then

$$u_j^*(\tilde{p}) = \arg \min_{u_j \in U_j(p)} J_j^I(x_j^*(\tilde{p}), u_j) \tag{1.11}$$

Also letting $U(p) \equiv (U_0(p), U_1(p), \ldots, U_{M-1}(p))$, one concludes $u^*(\Pi_p)$ is a subset of $U(p)$; and if T is an open set containing p, then Π_p can be chosen so that Π_p is a subset of T.

Theorem 1

Assuming a continuous algorithm, if the basic problem (P) has C^n solutions, $n \geqslant 2$, and $u^* = u^*(p^*)$, then $J(u^*(p)) = J^I(p)$ has a local minimum at p^* if and only if $J(u)$ has a local minimum at u^*.

Lemma 2

If the basic problem has C^n solutions, $n \geqslant 2$, then for any given $\tilde{p}$, there exists a one-to-one correspondence between p and x via C^{n-1} functions $p^*(x) = p$ defined on the set $x^*(\Pi_p)$ and $x^*(p) = x$ defined on Π_p. Furthermore $x^*(\Pi_p)$ is an open set.

Corollary 1: $x^*(p)$ is an open map

If the basic problem has at least C^2 solutions, then $x^*(p)$ is an open map and therefore the conclusions of Theorem 1 in the work by Bromberg *et al.* (1989) hold.

In addition to the previous assumptions, the following assumptions will be needed.

Assumption 5

Assume that the basic problem has at least C^3 solutions.

Assumption 6

It is assumed that $x^*(P)$ contains a closed (compact) ball B_x, which contains all local minima of interest.

$J^T(x)$ is defined on this ball, and by Assumptions 5 and 6, it will be a C^3 function of the conglomerate end states x. Also note by Assumption 5 and Lemma 2 that $p^*(x)$ and $x^*(p)$ are C^2 functions defined on B_x and P respectively.

2. Parallel algorithm

Assume the algorithm being used is continuous and has C^2 solutions. By Theorem 1 it will be possible to solve the basic problem (P), by solving the high-level problem (P − H). This is because a local minimum for one will correspond to a local minimum for the other. How to find p^*, a local minimum of $J^1(p)$, is now addressed, so that a local minimum $u_j^* \equiv u_j^*(p^*)$ of (P) can be obtained.

Recall the definition of compact below in Assumption 3. Here it is assumed $J(u)$ is compact below. It should then be clear that $J^T(x)$ is compact below. To see this, note that from Assumption 3, $C \equiv \{u : J(u) \leqslant r\}$ is a compact set. Now suppose that x is in the set $D \equiv \{x : J^T(x) \leqslant r\}$. This means that

$$\sum_{j=0}^{M-1} g_j^*(x_j, x_{j+1}) \leqslant r \tag{2.1}$$

which says that controls $\bar{u}$ can be found, which minimize (locally) $g_j(x_j, u_j)$ such that $f_j(x_j, u_j) = x_{j+1}$. So this yields $F(\bar{u}) = x$ and

$$\sum_{j=0}^{M-1} g_j(x_j, \bar{u}_j) = J(\bar{u}) \leqslant r \tag{2.2}$$

This means that $\bar{u} \in C$ and thus $x \in F(C)$. So that D is a subset of $F(C)$. But $F(\cdot)$ is continuous, thus $F(C)$ is compact. The $g_j^*(x_j, x_{j+1})$ are continuous by Definition 4, so that $J^T(x)$ is continuous which implies that D is closed. However, since closed subsets of compact sets are compact, one sees that D is compact and thus $J^T(x)$ is compact below. In a similar fashion, since p is a continuous function of x, one can see that $J^T(x^*(p)) = J(u^*(p)) \equiv J^1(p)$ is compact below in p.

For a sequence p^j, indicating the jth iterate in the numerical algorithms which follow, define the forward difference as $\Delta p^j \equiv p^{j+1} - p^j$. Define similar differences for ΔJ^{T_j} and Δx^j. Also define $J_x(x) \equiv \partial J^T(x)/\partial x$. If (1.3) are solved sequentially, the following theorem results.

Theorem 2

(*a*) For any p^0 in P, there exists an $\varepsilon > 0$ such that $\lim_j J_x(x^*(p^j)) = 0$, where p^j is defined recursively by

$$\Delta p^j = -\varepsilon \frac{\partial p^*(x^j)}{\partial x} J_x^T(x^*(p^j)) \tag{2.3}$$

where $x^j \equiv x^*(p^j)$.

(*b*) Also there exists k such that

$$|J_x(x^*(p^j))| < \left(\frac{k}{j+1}\right)^{1/2} \tag{2.4}$$

(c)

$$\lim_i x^*(p^j) \to Z \tag{2.5}$$

where $Z \equiv \{x : J_x(x) = 0\}$.

(d) The $x^*(p^j)$ lie in a compact set, and if a and b are limit points of $x^*(p^j)$ then $J^T(a) = J^T(b)$ and $J_x(a) = J_x(b) = 0$.

Proof

Let $C \equiv \{p : J^T(x(p)) \leqslant J^T(x(p^0))\}$. This will be compact since $J^T(\cdot)$ is compact below. Thus there exists a compact ball B such that C is a subset of B. By the first-order Taylor theorem with the Lagrange remainder, and by noting that $J^T(x(p)) \equiv J^T(p)$,

$$\Delta J^{Tj} \equiv J^T(x(p^{j+1})) - J^T(x(p^j)) = \frac{\partial J^T(p^j)}{\partial p} \cdot \Delta p^j + \frac{1}{2}\left(\Delta p^j \frac{\partial^2 J^T}{\partial p^2}(s) \Delta p^j\right) \tag{2.6}$$

where s lies on the line joining p^j and p^{j+1}.

By Definition 4 and Lemma 3, $\partial^2 J^T / \partial p^2(x^*(p))$ is a continuous matrix function of p. The matrix sup norm $|\partial^2 J^T/\partial p^2|$ will then be a continuous function of p into $\mathbb{R}$. Since B is compact and continuous functions of compact sets are compact, $|\partial^2 J^T/\partial p^2|$ assumes a maximum value in $\mathbb{R}$; call it M. Similarly in B, $|\partial J^T/\partial p|$ assumes a max called G. Also the matrix sup norm $|\partial p^*(x)/\partial x|$ will achieve a maximum N on B.

Now choose an $\varepsilon \leqslant 1/N^2M$. By Theorem 2, part (a), this means that

$$|\Delta p^j| \leqslant \frac{1}{N^2 M}\left|\frac{\partial p^*(x)}{\partial x}\right| \cdot |J_x^T| \leqslant \frac{G}{NM} \tag{2.7}$$

Let B' be the closed ball with the same centre as B, but with a radius G/NM larger, so that $r' = r + (G/NM)$. B' is still compact and the maxima of $|\partial^2 J^T/\partial p^2|$, $|\partial J^T/\partial p|$ and $|\partial p^*(x)/\partial x|$ on B' will be M', G' and N', respectively. Note that $M < M'$, $G \leqslant G'$ and $N \leqslant N'$. Choose $\varepsilon = 1/NM \leqslant 1/NM$ as desired.

If $p^j \in C$ which is a subset of B, and $p^{j+1} \in B'$ by (2.3), the choice of ε and by (2.6) it follows that

$$\Delta J^{Tj} \leqslant -\frac{1}{N^2 M'}\frac{\partial J^T}{\partial p}(x^*(p^j))\frac{\partial p^*(x^j)}{\partial x}\frac{\partial J^T(x^j)^t}{\partial x} + \frac{1}{N^4 M'^2}\frac{1}{2}\frac{\partial J^T(x^j)}{\partial x}\frac{\partial p^*(x^j)^t}{\partial x}\frac{\partial^2 J^T}{\partial p^2}(s)\frac{\partial p^*(x^j)}{\partial x}\frac{\partial J^T(x^j)}{\partial x} \tag{2.8}$$

Now in the first term,

$$\frac{\partial J^T}{\partial p}\frac{\partial p^*(x^j)}{\partial x} = \frac{\partial J^T(x^j)}{\partial x}$$

and the norm of the second term on the right-hand side of (2.8) above is less than or equal to

$$\frac{1}{2}\frac{1}{N^4 M'^2}\left|\frac{\partial J^T(x)}{\partial x}\right|^2 N^2 M'$$

Since all terms are evaulated at p^j which is in B except $(\partial^2 J^1/\partial p^2)$, which is evaluated at s in B', it follows from (2.8) that

$$\Delta J^{T'} \leqslant \frac{-|J_x(x^j)|^2}{2N^2M'} \tag{2.9}$$

where $x^j \equiv x^*(p^j)$. s is in B' because s is between p^j and p^{j+1} which are both elements of the convex ball B'. However, this implies $\Delta J^{T'} \leqslant 0$, so that $J^T(x^*(p^{j+1})) \leqslant J^T(x^*(p^j))$. This means that $p^{j+1} \in C$, by the definition of C. Thus since $p^0 \in C$ for the choice of $\varepsilon = 1/NM'$, and if $p^j \in C$ then $p^{j+1} \in C$, one concludes by induction $p^j \in C$ for all j.

Now sum (2.9) from 0 to $n-1$. This yields

$$\sum_{j=0}^{n-1} \Delta J^{T'} = J^T(x^n) - J^T(x^0) \leqslant \frac{-1}{2N^2M'} \sum_{j=0}^{n-1} |J_x(x^j)|^2 \tag{2.10}$$

This implies

$$J^T(x^n) \leqslant J^T(x^0) - \frac{1}{2N^2M'} \sum_{j=0}^{n-1} |J_x(x^j)|^2 \tag{2.11}$$

One sees at once $J^T(x^n)$ is decreasing, and since $J^T(x)$ is bounded below, it must follow that

$$\sum_{j=0}^{\infty} |J_x(x^j)|^2 < \infty \tag{2.12}$$

This implies

$$\lim_{j\to\infty} |J_x(x^{ij})|^2 = 0 \tag{2.13}$$

Since $\sum_{j=0}^{\infty} k/(j+1)$ diverges, one can find a k such that $|J_x(x^j)|^2 < k/(j+1)$. This yields,

$$|J_x(x^j)| < \left(\frac{k}{j+1}\right)^{1/2} \tag{2.14}$$

and (b) is proved.

Now $x^j \in x^*(C)$ is a compact set, but suppose x^j does not converge to

$$Z = \{x : J_x(x) = 0\}$$

then we could find subsequence x^{j_k} such that

$$|x^{j_k} - z| > \varepsilon \tag{2.15}$$

for all z in Z and for some $\varepsilon > 0$. However, since $x^*(C)$ is compact there is a convergent subsequence $x^{j_{k_n}}$ which converges to x^*. However, by (2.13) $\lim_{n\to\infty} J_x(x^{j_{k_n}}) = 0$. Thus since $J_x(\cdot)$ is continuous, $J_x\left(\lim_{n\to\infty} x^{j_{k_n}}\right) = 0$, so that $J_x(x^*) = 0$. This means that $x^* \in Z$ which contradicts (2.15). Thus (c) has been shown, $x^j \to Z$, by contradiction.

For part (d), suppose a and b are limit points of x^j. It is known that $x^j \in x^*(C)$ a compact set, so a limit point exists. Let $x^m \to a$ and $x^n \to b$. From the above result part (c) and the fact that $J_x(\cdot)$ is continuous, one immediately has $J_x(a) = J_x(b) = 0$. Now suppose $J^T(a) > J^T(b)$. Since $J^T(\cdot)$ is non-increasing on x^j and $J^T(\cdot)$ is continuous,

one must be able to find an n^* such that $J^T(a) > J^T(x^{n^*}) \geq J^T(b)$. However, since $x^m \to a$, one can find an $m^* > n^*$ such that $J^T(x^{m^*}) \geq J^T(a) > J^T(x^{n^*})$. This contradicts the need for $J^T(x^{m^*}) \geq J^T(x^{n^*})$, since $J^T(\cdot)$ is non-increasing on x^j. Thus by contradiction $J^T(a) = J^T(b)$—the $J^T(b) > J^T(a)$ case is entirely analogous. It is important to note that by (2.6), one could prove this theorem for any

$$0 < \varepsilon \leq \frac{1}{N^4 M^{12}} \tag{2.16}$$

□

To obtain parallel algorithms, note that the only linkage between subproblems is the final or initial states. Chang *et al.* (1987) have used a prediction method to decouple the subproblems. In the present work, a parallel algorithm is developed by relating the incentive coordination to the target coordination since the latter is completely decoupled. It is desired to solve all the subproblems $(P-j)$, independently for a given value of p. Let y denote the optimal final states of all subproblems for a given p and initial state x. One can define a map $H(x, p):(X, P) \to X$ by

$$y \equiv H(x, p) \equiv (x_1^*(x_0, p_0), x_2^*(x_1, p_1), \ldots, x_M^*(x_{M-1}, p_{M-1})) \tag{2.17}$$

Recall that

$$J^T(x) \equiv \sum_0^{M-1} g_j^*(x_j, x_{j+1}) \tag{2.18}$$

and define the state decoupled gradient as

$$J_x(x, y) \equiv \left(\frac{\partial g_0^*}{\partial x_1}\bigg|_{x_1 = y_1} + \frac{\partial g_1^*}{\partial x_1}\bigg|_{x_2 = y_2}, \frac{\partial g_1^*}{\partial x_2}\bigg|_{x_2 = y_2} + \frac{\partial g_2^*}{\partial x_2}\bigg|_{x_3 = y_3}, \ldots, \frac{\partial g_{M-1}^*}{\partial x_M}\bigg|_{x_M = y_M} \right) \tag{2.19}$$

Note that if $y = x$, then $J_x(x, x) = \partial J^T(x)/\partial x$. Here $J_x(x, y)$ is an approximation to the gradient when y is close to x. Here one stage (conglomerate) controllability is assumed, so that we can consider the end states, x_j as independent. Bromberg *et al.* (1989) have shown that p_j depends only on x_j and x_{j+1}. The relationship is given by

$$p_j = p_j^*(x_j, x_{j+1}) = \frac{\partial g_j^*(x_j, x_{j+1})}{\partial x_{j+1}} - Q x_{j+1} \tag{2.20a}$$

Now define the matrix

$$\left[\frac{\partial p}{\partial x}\right]_{ij}(x, y) \equiv \frac{\partial p_i}{\partial x_j}\bigg|_{x_{j+1} = y_{j+1}} \tag{2.20b}$$

It is desired to express the state decoupled gradient as a function of (x, p). Therefore using $H(x, p)$ define,

$$J_x(x, p) \equiv J_x(x, H(x, p)) \tag{2.21}$$

where y is related to $H(\cdot)$ in (2.17) with $y_0 = x_0$ by definition. Also define,

$$G(x, p) \equiv \frac{\partial p}{\partial x}(x, H(x, p)) \cdot J_x(x, p)^t \tag{2.22}$$

It is important to notice that with the above definitions

$$J_x(x, p^*(x)) = \frac{\partial J^T(x)}{\partial x} \tag{2.23}$$

and

$$G(x^*(p), p) = \frac{\partial p^*(x)}{\partial x} \frac{\partial J^T(x^*(p))'}{\partial x} \tag{2.24}$$

To be consistent with the notation adopted in Theorem 2 and because of (2.23), the coupled gradient, $\partial J^T(x)/\partial x$ will be often referred to as simply $J_x(x)$.

The following theorem allows us to iteratively solve the $(P-j)$ simultaneously by using linear approximations of what the update of p and x should be based on the gradient algorithm in Theorem 2.

Lemma 3

For any $\mu > 0$, $N > 0$, there exists a step size $\varepsilon > 0$ in Theorem 2 such that

$$|p^j - \hat{p}^j| < \mu \quad \text{and} \quad |x^j - \hat{x}^j| < \mu$$

for all $j \leqslant N$, where $\hat{x}_j$ and $\hat{p}_j$ are defined as in Theorem 2.

$$\Delta \hat{p}^j = -\varepsilon \frac{\partial p^*(\hat{x}^j)}{\partial x} J_x(\hat{x}^j) \tag{2.25}$$

such that $\hat{x}^j = x^*(\hat{p}^j)$, $x^0 \in C$; and x^j and p^j are determined by

$$\Delta p^j = -\varepsilon G(x^j, p^j), \quad p^0 = \hat{p}^0 \tag{2.26}$$

$$\Delta x^j = -\varepsilon J_x(x^j, p^j), \quad x^0 = \hat{x}^0 \tag{2.27}$$

Also the above can be done for fixed ε such that $0 < \varepsilon < \alpha$, for any $\alpha > 0$. (ε is dependent on α.)

Proof

Note that Lemma 3, (2.25) above can be written as

$$\Delta \hat{p}^j = -\varepsilon G(\hat{x}^j, \hat{p}^j) \tag{2.28}$$

by (2.24). By Lemma 2, $x^*(p)$ is a C^2 function of p, and thus by the first-order Taylor theorem with remainder, we have

$$\hat{x}_k^{j+1} = \hat{x}_k^j + \frac{\partial x^*(p)}{\partial p} \Delta \hat{p}^j + \frac{1}{2} \Delta \hat{p}^{jT} \frac{\partial^2 x_k^*(p^{k^*})}{\partial p^2} \Delta \hat{p}^j \tag{2.29}$$

Here p_k^* is some point on the line between $\hat{p}^j$ and $\hat{p}^{j+1}$, and the subscript k indicates the kth coordinate of $x^*(p)$. Assume that $0 < \varepsilon \leqslant \alpha$. In Theorem 2 increase the radius of the closed ball B by μ and call this B_μ; $x^*(B_\mu)$ is compact and is contained in a compact ball C_μ. The cartesian product $C_\mu \times B_\mu$ is a compact set and $|G(x, p)|$ and $|J_x(x, p)|$ achieve maxima G and K, respectively, in these compact sets. Let B'_μ be the closed ball B_μ with radius increased by αG, and C'_μ be the closed ball C_μ increased by αK. By (2.26) and (2.27) if $p^j \in B_\mu$ and $x^j \in C_\mu$, then $p^{j+1} \in B'_\mu$ and $x^{j+1} \in C'_\mu$. On B'_μ or C'_μ or $C'_\mu \times B'_\mu$ the functions $G(\cdot)$, $J_x(\cdot)$, $x^*(p)$ and $p^*(x)$, have Lipschitz

constants—they are all C^1 and defined on compact convex sets—and there exists M such that

$$\left| J_x \frac{\partial p^*(x)}{\partial x^l} \frac{\partial^2 x_k^*(p)}{\partial p^2} \frac{\partial p^*(x)}{\partial x} J_x^l \right| < M \tag{2.30}$$

for all k on B_n^l. The Lipschitz constants for a given function $F(x, y)$ will have the notation

$$\max \left| \frac{\partial F}{\partial x} \right| \equiv |F|_x, \quad \max \left| \frac{\partial F}{\partial y} \right| \equiv |F|_y$$

and so on. Inserting (2.25) into (2.29) yields

$$\Delta \hat{x}_k^l = -\varepsilon \frac{\partial x_k^*}{\partial p} \frac{\partial p^*(\hat{x}^j)}{\partial x} J_x(\hat{x}^j) + \frac{\varepsilon^2}{2} J_x \frac{\partial p^*(\hat{x}^j)}{\partial x^l} \frac{\partial^2 x_k^*(p^*)}{\partial p^2} \frac{\partial p^*(\hat{x}^j)}{\partial x} J_x^l \tag{2.31}$$

Simplifying one has

$$\Delta \hat{x}^{j_k} = -\varepsilon (J_x(\hat{x}^j))_k + \frac{\varepsilon^2}{2} M_k \tag{2.32}$$

where M_k is the coefficient of ε^2 2 in (2.31). Define

$$d_j^x \equiv |x^j - \hat{x}^j|, \quad d_j^p \equiv |p^j - \hat{p}^j|, \quad d_j^{x_k} \equiv |x_k^j - \hat{x}_k^j| \tag{2.33}$$

and $d_j \equiv d_j^x + d_j^p$. Subtracting (2.32) from the kth component of the vector equation (2.27) yields

$$\Delta x^j - \Delta \hat{x}^{j_k} = \varepsilon (J_x(\hat{x}^j, \hat{p}^j) - J_x(x^j, p^j))_k - \frac{\varepsilon^2}{2} M_k \tag{2.34}$$

Writing this as the vector equation

$$x^{j+1} - \hat{x}^{j+1} = x^j - \hat{x}^j + \varepsilon (J_x(\hat{x}^j, \hat{p}^j) - J_x(x^j, p^j)) - \frac{\varepsilon^2}{2} \vec{M} \tag{2.35}$$

where the kth entry of $\vec{M}$ is M_k. Taking the norm of both sides of (2.35) using the triangle inequality, one can write,

$$d_{j+1}^x \leqslant d_j^x + \varepsilon |J_x(\hat{x}^j, \hat{p}^j) - J_x(x^j, p^j)| + \frac{\varepsilon^2}{2} M(n)^{1/2} \tag{2.36}$$

Here by (2.30) $|\vec{M}| \leqslant M(n)^{1/2}$ where n is the dimension of X, the state space. Subtracting (2.28) from (2.26), rearranging terms similar to (2.35), taking the norm of both sides, and applying the triangle inequality yields

$$d_{j+1}^p \leqslant d_j^p + \varepsilon |G(\hat{x}^j, \hat{p}^j) - G(x^j, p^j)| \tag{2.37}$$

Now, employing the Lipschitz constants and using the triangle inequality again on (2.36) and (2.37) we get

$$d_{j+1}^x \leqslant d_j^x + \varepsilon (|J_x|_x d_j^x + |J_x|_p d_j^p) + \frac{\varepsilon^2}{2} M(n)^{1/2} \tag{2.38}$$

and

$$d_{j+1}^p \leqslant d_j^p + \varepsilon (|G|_x d_j^x + |G|_p d_j^p) \tag{2.39}$$

Noting $d_j^x \leqslant d_j$ and $d_j^p \leqslant d_j$ (inside the parentheses), add (2.38) and (2.39) to yield

$$d_{j+1} \leqslant d_j + \varepsilon(|J_x|_x + |J_x|_p + |G|_x + |G|_p)\, d_j + \frac{\varepsilon^2}{2} M(n)^{1/2} \tag{2.40}$$

Let $L \equiv |J_x| + |J_x|_p + |G|_x + |G|_p$, the sum of the Lipschitz constants. So now (2.40) can be written in the form.

$$d_{j+1} \leqslant d_j(1 + \varepsilon L) + \frac{\varepsilon^2}{2} M(n)^{1/2} \tag{2.41}$$

Suppose a sequence c_k was found, such that

$$c_{k+1} = c_k(1 + \varepsilon L) + \frac{\varepsilon^2}{2} M(n)^{1/2} \tag{2.42}$$

and $c_0 = d_0 = 0$. Now this yields $d_0 \leqslant c_0$ and suppose $d_k \leqslant c_k$, then

$$d_{k+1} \leqslant d_k(1 + \varepsilon L) + \frac{\varepsilon^2}{2} M(n)^{1/2} \leqslant c_{k+1} \tag{2.43}$$

Thus by induction $c_k \geqslant d_k$ for all k. Equation (2.42) however, can be solved, say via Z-transform methods. The solution is

$$c_k = \frac{\varepsilon M(n)^{1/2}}{2L}((1 + \varepsilon L)^k - 1) \tag{2.44}$$

Now choose ε such that

$$0 < \varepsilon((1 + \varepsilon L)^N - 1) < \min\left(\alpha, \frac{2L\mu}{M(n)^{1/2}}\right) \tag{2.45}$$

This can always be done, in fact one can let

$$\varepsilon = \frac{\min\left(\alpha, \dfrac{2L\mu}{M(n)^{1/2}}\right)}{(1 + \alpha L)^N - 1} \tag{2.46}$$

With this choice, $d_k \leqslant c_k \leqslant c_N \leqslant \mu$ and $\varepsilon < \alpha$ as required. This yields

$$|\bar{x}^k - x^k| < \mu \quad \text{and} \quad |\bar{p}^k - p^k| < \mu \tag{2.47}$$

for $k \leqslant N$, since $d_j^x \leqslant d_k$ and $d_k^p \leqslant d_k$. This implies also that $x^k \in C_\mu$ and $p^k \in B_\mu$ for $k \leqslant N$. □

With Lemma 3 it is now possible to decouple the gradient algorithm in Theorem 2 and prove its convergence by noting that the decoupled iterates can be made arbitrarily close to the coupled iterates, which have known convergence properties. This is achieved by choosing a sufficiently small step size. By using the difference equations in (2.25) and (2.26), one no longer has to calculate the initial states for the subproblems sequentially by waiting for the solution of the previous subproblem. In this way the subproblems can be solved in parallel, with the high-level problem coordinating the updating of the gradient information at each iteration. Further parallelization can be achieved by noting that the gradient information needed for

problem $(P-j)$ depends only on its nearest neighbours, problems $(P-j-1)$ and $(P-j+1)$. The proof of the convergence of this decoupling scheme follows.

Theorem 3

For any $\delta > 0$, there exists $\varepsilon > 0$ and $N > 0$ such that $|J_x(x^*(p^N))| < \delta$ and $|J_x(x^N)| < \delta$ where for $j \leqslant N$, x^j and p^j are determined by

$$\Delta p^j = -\varepsilon G(x^j, p^j), \quad p^0 = \hat{p}^0 \tag{2.48}$$

$$\Delta x^j = -\varepsilon J_x(x^j, p^j), \quad x^0 = \hat{x}^0 = x^*(\hat{p}^0) \tag{2.49}$$

where $\hat{x}^j \equiv x^*(\hat{p}^j)$.

Proof

Lemma 3 will be applied to the system

$$\Delta \hat{p}^j = -\varepsilon \frac{\partial p^*(\hat{x}^j)}{\partial x} J_x'(x^*(\hat{p}^j)) \tag{2.50}$$

In Lemma 3 choose $\alpha = 1/N^4 M'^2$ as in Theorem 2, (2.16). Thus ε will be less than α by Lemma 3. By Theorem 2 then, one can choose an N such that $|J_x(\hat{x}^N)| < \delta/2$. Since $J_x(x^*(p))$ is continuous, choose a μ such that

$$|\hat{p}^N - p^N| < \mu \rightarrow |J_x(x^*(\hat{p}^N)) - J_x(x^*(p^N))| < \frac{\delta}{2} \tag{2.51}$$

and

$$|\hat{x}^N - x^N| < \mu \rightarrow |J_x(\hat{x}^N) - J_x(x^N)| < \frac{\delta}{2} \tag{2.52}$$

By Lemma 3 one can find a step size $\delta < \alpha$ guaranteeing both (2.51) and (2.52). Thus by the triangle inequality

$$\begin{aligned} |J_x(x^*(p^N))| &\leqslant |J_x(x^*(\hat{p}^N))| + |J_x(x^*(p^N)) - J_x(x^*(\hat{p}^N))| \\ &\leqslant \frac{\delta}{2} + \frac{\delta}{2} = \delta \end{aligned} \tag{2.53}$$

and

$$|J_x(x^N)| \leqslant |J_x(\hat{x}^N)| + |J_x(x^N) - J_x(\hat{x}^N)| \leqslant \frac{\delta}{2} + \frac{\delta}{2} = \delta \tag{2.54}$$

as desired. □

How to implement the algorithm suggested by Theorems 2 and 3 can now be considered. Note that in these theorems an extremum of J is actually being found as a function of x. In Theorem 1, given by Bromberg *et al.* (1989), however, one is only guaranteed a local minimum for $J(u)$ at a local minimum of $J(u^*(p))$. Since x and p are in a one-to-one correspondence via C^2 functions, however, a local minimum at p^* will be a local minimum in x at $x^* = x^*(p^*)$. One proceeds with the algorithm as follows. Construct a $\hat{p}^0$, $\hat{x}^0$ pair by solving the following problem.

Problem: (P $-j$)

$$\operatorname{lmin}_{u_j} J_j^l, \quad \text{where } J_j^l \equiv g_j(x_j, u_j) + p_j x_{j+1} + \tfrac{1}{2} Q x_{j+1}^2 \tag{2.55}$$

with $x_{j+1} = f_j(x_j, u_j)$ and 'lmin' refers to the local minimum.

Step 1

Solve (P $-j$) sequentially at first. Thus use p_0 to solve (P $-$ 0) which will determine x_1, which can be used with p_1 to solve (P $-$ 1) and so on until (P $-$ ($M-1$)) is solved.

Step 2

Choose an ε as in Theorem 3 to bring the Nth iterate of the decoupled algorithm in Theorem 3 δ close to an extreme point of J, as guaranteed by Theorem 3. In practice, one would continue this process until $J(x, p)$ stops decreasing; $J(x, p)$ is an estimate of the cost defined by

$$J(x, p) \equiv \sum_{j=0}^{M-1} g_j(x_j, u_j) \tag{2.56}$$

where the x_j are determined by (2.49) and thus only approximately obey the system dynamics. The u_j are determined by solving (2.55). Then consider p^N as p^0 in Step 1 above and resolve the problem sequentially, until one is as close to an extremum as one desires.

In this way one is guaranteed to get arbitrarily close to an extremum of J by Theorems 2 and 3.

The following simple example illustrates the feasibility of the above method. For problem (P)

$$J(u) \equiv \sum_{j=0}^{j=3} \left(\frac{1}{2} |u_j|^2 + \exp |x_j|^2 \left(2 + \sin \left(\sum_{i=0}^{2} u_{ji} \right) \right) \right) \tag{2.57}$$

is used and the system dynamics is

$$x_{j+1} = u_j - x_j \tag{2.58}$$

Here u_{ji} is the ith component of u_j. Also dim $u_j = 3$ and dim x_j = dim $p_j = 2$. In the two trials which follow, the step size was chosen to be $\varepsilon = 0{\cdot}05$. The low-level problems were solved using the DDP method (Yakowitz and Rutherford 1984), where all the u_j were initialized to the zero vector. The initial x vector, x_0, was a 2×1 vector whose entries all had the value 1·0. The initial values for p are shown in Table 1.

Tables 2 and 3 show the convergence of both the sequential algorithm and the parallel or decoupled algorithm. Since the decoupled algorithm switches back to the sequential algorithm when needed, an important parameter for determining the effectiveness of the parallel algorithm is the percent of actual decrease of the cost done

p_0	p_1	p_2	p_3
−50·0	10·0	−10·0	10·0
50·0	−10·0	10·0	−10·0

Table 1. Initial p.

Iterate	Cost	u_0	u_1	u_2	u_3
1	2·600172E + 01	1·902000E + 00	6·647756E − 01	−5·116095E − 02	−2·835344E − 02
10	1·529603E + 01	4·674931E − 01	−6·389089E − 01	−2·005053E − 01	−4·302932E − 01
20	1·525929E + 01	4·332305E − 01	−6·040295E − 01	−1·736391E − 01	−4·622791E − 01
80	1·524492E + 01	4·279819E − 01	−5·934228E − 01	−1·869863E − 01	−4·063720E − 01
116	1·524475E + 01	4·278151E − 01	−5·927052E − 01	−1·892551E − 01	−3·989678E − 01

Table 2. Sequential algorithm.

u_0	u_1	u_2	u_3
4·278E − 01	−5·927E − 01	−1·893E − 01	−3·990E − 01
4·277E − 01	−5·921E − 01	−1·911E − 01	−3·928E − 01
−2·133E + 00	−2·538E − 01	−5·765E − 01	−3·960E − 01

Table 3. Sequential algorithm: optimal control.

while solving the problem in parallel. This figure (percent decoupled) along with the norm of the gradient $|J_x(x)|$ (gradient size) at the terminus of the algorithms is provided. The stopping criteria was $|\Delta J^j| < 10^{-6}$. The values of the control listed here are the first component of the u_j vector for each stage, with the final iterate listed in the last column. The entire vector solution is shown for the optimal control and optimal trajectories.

Although both algorithms had about the same number of iterations, the sequential algorithm converted to a point closer to the optimal solution. The parallel algorithm however, was almost 100 % decoupled. However, one must remember that this is also offset by the need for smaller epsilon in order to achieve convergence for the parallel algorithm.

x_0	x_1	x_2	x_3
1·000E + 00	−5·722E − 01	−2·052E − 02	−1·687E − 01
1·000E + 00	−5·723E − 01	−1·977E − 02	−1·714E − 01

Optimal cost	Gradient size	Percent decoupled
15·244747	4·52408E − 03	0·000000

Table 4. Sequential algorithm: optimal trajectory.

Iterate	Cost	u_0	u_1	u_2	u_3
1	2·600172E + 01	1·902000E + 00	6·647756E − 01	−5·116095E − 02	−2·835344E − 02
10	1·527228E + 01	4·211970E − 01	−6·167420E − 01	−2·341634E − 01	−4·121033E − 01
20	1·526684E + 01	3·833797E − 01	−5·949022E − 01	−1·931717E − 01	−4·561516E − 01
70	1·526379E + 01	3·782672E − 01	−5·852563E − 01	−2·000384E − 01	−4·099358E − 01
113	1·525866E + 01	3·780903E − 01	−6·442229E − 01	−2·109204E − 01	−4·275023E − 01

Table 5. Parallel algorithm.

u_0	u_1	u_2	u_3
3·781E − 01	−6·442E − 01	−2·109E − 01	−4·275E − 01
4·455E − 01	−5·770E − 01	−2·143E − 01	−4·198E − 01
−2·105E + 00	−2·330E − 01	−5·563E − 01	−3·711E − 01

Table 6. Parallel algorithm: optimal control.

x_0	x_1	x_2	x_3
1·000E + 00	−6·219E − 01	−2·231E − 02	−1·886E − 01
1·000E + 00	−5·545E − 01	−2·247E − 02	−1·918E − 01

Optimal cost	Gradient size	Percent decoupled
15·258655	5·034617E − 01	99·115044

Table 7. Parallel algorithm: optimal trajectory.

3. Explicit formulae for numerical calculations

In performing calculations for the gradient methods considered above, it remains to determine explicit formulae for $G(x, p)$ and $J_x(x, p)$. To do this one notes the following. If the state x is given, then in order to obtain $y_{j+1} = x^*_{j-1}(x_j, p_j)$ one merely solves subproblem $(P - j)$. The presence of a large Q makes the necessary conditions sufficient as well for a local minimum thus.

$$y_{j+1} = x^*_{j-1}(x_j, p_j) = f_j(x_j, u^*_j) \tag{3.1}$$

Now in order to calculate the decoupled derivatives

$$\frac{\partial p^*(x, y)}{\partial x} = \frac{\partial p^*(x, H(x, p))}{\partial x}$$

and $J_x(x, y)$, note the following. If j is not i or $i+1$, $\partial p^*_i(x)/\partial x_j = 0$. Otherwise,

$$\frac{\partial p^*_j(x)}{\partial x_j}(x, y) = -\frac{\partial^2 g^*_j(x_j, x_{j+1})}{\partial x_j\, \partial x_{j+1}}\bigg|_{x_{j-1}=y_{j-1}} \tag{3.2}$$

and

$$\frac{\partial p_j}{\partial x_{j+1}}(x, y) = -\frac{\partial^2 g^*_j(x_j, x_{j+1})}{\partial x^2_{j+1}}\bigg|_{x_{j-1}=y_{j-1}} - Q \tag{3.3}$$

Similarly to calculate $J_x(x, y)$, one needs

$$J_{x_i}(x, y) \equiv \frac{\partial g^*_{i-1}(x_{i-1}, y_i)}{\partial x_i} + \frac{\partial g^*_i(x_i, y_{i+1})}{\partial x_i} \tag{3.4}$$

and for the last term

$$J_{x_M}(x, y) \equiv \frac{\partial g^*_{M-1}}{\partial x_M}(x_{M-1}, y_M) \tag{3.5}$$

Notice that $y_M = x_M$ and $y_0 = x_0$. Since one-stage conglomerate controllability is assumed, if the dimension of the state vectors x_j is n, and the dimension of the control vectors u_j is 1, then one can divide u_j into two parts, $u_j = (u_{D_j}, u_{I_j})$ as in Assumption 2, where $\dim u_{D_j} = n$, $\dim u_{I_j} = l - m$ and $\partial f_j(x_j, u_{D_j}, u_{I_j})/u_{D_j}$ is invertible. From Assumption 2, the implicit function $u_{D_j}(x_j, x_{j+1}, u_{I_j})$ exists, such that

$$f_j(x_j, u_{D_j}(x_j, x_{j+1}, u_{I_j}), u_{I_j}) = x_{j+1} \tag{3.6}$$

This formula also applies

$$g_j^*(x_j, x_{j+1}) = \min_{u_{I_j}} g_j(x_j, u_{D_j}(x_j, x_{j+1}, u_{I_j}), u_{I_j}) \tag{3.7}$$

Also recall that

$$\hat{g}_j(x_j, x_{j+1}, u_{I_j}) \equiv g_j(x_j, u_{D_j}(x_j, x_{j+1}, u_{I_j}), u_{I_j}) \tag{3.8}$$

The derivatives of $\hat{g}_j(\cdot)$ can be found explicitly and will be used to calculate the quantities in (3.2), (3.3), and (3.4). It is convenient to assume that the solution to (3.7) is a C^3 function $u_{I_j}^*(x_j, x_{j+1})$. This will be true locally if $\partial^2 \hat{g}_j/\partial u_{I_j}^2 > 0$. So one can write,

$$g_j^*(x_j, x_{j+1}) = \hat{g}_j(x_j, x_{j+1}, u_{I_j}^*(x_j, x_{j+1})) \tag{3.9}$$

It is desired to determine how to take partial derivatives of (3.9). Now

$$\frac{\partial g_j^*(x_j, x_{j+1})}{\partial x_j} = \frac{\partial \hat{g}_j}{\partial x_j} + \frac{\partial \hat{g}_j}{\partial u_{I_j}} \frac{\partial u_{I_j}^*(x_j, x_{j+1})}{\partial x_j} \tag{3.10}$$

but $\partial \hat{g}_j/\partial u_{I_j} = 0$ at $u_{I_j} = u_{I_j}^*(x_j, x_{j+1})$ so that it follows that

$$\frac{\partial g_j^*(x_j, y_{j+1})}{\partial x_j} = \frac{\partial \hat{g}_j}{\partial x_j} = \frac{\partial g_j(x_j, u_j)}{\partial x_j} + \frac{\partial g_j}{\partial u_{D_j}} \frac{\partial u_{D_j}}{\partial x_j}\bigg|_{x_{j+1} = y_{j+1}} \tag{3.11}$$

To calculate $\partial u_{D_j}/\partial x_j$ one notes that by (3.6) $f_j(x_j, u_{D_j}, u_{I_j}) = x_{j+1}$. Thus

$$\frac{\partial f_j}{\partial x_j} + \frac{\partial f_j}{\partial u_{D_j}} \frac{\partial u_{D_j}}{\partial x_j} = 0 \tag{3.12}$$

so that

$$\frac{\partial u_{D_j}}{\partial x_j} = -\left(\frac{\partial f_j}{\partial u_{D_j}}\right)^{-1} \frac{\partial f_j}{\partial x_j} \tag{3.13}$$

Both (3.11) and (3.13) will be evaluated at the solution to $(P-j)$, $u_j^* = (u_{D_j}^*, u_{I_j}^*)$. Note that

$$y_{j+1} = x_{j+1}^*(x_j, p_j) = f_j(x_j, u_j^*)$$

in this calculation.

In a similar fashion one calculates

$$\frac{\partial u_{D_j}}{\partial x_{j+1}} = \left(\frac{\partial f_j}{\partial u_{D_j}}\right)^{-1} \tag{3.14}$$

and

$$\frac{\partial g_j^*(x_j, x_{j+1})}{\partial x_{j+1}} = \frac{\partial \hat{g}_j}{\partial x_{j+1}} = \frac{\partial g_j}{\partial u_{D_j}} \frac{\partial u_{D_j}}{\partial x_{j+1}} \tag{3.15}$$

To calculate the second derivatives of J^T and the first derivatives of $p^*(x)$, one needs to calculate the second derivatives of $g_j^*(x_j, x_{j+1})$. From (3.15) and (3.9)

$$\frac{\partial^2 g_j^*(x_j, x_{j+1})}{\partial x_{j+1}^2} = \frac{\partial^2 \hat{g}_j}{\partial x_{j+1}^2} + \frac{\partial^2 \hat{g}_j}{\partial x_{j+1}\,\partial u_{I_j}} \frac{\partial u_{I_j}^*(x_j, x_{j+1})}{\partial x_{j+1}} \tag{3.16}$$

Now $\partial \hat{g}_j / \partial u_{I_j}|_{u_{I_j} = u_{I_j}^*(x_j, x_{j+1})} = 0$. Differentiating yields

$$\frac{\partial^2 \hat{g}_j}{\partial u_{I_j}\,\partial x_{j+1}} + \frac{\partial^2 \hat{g}_j}{\partial u_{I_j}^2} \frac{\partial u_{I_j}^*(x_j, x_{j+1})}{\partial x_{j+1}} = 0 \tag{3.17}$$

and

$$\frac{\partial^2 \hat{g}_j}{\partial u_{I_j}\,\partial x_j} + \frac{\partial^2 \hat{g}_j}{\partial u_{I_j}^2} \frac{\partial u_{I_j}^*(x_j, x_{j+1})}{\partial x_j} = 0 \tag{3.18}$$

where u_{I_j} is restricted to be equal to $u_{I_j}^*(x_j, x_{j+1})$. Thus

$$\frac{\partial u_{I_j}^*(x_j, x_{j+1})}{\partial x_{j+1}} = -\left(\frac{\partial^2 \hat{g}_j}{\partial u_{I_j}^2}\right)^{-1} \frac{\partial^2 \hat{g}_j}{\partial u_{I_j}\,\partial x_{j+1}} \tag{3.19}$$

and

$$\frac{\partial u_{I_j}^*(x_j, x_{j+1})}{\partial x_j} = -\left(\frac{\partial^2 \hat{g}_j}{\partial u_{I_j}^2}\right)^{-1} \frac{\partial^2 \hat{g}_j}{\partial u_{I_j}\,\partial x_j} \tag{3.20}$$

Also similar to (3.16) one can write

$$\frac{\partial^2 g_j^*(x_j, x_{j+1})}{\partial x_{j+1}\,\partial x_j} = \frac{\partial^2 \hat{g}_j}{\partial x_{j+1}\,\partial x_j} + \frac{\partial^2 \hat{g}_j}{\partial x_{j+1}\,\partial u_{I_j}} \frac{\partial u_{I_j}^*(x_j, x_{j+1})}{\partial x_j} \tag{3.21}$$

and

$$\frac{\partial^2 g_j^*(x_j, x_{j+1})}{\partial x_j^2} = \frac{\partial^2 \hat{g}_j}{\partial x_j^2} + \frac{\partial^2 \hat{g}_j}{\partial x_j\,\partial u_{I_j}} \frac{\partial u_{I_j}^*(x_j, x_{j+1})}{\partial x_j} \tag{3.22}$$

Notice the derivatives of $\hat{g}_j(\cdot)$ can be calculated from (3.8), using (3.19), (3.20), (3.14), and (3.13) where needed.

For these calculations the following tensors will be needed:

$$\frac{\partial^2 u_{D_j}}{\partial x_j^2}, \frac{\partial^2 u_{D_j}}{\partial x_j\,\partial x_{j+1}}, \frac{\partial^2 u_{D_j}}{\partial x_{j+1}^2}, \text{ and } \frac{\partial u_{D_j}}{\partial u_{I_j}}$$

Now similar to (3.13)

$$\frac{\partial u_{D_j}}{\partial u_{I_j}} = -\left(\frac{\partial f_j}{\partial u_{D_j}}\right)^{-1} \frac{\partial f_j}{\partial u_{I_j}} \tag{3.23}$$

By differentiating (3.6) twice, one writes

$$\frac{\partial^2 (u_{D_j})_k}{\partial x_j^2} = -\sum_l \left[\left(\frac{\partial f_j}{\partial u_{D_j}}\right)_{kl}^{-1} \left(\frac{\partial^2 (f_j)_l}{\partial x_j^2} + \sum_m \frac{\partial^2 (f_j)_l}{\partial x_j\,\partial (u_{D_j})_m} \frac{\partial (u_{D_j})_m}{\partial x_j}\right)\right] \tag{3.24}$$

$$\frac{\partial^2 (u_{D_j})_k}{\partial x_{j+1}^2} = -\sum_l \left[\left(\frac{\partial f_j}{\partial u_{D_j}}\right)_{kl}^{-1} \left(\left(\frac{\partial u_{D_j}}{\partial x_{j+1}}\right)^t \frac{\partial^2 (f_j)_l}{\partial u_{D_j}^2} \frac{\partial u_{D_j}}{\partial x_{j+1}}\right)\right] \tag{3.25}$$

$$\frac{\partial^2 u_{D_j}}{\partial x_j\,\partial x_{j+1}} = -\sum_l \left[\left(\frac{\partial f_j}{\partial u_{D_j}}\right)_{kl}^{-1} \left(\frac{\partial^2 (f_j)_l}{\partial x_j\,\partial u_{D_j}} + \frac{\partial u_{D_j}^t}{\partial x_j} \frac{\partial^2 (f_j)_l}{\partial u_{D_j}^2}\right) \frac{\partial u_{D_j}}{\partial x_{j+1}}\right] \tag{3.26}$$

It will be necessary to calculate later $\partial x^*_{j+1}/\partial x_j$ and $\partial x^*_{j+1}/\partial p_j$. Now by (2.20 *a*)

$$p_j = -Qx^*_{j+1}(x_j, p_j) - \frac{\partial g^*_j(x_j, x^*_{j+1}(x_j, p_j))}{\partial x_{j+1}} \tag{3.27}$$

thus

$$I = -Q\frac{\partial x^*_{j+1}}{\partial p_j} - \frac{\partial^2 g^*_j}{\partial x^2_{j+1}}\frac{\partial x^*_{j+1}}{\partial p_j} \tag{3.28}$$

This implies

$$\frac{\partial x^*_{j+1}}{\partial p_j} = -\left(Q + \frac{\partial^2 g^*_j(x_j, x_{j+1})}{\partial x^2_{j+1}}\right)^{-1}\Bigg|_{x_{j+1}=x^*_{j+1}(x_j,p_j)} \tag{3.29}$$

similarly

$$\frac{\partial x^*_{j+1}}{\partial x_j} = -\left(Q + \frac{\partial^2 g^*_j(x_j, x_{j+1})}{\partial x^2_{j+1}}\right)^{-1}\frac{\partial^2 g^*_j(x_j, x_{j+1})}{\partial x_{j+1}\,\partial x_j}\Bigg|_{x_{j+1}=x^*_{j+1}(x_j,p_j)} \tag{3.30}$$

4. Higher-order algorithm

Now an improvement to the algorithm proposed in Theorem 3 can be offered. For suppose a step size ε is chosen that will bring us on a decreasing sequence of J^T, within δ of an extreme point of J^T. (This extreme point will either be a local minimum or at worst a saddle point.) At this point one can switch to the multi-dimensional Newton algorithm to solve the vector non-linear equation

$$J_x(x, p) = 0 \tag{4.1}$$

$$H(x, p) = x \tag{4.2}$$

Note that if (4.2) is satisfied then must have $p = p^*(x)$. Also if x_j and p_j have dimension n and there are M stages, then (4.1) and (4.2) form a $2nM$ system of equations in $2nM$ unknowns $\begin{pmatrix} x \\ p \end{pmatrix}$. Combine—for notational purposes—(4.1) and (4.2) into one conglomerate vector equation

$$T(x, p) = 0 \quad \text{where } T(x, p) \equiv \begin{pmatrix} J_x(x, p) \\ H(x, p) - x \end{pmatrix} \tag{4.3}$$

The Newton iterates will be

$$\begin{pmatrix} x^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ p^k \end{pmatrix} - (T_z(x^k, p^k))^{-1} T(x^k, p^k) \tag{4.4}$$

where $z \equiv \begin{pmatrix} x \\ p \end{pmatrix}$ and $T_z(\cdot)$ is the gradient of T with respect to z. The algorithm, being a standard Newton algorithm is guaranteed to converge quadratically provided one is close enough to an extreme point, and that $T_z(\cdot)$ is invertible, and that the gradient matrix

$$(z - T_z^{-1}T)_z = T_z^{-1}T_{zz}T_z^{-1}T \tag{4.5}$$

is bounded in a neighbourhood of the extremum. Here T_{zz} is a tensor and the sum of products is taken over the appropriate subscripts.

At this point before going into the details of the above implementation it is instructive to compare the advantages and disadvantages between the above algorithm, and that proposed by Theorem 3. In Theorem 3, to calculate the updated p_j^{i+1}, and x_{j+1}^{i+1}, one only needs information from problems $(P-j)$ and $(P-(j+1))$. This allows for an almost complete decoupling of the high-level problem so that the subproblems need only communicate with their neighbours in order to obtain the updated p_j and x_j.

The method proposed here, however, will have to wait for information to propagate from the solution of the last subproblem $(P-(M-1))$ to $(P-j)$ in order to obtain the p_j and x_{j+1} updates. The subproblems are still being solved in parallel, however, it is merely that the high-level update requires propagation of information from $(P-(M-1))$ to $(P-j)$ in this case. On the other hand, the Newton method has the advantage over Theorem 3 in that it offers a quadratic convergence rate and no propagation errors with time. What is meant by this is that the decoupled updating scheme in Theorem 3 offers only an ε approximation to the actually convergent scheme in Theorem 3, as Lemma 3 indicates. The error in this approximation increases in 'time' (proportional to the number of iterations), but decreases with smaller step sizes. Thus small step sizes are often required to achieve convergence. This unfortunately increases the number of iterations until convergence is achieved. The Newton method, however, will converge to the exact solution and there is no small step size required to achieve convergence.

To see how one can calculate the Newton iterates define

$$\begin{pmatrix} d^k \\ e^k \end{pmatrix} = (T_z(x^k, p^k))^{-1} T(x^k, p^k) = \begin{pmatrix} x^k - x^{k+1} \\ p^k - p^{k+1} \end{pmatrix} \tag{4.6}$$

so that

$$T_z : \begin{pmatrix} d^k \\ e^k \end{pmatrix} = T \tag{4.7}$$

In what follows, drop the superscripts and refer to the successor as $\begin{pmatrix} x' \\ p' \end{pmatrix}$.

Now (4.7) will be written more explicitly. From (3.4), letting

$$y_j = x_j^*(x_{j-1}, p_{j-1}) \quad \text{and} \quad y_{j+1} = x_{j+1}^*(x_j, p_j)$$

one can write

$$\frac{\partial J_{x_i}}{\partial x_{i-1}} d_{i-1} + \frac{\partial J_{x_i}}{\partial y_i}\frac{\partial h_{i-1}}{\partial x_{i-1}} d_{i-1} + \frac{\partial J_{x_i}}{\partial x_i} d_i + \frac{\partial J_{x_i}}{\partial y_{i+1}}\frac{\partial h_i}{\partial x_i} d_i + \frac{\partial J_{x_i}}{\partial y_i}\frac{\partial h_{i-1}}{\partial p_{i-1}} e_{i-1} + \frac{\partial J_{x_i}}{\partial y_{i+1}}\frac{\partial h_i}{\partial p_i} e_i = J_{x_i} \tag{4.8}$$

and

$$-d_i + \frac{\partial h_{i-1}}{\partial x_{i-1}} d_{i-1} + \frac{\partial h_{i-1}}{\partial p_{i-1}} e_{i-1} = h_{i-1}(x_{i-1}, p_{i-1}) - x_i \tag{4.9}$$

where $h_j(x_j, p_j)$ is defined as the jth vector function in $H(x, p)$ and thus $h_j(x_j, p_j) = x^*_{j+1}(x_j, p_j)$.

Combining terms one has the recursive relations in (4.12) written in matrix form. However first define the following matrices:

$$\begin{bmatrix} \dfrac{\partial J_{x_i}}{\partial x_{i-1}} + \dfrac{\partial J_{x_i}}{\partial y_i}\dfrac{\partial h_{i-1}}{\partial x_{i-1}} & \dfrac{\partial J_{x_i}}{\partial y_i}\dfrac{\partial h_{i-1}}{\partial p_{i-1}} \\ \dfrac{\partial h_{i-1}}{\partial x_{i-1}} & \dfrac{\partial h_{i-1}}{\partial p_{i-1}} \end{bmatrix} \equiv R_i \tag{4.10}$$

and

$$\begin{bmatrix} \dfrac{\partial J_{x_i}}{\partial x_i} + \dfrac{\partial J_{x_i}}{\partial y_{i+1}}\dfrac{\partial h_i}{\partial x_i} & \dfrac{\partial J_{x_i}}{\partial y_{i+1}}\dfrac{\partial h_i}{\partial p_i} \\ -1 & 0 \end{bmatrix} \equiv S_i \tag{4.11}$$

where 1 is used to represent the identity matrix. The matrix equation is

$$R_i \cdot \begin{pmatrix} d_{i-1} \\ e_{i-1} \end{pmatrix} + S_i \cdot \begin{pmatrix} d_i \\ e_i \end{pmatrix} = \begin{pmatrix} J_{x_i} \\ h_{i-1} - x_i \end{pmatrix} \tag{4.12}$$

Thus given e_i and d_i one can solve for d_{i-1} and e_{i-1} via (4.13)

$$\begin{pmatrix} d_{i-1} \\ e_{i-1} \end{pmatrix} = R_i^{-1} \cdot \left[\begin{pmatrix} J_{x_i} \\ h_{i-1} - x_i \end{pmatrix} - S_i \begin{pmatrix} d_i \\ e_i \end{pmatrix} \right] \tag{4.13}$$

The four submatrices of R_i and S_i can be evaluated from (3.4) and from the appropriate relationships which follow (3.4) for the first and second derivatives of the $g^*_j(x_j, y_{i+1})$ and the first derivatives of $h_i(x_j, p_j)$. The initial conditions for d_{M-1} and e_{M-1} are needed. First note since by definition $y_M = x_M$, then

$$h_{M-1}(x_{M-1}, p_{M-1}) = y_M = x_M \tag{4.14}$$

for all iterates k. This implies $d_M = 0$—see (4.6). Note that (4.7) can be written without iterate superscripts as

$$T_z \cdot \begin{bmatrix} d_1 \\ \vdots \\ d_M \\ e_0 \\ \vdots \\ e_{M-1} \end{bmatrix} = T \tag{4.15}$$

Using (4.14), the Mth equation and the $2M$th equation from (4.15) yield,

$$\begin{bmatrix} \dfrac{\partial J_{x_M}}{\partial x_{M-1}} + \dfrac{\partial J_{x_M}}{\partial y_M}\dfrac{\partial h_{M-1}}{\partial x_{M-1}} & \dfrac{\partial J_{x_M}}{\partial y_M}\dfrac{\partial h_{M-1}}{\partial p_{M-1}} \\ \dfrac{\partial h_{M-1}}{\partial x_{M-1}} & \dfrac{\partial h_{M-1}}{\partial p_{M-1}} \end{bmatrix} \cdot \begin{bmatrix} d_{M-1} \\ e_{M-1} \end{bmatrix} = \begin{bmatrix} J_{x_M} \\ 0 \end{bmatrix} \tag{4.16}$$

or

$$R_M \begin{bmatrix} d_{M-1} \\ e_{M-1} \end{bmatrix} = \begin{bmatrix} J_{x_M} \\ 0 \end{bmatrix} \tag{4.17}$$

This implies

$$\begin{bmatrix} d_{M-1} \\ e_{M-1} \end{bmatrix} = R_M^{-1} \begin{bmatrix} J_{x_M} \\ 0 \end{bmatrix} \tag{4.18}$$

The d_i and e_i can now be calculated recursively using (4.13) and (4.18). The next iterate or successor will be

$$\begin{pmatrix} x' \\ p' \end{pmatrix} = \begin{pmatrix} x \\ p \end{pmatrix} - \begin{pmatrix} d \\ e \end{pmatrix} \tag{4.19}$$

from (4.4).

5. Discussion

It has been demonstrated that quadratic incentive terms can be used to solve optimal control problems in a parallel environment. It is clear, however, that the parallel scheme proposed in Theorems 2 and 3 suffer from three drawbacks. The first drawback is that the parallel algorithm takes longer to converge than the sequential. This, however, can easily be more than offset by the use of a large number of parallel processors. The second drawback is that both the sequential and parallel algorithms offer only linear convergence rates. The final drawback is, in some cases, the necessity of using a small step size in order to achieve significant decoupling. It is believed that the last two drawbacks can be overcome by using a second-order Newton method for solving the high-level problem. The algorithm consists of using the Newton method to solve the following non-linear vector equations:

$$J_x(x, p) = 0 \tag{5.1}$$

$$H(x, p) = x \tag{5.2}$$

When these equations are satisfied, one is at a stationary point and the system dynamics are satisfied for x and p. Here one is not forced to use a small step size and the Newton method has a quadratic convergence rate. Since the updated x_j and p_j are calculated sequentially via (4.13), we can apply pipelining techniques to achieve further parallelization. Detailed numerical testing of the Newton algorithm will be reported in a forthcoming paper.

Acknowledgment

This work was partly supported by NSF Grant ECS-85-12815, ECS-85-04133, and ECS-85-13163.

References

Brumberg, M. C., Chang, T. S., and Luh, P. B., 1989, *Int. J. Control*, 49, 433.

Chang, T. S., Jin, X. X., and Luh, P. B., 1987, A parallel algorithm for large-scale convex optimal control problems, 1987 American Control Conference, Minneapolis, Minnesota, p. 1975.

Yakowitz, S., and Rutherford, B., 1984, *Applied. Math. Comput.*, 15, 29.