AN ALGORITHM FOR MINIMUM ENERGY CONTROL

by

E. Polak and M. Deparis

Memorandum No. ERL-M 225

1 November 1967

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

An Algorithm for Minimum Energy Control

E. Polak and M. Deparis

Department of Electrical Engineering
and Computer Sciences
Electronics Research Laboratory
University of California, Berkeley

ABSTRACT

This paper presents a very effective, large step, parametric method for solving fixed time, minimum energy, discrete optimal control problems with linear plants and convex constraints on the terminal state. The reason for using parametrization is that it removes the usual severe ill-conditioning effects.

The research reported herein was supported by the National Aeronautics and Space Administration under Grant NsG-354, Suppl. 4.

INTRODUCTION

When confronted with a minimum energy discrete optimal control problem such as the one discussed in this paper, a control engineer finds it natural to invoke necessary conditions such as those given in [1] and to reduce the problem to that of finding a set of co-states and multipliers. Such an approach leads to gradient methods which search in the co-state space, and are similar to the one described by Plant [2] for continuous systems. Unfortunately, these algorithms cannot be proven to converge and, in addition, since they require repeated inversion of highly ill conditioned matrices, they are computationally very inefficient.

A somewhat less direct approach consists of transcribing the discrete optimal control problem into the form of a nonlinear programming problem and then of applying a standard convex programming algorithm, such as Zoutendijk's method of feasible directions [3], which is known to converge. Unfortunately, ill conditioning effects again make these methods computationally inefficient.

Since a straightforward approach does not yield efficient computational procedures, the authors have resorted to an imbedding, or parametric, method which does not depend on matrix inversions and hence does not suffer from ill conditioning effects. Although such an approach may not be particularly appealing esthetically, it has been found by the authors to be extremely efficient computationally. In fact, it enjoys the best of the optimal control and nonlinear programming worlds. It searches the co-state space, it does not involve matrix inversions, it utilizes an "antizigzagging precaution" common in nonlinear programming, it can be shown to converge, and it is simple to program. On the dozen or so examples worked by the authors it proved to be roughly ten times faster than a good optimal control gradient method. The example given at the end of this paper gives some idea of the performance of the algorithm described.

I. Statement of the Problem.

Consider a discrete dynamical system described by a vector difference equation of the form

1 $$x_{i+1} = A_i x_i + b_i u_{i+1}, \quad i = 0, 1, 2, \ldots, N-1,$$

where, for $i = 0, 1, 2, \ldots, N$, $x_i \epsilon R^n$ is the state of the system at time i, and, for $i = 1, 2, \ldots, N$, $u_i \epsilon R^1$ is the input at time i. The matrices A_i, b_i, $i = 0, 1, 2, \ldots, u-1$, are real and are of dimension $n \times n$ and $n \times 1$, respectively.

Suppose that we are given the initial state $\hat{x}_0$ of the system and that we wish to minimize

2 $$\sum_{i=1}^{N} u_i^2$$

subject to the constraints

3 $$|u_i| \leq 1 \quad \text{for} \quad i = 1, 2, \ldots, N$$

and

4 $$q^j(x_N) \leq 0 \quad \text{for} \quad j = 1, 2, \ldots, m,$$

where the $q^j(\cdot)$ are strictly convex, continuously differentiable functions, and x_N is the solution of (1) at time N, corresponding to $x_0 = \hat{x}_0$

and the input-sequence $u = (u_0, u_1, \cdots, u_{N-1})$, i.e.,

5 $$x_N = A_{N-1}A_{N-2}\cdots A_0 x_0 + \sum_{i=0}^{N-1} A_{N-1}A_{N-2}\cdots A_{i+1} b_i u_{i+1}$$

It is convenient to rephrase this optimal control problem as a convex programming problem, by introducing the following substitutions.

Let $u = (u_1, u_2, \cdots, u_N)$, let $d = A_{N-1}A_{N-2}\cdots A_0 x_0$ and, for $i = 0, 1, \cdots, N-1$, let $r_{i+1} = A_{N-1}A_{N-2}\cdots A_{i+1} b_i$, (with $r_N = b_{N-1}$). Then (5) becomes

6 $$x_N = d + \sum_{i=1}^{N} r_i u_i \, .$$

Clearly, (6) defines an affine map from R^N into R^n, and we shall designate this map by r, i.e.,

7 $$r(u) = d + \sum_{i=1}^{N} r_i u_i$$

Finally, let K be a unit hypercube in R^N with center at the origin, i.e.,

8 $$K = \{u = (u_0, u_1, \ldots, u_n) \in E^n \,\big|\, |u_i| \leq 1\}$$

and let Ω be the set of admissible terminal states in R^n, i.e.,

9 $$\Omega = \{x \in E^n \,|\, q^j(x) \leq 0 \quad \text{for} \quad j = 1, 2, \ldots, m\}.$$

then the optimal control problem can be restated as follows.

10 <u>The Convex Programming Problem:</u> Given the map $r : R^N \to R^n$ and the sets K, Ω, defined by (7), (8) and (9) respectively, find a vector $\hat{u}$ in R^N such that

11 $$\hat{u} \in K, \qquad (\text{i.e.}, \ |\hat{u}_i| \leq 1 \ \text{ for } \ i = 1, 2, \ldots, N)$$

12 $$r(\hat{u}) \in \Omega \qquad (\text{i.e.}, \ q^i(r(\hat{u})) \leq 0 \ \text{ for } \ i = 1, 2, \ldots, m)$$

and for all $u \in R^N$ satisfying (11) and (12),

13 $$\|\hat{u}\|^2 \leq \|u\|^2,$$

where

$$\|u\|^2 = \sum_{i=1}^{N} u_i^2 .$$

Following the custom in nonlinear programming, we shall call any $\hat{u} \in R^N$ satisfying (11) and (12) a feasible solution, and we shall call a $\hat{u}$ satisfying (11), (12) and (13) an optimal solution.

Since the above is a standard convex programming problem, it can be solved by such well known algorithms as the method of feasible directions [3]. However, the standard methods would require us to invert matrices whose columns depend on the r_i, which may be exponentially related, and consequently, serious-ill conditioning might and, usually does occur. As it happens, after carefully examining the geometric properties of our problem, it is possible to construct a family of algorithms which not only do not suffer from the structure of the problem, but make great use of it in order to achieve very efficient calculations.

II. <u>The Geometry of the Problem.</u>

In this section we shall establish the geometric properties of our Convex Programming Problem (10), which lead to efficient algorithms for computing an optimal solution $\hat{u}$. We shall also state at this time our assumptions.

Let $\Sigma(\alpha)$ be a closed ball of radius α and center at the origin of R^N. Then $\Sigma(\alpha) \cap K = K$ for $\alpha \geq \sqrt{N}$, where K was defined in (8).

<u>Definition:</u> The map $\mathcal{S}$ from $[0, \sqrt{N}]$ into the set of all subsets of R^n † is defined by

† Whenever it will be necessary to establish the continuity of a function from one Euclidean space into the space of all subsets of another Euclidean space we shall use the Hausdorff metric in the range space. The Hausdorff distance between two sets $\Omega_1, \Omega_2 \subset R^\alpha$ is defined as follows: let $d_1 = \sup_{x \in \Omega_1} \inf_{y \in \Omega_2} ||x - y||$ and let $d_2 = \sup_{y \in \Omega_2} \inf_{x \in \Omega_1} ||x - y||$, then $d = \max(d_1, d_2)$ is the Hausdorff distance.

13 $$\mathcal{L}(\alpha) = r(\Sigma(\alpha)) \cap r(K) = r(\Sigma(\alpha) \cap K)$$

where r is the map defined in (7).

Let $\hat{u}$ be an optimal solution and let $\hat{\alpha}^2 = \langle \hat{u}, \hat{u} \rangle$, then $\hat{\alpha}$ is the smallest $\alpha \in [0, \sqrt{N}]$ for which $\mathcal{L}(\alpha) \cap \Omega$ is not empty, and $r(\hat{u}) \in \mathcal{L}(\hat{\alpha}) \cap \Omega$. We shall shortly show that the Convex Programming Problem (10) is equivalent to the problem of finding a hyperplane which separates the sets $\mathcal{L}(\hat{\alpha})$ and Ω.

We now state two assumptions which will simplify our construction.

14 Assumptions: It will be assumed that (i) the strictly convex set Ω ( ) is compact and that its interior is not empty;

(ii) the interior of $\mathcal{L}(\sqrt{N}) \cap \Omega$ is not empty (i.e., int $r(K) \cap \Omega \neq \emptyset$).

15 Definition: Let $P(v, s)$ be a hyperplane in R^n passing through the point $v \in R^n$, with unit normal s. Thus,

16 $$P(v, s) = \{x \mid \langle x - v, s \rangle = 0\}$$

17 Definition: Let $S = \{s \mid \langle s, s \rangle = 1\}$ be a unit sphere in R^n, with center at the origin and radius 1. Let $v : S \to \partial\Omega$ (the boundary of Ω) be a map defined by the relation

18 $$\langle x - v(s), s \rangle \leq 0 \quad \text{for all } x \in \Omega,$$

i.e., $v(s)$ is the point on the boundary of Ω at which $P(v(s), s)$ is a support hyperplane to Ω, with <u>outward</u> normal s. (Since Ω is strictly convex, the map v is well defined and continuous). The image of S under $v(\cdot)$ is the set $\partial\Omega$, the boundary of Ω (see Fig. 1).

19 <u>Definition:</u> Let $V \subset \partial\Omega$ be a set of points such that for every $v \in V$ there exists an $s \in S$ with the property that $v(s) = v$ and $P(v(s), s)$ separates Ω from the point $d = r(0)$, i.e.,

20 $$\langle d - v(s), s \rangle \geq 0$$

21 <u>Theorem:</u> There is exactly one optimal solution $\hat{u}$ to the Convex Programming Problem (10). Furthermore, the optimal solution $\hat{u}$ satisfies the relation

22 $$r(\hat{u}) \in V$$

<u>Proof:</u> The set $\{u \mid u \in K \text{ and } r(u) \in \Omega\}$ is compact, and not empty by assumption (14). Hence an optimal solution $\hat{u}$ exists. Now, suppose that $u' \neq u''$ are both optimal solutions. Then, for $\lambda \in (0,1)$, $(\lambda u' + (1-\lambda) u'') \in K$, $r(\lambda u' + (1-\lambda) u'') \in \Omega$ and

$$\sum_{i=1}^{N} (\lambda u_i' + (1-\lambda) u_i'')^2 < \lambda \sum_{i=1}^{N} u_i'^2 + (1-\lambda) \sum_{i=1}^{N} u_i''^2 = \sum_{i=1}^{N} u_i'^2 ,$$

Suppose that $\hat{u}$ is the optimal solution to (10). Let $\hat{\alpha} = \sqrt{\langle \hat{u}, \hat{u} \rangle}$, then $r(\hat{u}) \in \mathcal{B}(\hat{\alpha}) \cap \Omega$. But the optimal solution $\hat{u}$ is unique, and therefore $\mathcal{B}(\hat{\alpha}) \cap \Omega = \{r(\hat{u})\}$, singleton. Now, $\mathcal{B}(\hat{\alpha})$ and Ω are both convex and therefore there is a hyperplane $P(r(\hat{u}), s)$, with unit normal s, which separates $\mathcal{B}(\hat{\alpha})$, and hence the point d, from Ω, which proves that $r(\hat{u}) \in V$.

23 Proposition: Let $T = \{s \in S \mid v(s) \in V\}$. Then for every $s \in T$,

24 $$P(v(s), s) \cap \mathcal{B}(\sqrt{N}) \neq \emptyset,$$

where $\emptyset$ is the empty set. (Note, T is a closed set.)

Proof: Suppose $P(v(s), s) \cap \mathcal{B}(\sqrt{N}) = \emptyset$, then, since the hyperplane $P(v(s), s)$ separates the point $d \in \mathcal{B}(\sqrt{N})$ from the set Ω, $P(v(s), s)$ separates $\mathcal{B}(\sqrt{N})$ from Ω. But this contradicts our assumption (14) that $\text{int}\, \Omega \cap \mathcal{B}(\sqrt{N}) \neq \emptyset$.

The above fact enables us to define two functions which we shall need in the construction of our algorithms.

25 Definition: We define the surrogate cost function $c : T \to [0, \sqrt{N}]$ by the relation

26 $$c(s) = \min_{\alpha \in [0, \sqrt{N}]} \{\alpha \mid P(v(s), s) \cap \mathcal{B}(\alpha) \neq \emptyset\}$$

27 Proposition: Because of assumption (14), for every $s \in T$, $c(s) < \sqrt{N}$.

28 <u>Proposition:</u> Let $s \in T$ be arbitrary, then

29 $$\mathcal{S}(c(s)) \cap P(v(s), s) = \{w(s)\},$$

a singleton.

30 <u>Definition:</u> We define the map $w : T \to R^n$ by the relation (29).

We are now ready to relate the quantities developed above to $\hat{u}$, the optimal solution of (10).

31 <u>Proposition:</u> Let $\hat{u}$ be the optimal solution of (10) and let $\hat{v} = r(\hat{u})$. Then, (because of assumption (14)) there exists a vector $\hat{s} \in T$ such that

32 $$v(\hat{s}) = w(\hat{s}) = \hat{v}.$$

33 <u>Theorem:</u> Let $\hat{s} \in T$ be such that $w(\hat{s}) = v(\hat{s})$, and let $\hat{\lambda} \leq 0$ be the solution of the equation

34 $$\left\langle d + \sum_{i=1}^{N} \operatorname{sat} \langle \lambda \hat{s}, r_i \rangle r_i - v(\hat{s}), \hat{s} \right\rangle = 0,$$

then $\hat{u} = (\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_N)$, with

35 $$\hat{u}_i = \operatorname{sat} \langle \hat{\lambda} \hat{s}, r_i \rangle, \qquad i = 1, 2, \ldots, N,$$

is the optimal solution to the Convex Programming Problem (10).

Since $\hat{u}$, the optimal solution of (10) is unique, and, by assumption $v(\hat{s}) = w(\hat{s})$, we must have $r(\hat{u}) = v(\hat{s})$ and $\sqrt{\langle \hat{u}, \hat{u} \rangle} = c(\hat{s})$.

Now, consider the problem of minimizing $<u,u>$ subject to $u \epsilon K$ and $r(u) \epsilon P(v(\hat{s}), \hat{s})$. The solution u^* to this problem obviously satisfies $\sqrt{<u^*, u^*>} = c(\hat{s})$, $r(u^*) = w(\hat{s})$, and from the Kuhn-Tucker conditions [1, 4] which are both necessary and sufficient for this case, it follows that $\hat{u}$ is determined by (34) and (35). Thus, $u^* \epsilon K$, $r(u^*) \epsilon \Omega$ and $\sqrt{<u^*, u^*>} = c(\hat{s}) = \sqrt{<\hat{u}, \hat{u}>}$. Consequently, u^* is also an optimal solution to (1 0), and since the optimal solution to (1 0) is unique, we have $u^* = \hat{u}$, which completes our proof.

The effect of this theorem is to transcribe the original Convex Programming Problem (1 0) into a geometric problem in a lower dimension space (R^n).

36 The Geometric Problem: Find a vector $\hat{s} \epsilon T$ such that $w(\hat{s}) = v(\hat{s})$.

Because of the preceding discussion we shall refer to any $\hat{s} \epsilon T$ satisfying $w(\hat{s}) = v(\hat{s})$ as an optimal solution to the geometric problem. It should be clear from Theorem (33) that the point $w(s)$ is easily obtained by first computing the point $u(s) = (u_1(s), u_2(s), \ldots, u_k(s))$, according to the formula

37 $$u_i(s) = \text{sat} <\lambda(s)s, r_i>, \qquad i = 1, 2, \ldots, k,$$

where $\lambda(s)$ is the root of the equation

38 $$\left\langle d + \sum_{i=1}^{k} \mathrm{sat}\langle \lambda s, r_i \rangle r_i - v(s), s \right\rangle = 0,$$

and then by evaluating $w(s)$ according to the formula

39 $$w(s) = r(u(s)) = d + \sum_{i=1}^{N} u_i(s) r_i$$

The computation of $v(s)$ may present some problems when the point $v(s)$ is on an edge of Ω or when the functions q^j are of an unfavorable nature. We shall discuss the computation of $v(s)$ for the case when the q^j are quadratic forms in Section IV.

III. Algorithms.

The convergence of the algorithms we are about to develop depends on the continuity of the functions c and w defined in (26), (30), respectively. We therefore begin by establishing these properties.

39 Theorem: The function $c : T \to [0, \sqrt{N}]$ is continuous.

Proof: Let s be an arbitrary point in the relative interior of T. Then $c(s) > 0$ and by (27) $c(s) < \sqrt{N}$. Let $\epsilon > 0$ be any number such that $(c(s) \pm \epsilon) \in (0, \sqrt{N})$. Let $P_{\epsilon+}$, $P_{\epsilon-}$ be two hyperplanes parallel to $P(v(s), s)$ which are support hyperplanes to $\mathcal{L}(c(s) + \epsilon)$, $\mathcal{L}(c(s) - \epsilon)$, respectively. Then, since $\mathcal{L}(c(s) + \epsilon) \supset \mathcal{L}(c(s)) \supset \mathcal{L}(c(s) - \epsilon)$, and the containment is strict $P_{\epsilon+} \neq P_{\epsilon-} \neq P(v(s), s)$. Consequently, since $v: S \to \partial\Omega$ is continuous (see (17)) there exists a $\delta > 0$ such that for all $s' \in B(s, \delta) \cap T$, where $B(s, \delta)$ is an open ball of radius δ and center s, $P(v(s'), s')$ separates Ω from $\mathcal{B}(c(s) - \epsilon)$ but not from $\mathcal{B}(c(s) + \epsilon)$. Thus, $|c(s') - c(s)| < \epsilon$ for all $s' \in B(s, \delta) \cap T$, which proves that c is continuous at all points s in the relative inferior of T. Now let s be a point on the boundary of T. Then $c(s) = 0$, and for any sequence $s_i \in T$, $i = 1, 2, \ldots$, such that $s_i \to s$, it can be shown that $c(s_i) \to 0$. This completes our proof.

40 <u>Theorem:</u> The function $w: T \to R^n$ is continuous.

<u>Proof:</u> Let $s^* \in T$ be arbitrary and let $s_i \in T$, $i = 1, 2, 3, \ldots$ be any sequence which converges to s^*. Then, since $c(\cdot)$ is continuous, $c_i = c(s_i)$, $i = 1, 2, 3, \ldots$, is a sequence which converges to $c^* = c(s^*)$ and, since $\mathcal{B}(\cdot)$ is a continuous map, $\mathcal{B}(c_i) \to \mathcal{B}(c^*)$. Now consider the sequence $w_i = w(s_i)$, $i = 1, 2, \ldots$. Since $w_i \in \mathcal{B}(c_i) \subset \mathcal{B}(\sqrt{N})$, a compact set, the sequence w_i, $i = 1, 2, \ldots$ must have a convergent subsequence, w_j with $j \in K$, an index set. Suppose $w_j \to w^*$. Then, since $w_j \in \mathcal{B}(c_j)$, $w^* \in \mathcal{B}(c^*)$. Also, since $w_j \in P(v(s_j), s_j)$, $w^* \in P(v(s^*), s^*)$. Hence $w^* \in \mathcal{B}(c^*) \cap P(v(s^*), s^*) = \{w(s^*)\}$, i.e., $w^* = w(s^*)$, and all convergent subsequences of $\{w_i\}$ converge to $w^* = w(s^*)$. Consequently, $w_i \to w^*$, i.e., $w(s_i) \to w(s^*)$, which proves that w is a continuous map.

The conditions under which an algorithm for computing a point $\hat{s} \in T$ such that $v(\hat{s}) = w(\hat{s})$ will converge and the freedom with which it can be chosen are illuminated by the following two theorems.

41 Theorem: Let $a : T \rightarrow T$ be an algorithm† such that for every $s \in T$, for which $v(s) \neq w(s)$, there exists an $\epsilon(s) > 0$ and a $\delta(s) > 0$ with the property that

42 $$c(a(s')) - c(s') > \delta \quad \text{for all} \quad s' \in B(s,\epsilon) \cap T,$$

where $B(s,\epsilon)$ is an open ball of radius ϵ and center s. Let s_i, $i = 0, 1, 2, \ldots,$ be any sequence generated according to the rule

43 $$s_{i+1} = a(s_i), \quad i = 0, 1, 2, \ldots$$

with $s_0 \in T$ arbitrary. Then every convergent subsequence of the sequence $\{s_i\}$ converges to a point s^* in T satisfying $v(s^*) = w(s^*)$ (where s^* may depend on the subsequence).

Proof: Let $\{s_i\}$, generated by (43), be a sequence in the closed set T, and let $\{s_j\}$, $j \in K$, be a convergent subsequence of $\{s_i\}$. Now suppose that $s_j \rightarrow s^*$, for $j \in K$, and that $v(s^*) \neq w(s^*)$. Then, by hypothesis, there exist a n $\epsilon^* > 0$ and a $\delta^* > 0$, and a $p \in K$ such that

44 $$s_j \in B(s^*, \epsilon^*) \quad \text{for all } j \in K, \text{ satisfying } j \geq p$$

and

† We shall call a function an algorithm if it can be used for finding points $\hat{s} \in T$ such that $v(\hat{s}) = w(\hat{s})$.

45 $$c(s_\ell) - c(s_j) > \delta^* \text{ for all } \ell > j \geq p, \text{ with } \ell, j \in K.$$

But, c is a continuous function and hence $c(s_j) \to c(s^*) = c^*$ for $j \in K$, which is impossible in view of (45). Hence we have arrived at a contradiction which proves the theorem.

As an immediate consequence we get the following results.

46 Corollary: Let $a : T \to T$ be a continuous function such that for every $s \in T$ for which $v(s) \neq w(s)$

47 $$c(a(s)) - c(s) > 0$$

then a is an algorithm which satisfies the assumptions of theorem (41).

48 Corollary: Let $a : T \to T$ be an algorithm satisfying the conditions of theorem (41) and let $\{s_i\}$ $i = 0, 1, 2, \ldots,$ be any sequence in T with the property that

49 $$c(s_{i+1}) - c(a(s_i)) \geq 0 \text{ for } i = 0, 1, 2, \ldots$$

Then every convergent subsequence $\{s_j\}$, $j \in K$, of $\{s_i\}$ converges to a point s^* satisfying $v(s^*) = w(s^*)$ (where s^* may depend on the subsequence).

50 Corollary: Let $a : T \to T$, $b : T \to T$ be any two algorithms which satisfy the conditions of theorem (41). If $d : T \to T$ is an algorithm with the property that

51 $$d(s) = a(s) \quad \text{or} \quad d(s) = b(s) \quad \text{for all} \quad s \in T,$$

then d also satisfies the conditions of theorem (40).

The gist of the last corollary is that we may switch from one algorithm to another in any way we please. This observation is important, since the algorithm a may be fast only in some parts of T while b may be faster than a in others.

The manner in which the corollaries (46) and (48) are used will become clear from what follows.

We begin with the preliminary steps of the construction of a family of algorithms for computing points $s \in T$ which satisfy $v(s) = w(s)$.

52 <u>Definition:</u> Let σ be a map from T into the set of all subsets of T defined by

$$\sigma(s) = \{s' \in T \,|\, s' = \frac{s + \lambda(w(s) - v(s))}{||s + \lambda(w(s) - v(s))||}, \quad 0 \leq \lambda \leq 1$$

Since w - v is a continuous function, so is σ.

53 <u>Theorem:</u> For any $s \in T$, the function c assumes its maximum value on the curve $\sigma(s)$ at exactly one point. Furthermore the point $s^* \in \sigma(s)$ which maximizes c(s) on $\sigma(s)$ is also the only local maximum point of c in $\sigma(s)$.

<u>Proof:</u> Let

54 $$s(\lambda) = \frac{s + \lambda(w(s) - v(s))}{||s + \lambda(w(s) - v(s))||},$$

then $s(\lambda) \in \sigma(s)$ for all $\lambda \in [0, \lambda']$, where λ' is the largest value of λ in $[0,1]$ satisfying $s(\lambda') \in \sigma(s)$. Now, suppose that the function $c(s)$ does not assume a unique local maximum on $\sigma(s)$. Then there must be at least three points $\lambda_1 < \lambda_2 < \lambda_3$ in $[0, \lambda']$ such that

55 $$c(s(\lambda_1)) = c(s(\lambda_2)) = c(s(\lambda_3) .$$

But this contradicts the fact that there can only be two hyperplanes $p(v(s(\lambda_1)), s(\lambda_1)), P(v(s(\lambda_2)), s(\lambda_2))$, with normals in the two dimensional plane spanned by the vectors s and $(w(s) - v(s))$, which separates Ω from $(c(s(\lambda_1)))$ and which, at the same time, are support hyperplanes to these sets. Consequently for the function c there is a unique local maximum point on $\sigma(s)$ which is also a global maximum point.

Definition: Let m be a map from T into $[0, \sqrt{N}]$ defined by

56 $$m(s) = \max_{s' \in \sigma(s)} c(s')$$

57 Proposition: The map m defined by (56) is continuous.

Proof: Since c and σ are continuous, the composite map $c\sigma$ which maps T into subintervals of $[0, \sqrt{N}]$, according to $(c\sigma)(s) = \{\alpha \in [0, \sqrt{N}] \mid \alpha = c(s'), s' \in \sigma(s)\}$, is continuous. Hence the map m is continuous.

58 Proposition: For every $s \in T$ such that $w(s) \neq v(s)$ $m(s) > c(s)$.

<u>Proof:</u> Let $s \in T$ be such that $v(s) \neq w(s)$. Then $P(v(s), s)$ separates Ω from $(c(s))$ and is a support hyperplane to both Ω and to $(c(s))$, with the respective points of support being $v(s)$, $w(s)$, which are distinct. Hence there must be a hyperplane $P(v(s'), s')$, with $s' \in \sigma(s)$, which separates $(c(s))$ from Ω and $P(v(s'), s') \cap (c(s)) = \emptyset$. Hence $c(s') > c(s)$ and therefore $m(s) > c(s)$.

59 <u>Definition:</u> Let the algorithm $a : T \rightarrow T$ be defined by the relation that $a(s) \in \sigma(s)$ is the point in $\sigma(s)$ which maximizes $c(s)$ over $\sigma(s)$, i.e., $c(a(s)) = m(s)$.

60 <u>Theorem:</u> The algorithm a defined in (59) satisfies the assumptions of theorem (41).

<u>Proof:</u> By construction, $c(a(s)) > c(s)$ for all s such that $v(s) \neq w(s)$. Hence, by corollary (46), we only need to show that a is continuous.

Let s be any point in T and let $s_i \in T$, $i = 0, 1, 2, \ldots$, be any sequence which converges to s. Then the sequence $a(s_i)$ $i = 0, 1, 2, \ldots$, must contain a convergent subsequence, say $a(s_j)$ with $j \in K$, an index set. Suppose that for $j \in K$, $a(s_j) \rightarrow s^*$ and that $s^* \neq a(s)$. Then since $a(s_i) \in \sigma(s_i)$ and a is continuous, it follows that $s^* \in \sigma(s)$ and hence, by theorem (53)

61 $$m(s) > c(s^*)$$

since $m(s) = c(a(s))$.

Now, $m(s_i) = c(a(s_i))$ by definition of a, and m and c are continuous. Hence for $j \in K$, $m(s_j) \to m(s)$, $c(a(s_j)) \to c(s^*)$, and therefore $m(s) = c(s^*)$. But this contradicts (61). Hence $s^* = a(s)$. Since every convergent subsequence of $\{a(s_i)\}$ converges to a(s), it follows that $a(s_i) \to a(s)$ and hence a is a continuous map.

62 Corollary: Let a be the algorithm defined in (59). Then, for $0 < \lambda \leq 1$, the map $a_\lambda : T \to T$ defined by

63 $$a_\lambda(s) = \frac{s + \lambda(a(s) - s)}{\|s + \lambda(a(s) - s)\|}$$

is also an algorithm satisfying the assumptions of theorem (41).

Proof: For λ fixed, a_λ is obviously continuous and by theorem (53) and proposition (58), $c(a_\lambda(s)) > c(s)$ for all $s \in T$ such that $v(s) \neq w(s)$.

64 Remark: Since $\lambda > 0$ can be chosen arbitrarily small without upsetting the important properties of $a_\lambda(s)$, it is easy to choose simple rules for picking s_{i+1} on $\sigma(s_i)$ and still satisfy $c(s_{i+1}) > c(a_\lambda(s_i))$ for some fixed λ. The resulting sequences $\{s_i\}$ will contain subsequences $\{s_j\}$, $j \in K$, which converge to points s^* satisfying $v(s^*) = w(s^*)$.

IV. Computational Procedures and Antizigzagging Precautions.

We shall now suppose that the constraint functions q^i are quadratic forms, i.e., 65 $$q^i(x) = \langle x - \xi_i, Q_i(x - \xi_i)\rangle - \beta_i, \quad i = 1, 2, \ldots, m, \text{ with } \beta_i > 0,$$

where the ξ_i are given n-vectors and the Q_i are $n \times n$ symmetric, positive definite matrices. Suppose we wish to implement the algorithm $a(\cdot)$ defined in (59), or for some $0 < \lambda \leq 1$, the algorithm $a_\lambda(\cdot)$ defined in (63). For the computation to be effective, we must be able to compute $a_\lambda(s)$, for a given s, in a finite number of steps. Now from the way $a(\cdot)$ was defined, we face two sources of difficulty. The first is due to the fact that $m(s)$ cannot be computed in a finite number of steps. However, because of corollaries (48) and (62), and the fact that λ may be taken very small in $a_\lambda(\cdot)$, it is clear that given $s_i \in T$, we may choose s_{i+1} to be just about any point in $\sigma(s_i)$ for which $c(s_{i+1}) > c(s_i)$ and the sequence $\{s_i\}$ will have subsequences which converge to optimal points s^*, satisfying $v(s^*) = w(s^*)$.

The major source of difficulty, therefore, is due to the fact that $v(s)$ cannot be evaluated in a finite number of steps when $v(s)$ is on an edge of the boundary of the set $\Omega = \{v \mid q^i(v) \leq 0, \quad i = 1, 2, \ldots, m\}$, i.e., when there is more than one $i \in \{1, 2, \ldots, m\}$ such that $q^i(v(s)) = 0$ and $\nabla q^i(v(s)) \neq \alpha s$, for some $\alpha > 0$ and $i = 1, 2, \ldots, m$. (The reason for this is that when $v(s)$ is on an edge we must solve for the intersection of at least two surfaces and one hyperplane, and this cannot be done in a finite number of steps). Hence, we must avoid evaluating $v(s)$ exactly for all points $s \in T$ for which $v(s) \neq w(s)$.

The above considerations are reflected in the following sub-procedures P1 to P6 which are then organized into a flow chart. The antizigzagging precaution, or the manner in which we improve the accuracy of the evaluation of v(s), w(s), c(s) and a(s) should be clear from the flow chart. The scheme we use is related to that of Zoutendijk[3].

Finally, since the assumption $\mathcal{S}(\sqrt{N}) \cap \Omega \neq \emptyset$ was only made for the sake of simplifying our exposition, we relinquish this assumption at this stage and shall have no test for it. Instead, we include a test to establish that the problem has no solution. (When $\mathcal{S}(\sqrt{N}) \cap \Omega \neq \emptyset$ but $\text{int}\, \mathcal{S}(\sqrt{N}) \cap \Omega = \emptyset$, the empty set, the algorithm $a(\cdot)$ defined in (59) may have discontinuities at some optimal points $\hat{s}$ described in P6. However, it is not too difficult to show that these discontinuities do not affect the convergence of the sequences generated by $a(\cdot)$ to an optimal solution).

P1. Initial Guess Procedure: s_0

Step 1 Choose a point z in the interior of Ω, i.e., z satisfies[+]

66 $$q^i(z) < 0 \quad \text{for} \quad i = 1, 2, \ldots, m$$

Step 2 Find $\lambda \in [0,1]$ such that

67 $$q^i(\lambda z + (1-\lambda) d) = 0 \quad \text{for some} \quad i \in \{1, 2, \ldots, m\}$$

and

68 $$q^j(\lambda z + (1-\lambda) d) \leq 0 \quad \text{for all} \quad j \in \{1, 2, \ldots, m\}$$

Step 3 With λ computed in step 2 and i any integer in $\{1, 2, \ldots, m\}$ satisfying (67), let

69 $$s_0 = \frac{\nabla q^i(\lambda z + (1-\lambda)d)}{\|\nabla q^i(\lambda z + (1-\lambda)d)\|}$$

then s_0 is in T and $v(s_0) = \lambda z + (1-\lambda)d$.

[+] To find such a point z, use P2 to compute two points v', v'' on the boundary of Ω and then set $z = \frac{1}{2}(v' + v'')$. Some experimentation will be required to find a z which is "centrally" located in the interior of Ω.

P2 <u>Computation of v(s), w(s), c(s), $\sigma(s)$ when v(s) is not on an edge of Ω.</u>

Let $s \in T$ be given. To verify that v(s) is not on an edge, carry out the following calculation.

<u>Step 1</u> For $i = 1, 2, \ldots, m$ solve for α^i, v_i the equations

70 $$\nabla q^i(v_i) = \alpha^i s, \quad \alpha^i > 0$$

71 $$q^i(v_i) = 0 \, .$$

If there is a v_j among the v_i computed above such that $q^i(v_j) \leq 0$ for all $i \in \{1, 2, \ldots, m\}$, then v(s) is not on an edge.

<u>Step 2</u> Since v(s) is not an edge of Ω by assumption, find the $\ell \in \{1, 2, \ldots, m\}$ satisfying

72 $$q^j(v^\ell) \leq 0, \quad j = 1, 2, \ldots, m,$$

and set $v(s) = v_\ell$.

<u>Step 3</u> Find the $\lambda^* \leq 0$ which satisfies

73 $$\left\langle d + \sum_{i=1}^{N} \text{sat} < \lambda s, r_i > r_i - v(s), s \right\rangle = 0$$

then

74 $$w(s) = d + \sum_{i=1}^{N} \text{sat} < \lambda^* s, r_i > r_i$$

75 $$c(s) = \left(\sum_{i=1}^{N} (\text{sat} < \lambda^* s, r_i >)^2 \right)^{1/2}$$

76 $$\sigma(s) = \{ s' \mid s' = s + \lambda(w(s) - v(s)),\ 0 \leq \lambda \leq 1, \text{ and }$$

$$< d - v(s'), s' > \ \geq \ 0 \}$$

P3 <u>ϵ - Approximate Computation of $v(s)$, $w(s)$, $c(s)$ and $\sigma(s)$ when $v(s)$ is on an edge of Ω and $s \epsilon \sigma(s_0)$</u>

Suppose s_0, $v(s_0)$, $\sigma(s_0)$ and $\epsilon > 0$ are given and suppose $s \epsilon \sigma(s_0)$.

<u>Step 1</u> For $i = 1, 2, \ldots, m$, solve for α^i, v_i the equations

77 $$\nabla q^i(v_i) = \alpha^i s, \quad \alpha^i > 0$$

78 $$q^i(v_i) = 0$$

Verify that $v(s)$ is on an edge by showing that for every $i \epsilon \{1, 2, \ldots, m\}$ there is a $j \epsilon \{1, 2, \ldots, m\}$ such that

79 $$q^j(v_i) \geq 0,$$

when v_i satisfies (77), (78).

Step 2a Suppose $v(s_0)$ is not on an edge of Ω. Then, by a process of consecutive halving of subarcs of $\sigma(s_0)$, between s_0 and s, find a point $s' \in \sigma(s_0)$ between s_0 and $\hat{s}$, such that $v(s')$ is not on an edge, i.e., for some $\ell \in \{1, 2, \ldots, m\}$

80 $$\nabla q^{\ell}(v(s')) = \alpha s', \quad \text{with} \quad \alpha > 0$$

81 $$q^{\ell}(v(s')) = 0 ,$$

and for all $i \in \{1, 2, \ldots, m\}$

82 $$q^{i}(v(s')) \leq 0 .$$

In addition, $v(s')$ must satisfy

83 $$q^{j}(v(s')) + \epsilon \geq 0$$

for some $j \in \{1, 2, \ldots, m\}$.

Step 3 For the j which maximizes the left hand side of (83), find $\alpha > 0$ such that

84 $$q^{j}(z + \alpha(v(s') - z)) = 0,$$

where z is the interior point of Ω used in P1. Let $v' = z + \alpha((v(s') - z))$.

We now set $v_{\epsilon}(s)$ to be the point which minimizes

85 $$||v(s') - v||^2$$

subject to

86 $$<\nabla q^{\ell}(v(s')), v - v(s')> = 0$$

$$<\nabla q^{j}(v'), v - v'> = 0$$

i.e.,

88 $$v_{\epsilon}(s) = N^T(N N^T)^{-1}(b - N v(s') = v(s')$$

where, $b = (<\nabla q^{\ell}(v(s'), v(s')>, <\nabla q^{j}(v'), v'>)$, and N is a $2 \times n$ matrix whose first row is $\nabla q^{\ell}(v(s'))$ and whose second row is $\nabla q^{j}(v')$.

Step 2b Suppose that $v(s_0)$ (or $v_{\epsilon}(s_0)$) is on an edge of Ω. Since $v(s_0)$ is known, $w(s_0)$ (or $w_{\epsilon}(s_0)$) is known. Find an $\alpha > 0$ such that the point

89 $$v(s_0') = z + \alpha(w(s) - z)$$

is a point on the boundary of Ω, i.e.,

90 $$q^i(v(s_0')) \leq 0 \text{ for } i = 1, 2, \ldots, m$$

and for at least one $j \in \{1, 2, \ldots, m\}$

91 $$q^j(v(s_0')) = 0$$

With s_0' defined by

92 $$s_0' = \frac{\nabla q^j(v(s_0'))}{||\nabla q^j(v(s_0'))||} ,$$

we see that $v(s_0')$ is consistently defined. Let

93 $$\tilde{\sigma}(s_0') = \left\{ s' \,|\, s' = \frac{(\lambda s_0' + (1 - \lambda) s)}{||\lambda s_0' + (1 - \lambda) s)||} , \quad 0 \leq \lambda \leq 1 \right\}$$

Now set $s_0 = s_0'$, $\sigma(s_0) = \tilde{\sigma}(s_0')$ and use steps 2a and 3 to compute $v_\epsilon(s)$.

<u>Step 4</u> Let $\lambda_\epsilon \leq 0$ be the solution to the equation

94 $$\left\langle d + \sum_{i=1}^{N} \text{sat} < \lambda s, \; r_i > r_i - v_\epsilon(s), \; s \right\rangle = 0$$

then we set

95 $$w_\epsilon(s) = d + \sum_{i=1}^{N} \text{sat} < \lambda_\epsilon s, \; r_i > r_i$$

96 $$c_\epsilon(s) = \left(\sum_{i=1}^{N} (\text{sat} < \lambda_\epsilon s, \; r_i)^2 \right)^{1/2}$$

97 $$\sigma_\epsilon(s) = \left\{ s' \,\middle|\, s' = \frac{s + \lambda(w_\epsilon(s) - v_\epsilon(s))}{\|s + \lambda(w_\epsilon(s) - v_\epsilon(s))\|} \text{ for } 0 \leq \lambda \leq 1, \text{ and } \langle d - v_\epsilon(s'), s' \rangle \geq 0 \right\}$$

98 Remark: Note that v_ϵ, w_ϵ, are point to set and not point to point maps since their values are not defined uniquely. Also note that in the approximate evaluation of v_ϵ, we have substituted the wedge formed by the hyperplanes $P(v(s'), s')$, and

$$P\left(v', \frac{\nabla q^j(v')}{\|\nabla q^j(v')\|}\right)$$

for the set Ω. This wedge contains the set Ω and serves as a satisfactory local approximation to Ω for our purposes.

P4. Computation of s_{i+1} from s_i.

Suppose we have computed s_i, $i = 0, 1, 2, \ldots$ in T, and let M be a given positive integer.

Step 1. Compute M points (equally spaced) $y_1, y_2, \ldots, y_M$ in $\sigma(s_i)$ (or $\sigma_\epsilon(s_i)$) using P2 or P3.

Step 2 Compute $c(y_i)$ (or $c_\epsilon(y_i)$) for $i = 1, 2, \ldots, M$ and find $j \in \{1, 2, \ldots, M\}$ such that

99 $$c(y_j) \geq c(y_i) \quad i \in \{1, 2, \dots, M\}$$

or

100 $$c_\epsilon(y_j) \geq c_\epsilon(y_i)$$

Step 3 Set

101 $$s_{i+1} = y_j$$

In the flow chart given, M is first taken to be one, and is only increased if an increase in the "surrogate cost" c is not obtained in the first try.

P 5. Verification of Feasibility.

If for any $s \in T$, the plane $P(v(s), s)$ (or $P(v_\epsilon(s), s)$) separates strictly the set $\mathcal{L}(\sqrt{N})$ from Ω, then, obviously, there is no feasible solution.

Step 1 Compute $v(s)$ (or $v_\epsilon(s)$) by P 2.

Step 2 Compute $\lambda^* < 0$ such that

102 $$|\langle \lambda^* s, r_i \rangle| \geq 1 \quad \text{for} \quad i = 1, 2, \dots, k .$$

If

103 $$\left\langle d + \sum_{i=1}^{k} \text{sat} \langle \lambda^* s, r_i \rangle r_i - v(s), s \right\rangle > 0$$

then there is no feasible solution to the optimal control problem.

P6. Verification of Optimality of s_0

When the interior of $\Omega \cap \mathcal{B}(\sqrt{N})$ is empty, but $\Omega \cap \mathcal{B}(\sqrt{N}) \neq \emptyset$, i.e., there exist feasible solutions, it may happen that $\hat{s}$ is an optimal solution in the sense that

104 $$\hat{u}_i = \text{sat}\langle \hat{\lambda}\hat{s}, r_i \rangle, \quad i = 1, 2, \ldots, k$$

is an optimal control sequence (for a suitably chosen $\hat{\lambda}$), but

105 $$w(\hat{s}) \neq v(\hat{s})$$

Nevertheless, any convergent sequence $\{s_i\}$ generated by the algorithm $a(\cdot)$ defined in (59), will converge to an optimal $\hat{s}$, whether this $\hat{s}$ satisfies $w(\hat{s}) = v(\hat{s})$ or not. However, if s_0 is optimal, and $v(s_0) \neq w(s_0)$, s_1 will not be optimal and hence, after the initial guess s_0 is computed, we should verify its optimality as follows.

Step 1 Find if there is more than one root $\hat{\lambda}$ to the equation

106 $$\left\langle d + \sum_{i=1}^{k} \text{sat} \langle \lambda s_0, r_i \rangle r_i - v(s_0), s_0 \right\rangle = 0$$

Step 2 If the answer is yes then s_0 is optimal. If there is exactly one solution to (3), check if $v(s_0) = w(s_0)$. If there is only one $\hat{\lambda}$ satisfying (106) and $v(s_0) \neq w(s_0)$ then s_0 is not optimal.

This procedure is not shown in the flow chart.

Example: The behavior of the algorithm may be judged to some extent by its performance in the following case: a 50 sampling period process for a tenth order system with two quadratic constraints on the terminal state.

Thus, $A = (a_{ij})$ is a diagonal 10×10 matrix with $a_{ij} = 0$ for $i \neq j$ and $a_{11} = 0.9$, $a_{22} = 0.9$, $a_{33} = 0.5$, $a_{44} = 0.5$, $a_{55} = 0.9$, $a_{66} = 0.9$, $a_{77} = 0.6$, $a_{88} = 0.6$, $a_{99} = 0.9$, and $a_{1010} = 0.9$.

$b = (7.60,\ 7.60,\ 0.10,\ 15.20,\ 15.20,\ 0.10,\ 0.10,\ 7.60,\ 7.60)$

$x_0 = (3000.0,\ 3000.0,\ 1000.0,\ 6000.0,\ 6000.0,\ 1000.0,\ 3000.0,\ 3000.0).$

$$q'(x) = (x^1 - 4)^2 + (x^2)^2 + (x^3)^2 + (x^4)^2 + (x^5)^2 + (x^6)^2 + (x^7)^2 + (x^8)^2 + (x^9)^2 + (x^{10})^2 - 25$$

$$q^2(x) = q^1(x) - (x^1 - 4)^2 + (x^1 + 4)^2$$

The extent of possible ill conditioning that could occur in a gradient method approach to the problem is indicated by the nature of A^{50}, which is computed to be

$$A^{50} = \mathrm{Diag}(0.51537738 \times 10^{-2},\ 0.51537738 \times 10^{-2},\ 0.88817842 \times 10^{-15},\ 0.88817844 \times 10^{-15},\ 0.51537738 \times 10^{-2},\ 0.51537738 \times 10^{-2},\ 0.80828110 \times 10^{-11},\ 0.80828110 \times 10^{11},\ 0.51537738 \times 10^{-2},\ 0.51537738 \times 10^{-2}).$$

The resulting vector d is

$$d = (0.15461321 \times 10^{2},\ 0.15461321 \times 10^{2},\ 0.88817842 \times 10^{-12},\ 0.88817842 \times 10^{-12},\ 0.30922643 \times 10^{2},\ 0.30922643 \times 10^{2},$$

$0.80828110 \times 10^{-8}$, $0.80828110 \times 10^{-8}$, 0.15461321×10^{2},

0.15461321×10^{2},

With $\epsilon' = 10^{-2}$, it can be seen from the table below that it took 3 steps to solve the problem, with $||v(s_2) - w(s_2)|| = 0.24153891 \times 10^{-3}$. The optimal value of the surrogate cost, $c(s_2) = 0.36357584$. The time required to solve this problem on a CDC 6400 digital computer was 5 seconds.

CONCLUSION

This paper has presented an idea for coping with ill conditioning effects which arise when standard gradient methods or nonlinear programming algorithms are applied directly to discrete optimal control problems. The gist of the idea is to devise parametric methods which do not suffer from ill conditioning effects and at the same time retain a number of the most attractive features of nonlinear programming algorithms. The result, for the case worked in detail, is seen to be a very fast, and demonstrably convergent, large step, parametric algorithm.

It is hoped that the approach presented in this paper will give rise to a greater utilization of nonlinear programming ideas in control engineering.

ACKNOWLEDGMENT

The authors wish to thank S. Winograd for his interest and constructive comments.

REFERENCES

1. M. Canon, C. Cullum, and E. Polak, Constrained minimization problems in finite dimensional spaces, J. SIAM Control 4 (1966), 528-547.

2. J. B. Plant, "An Iterative Procedure for the Computation of Optimal Controls," Ph. D. Dissertation, Dept. of Electrical Engineering, MIT, June 1965.

3. G. Zoutendijk, Methods of Feasible Directions, A Study in Linear and Non-linear Programming, Elsevier Publishing Co., Amsterdam, 1960.

4. H. W. Kuhn and A. W. Tucker, "Nonlinear Programming, Proc. of the Second Berkeley Symposium on Mathematic Statistics and Probability," pp. 481-492, Univ. of Calif. Press, Berkeley, Calif, 1951.

5. W. I. Zangwill, Applications of the Convergence Conditions, Working Paper No. 231, Univ. of Calif., Berkeley, August 1967.

Table 1. The computation of s^* satisfying $v_{\epsilon'}(s^*) = w_{\epsilon'}(s^*)$.

s_0	s_1	s_2
0.91892549×10^{0}	0.91872677×10^{0}	0.91872670×10^{0}
0.11892548×10^{0}	0.11872676×10^{0}	0.11872668×10^{0}
$0.68316957 \times 10^{-14}$	$-0.13492947 \times 10^{-1}$	$-0.13502530 \times 10^{-1}$
$0.68316951 \times 10^{-14}$	$-0.13402047 \times 10^{-1}$	$-0.13502530 \times 10^{-1}$
0.23785095×10^{0}	0.23745353×10^{0}	0.23745349×10^{0}
0.23785095×10^{0}	0.23745353×10^{0}	0.23745349×10^{0}
$0.62171410 \times 10^{-10}$	$-0.16132870 \times 10^{-1}$	$-0.16163510 \times 10^{-1}$
$0.62171410 \times 10^{-10}$	$-0.16132870 \times 10^{-1}$	$-0.16163510 \times 10^{-1}$
0.11892548×10^{0}	0.11872676×10^{0}	0.11872668×10^{0}
0.11892548×10^{0}	0.11872676×10^{0}	0.11872668×10^{0}
$v(s_0)$	$v(s_1)$	$v(s_2)$
0.59462738×10^{0}	0.59363380×10^{0}	0.59363091×10^{0}
0.59462738×10^{0}	0.59363380×10^{0}	0.59363091×10^{0}
$0.34158478 \times 10^{-13}$	$-0.67464734 \times 10^{-1}$	$-0.67573219 \times 10^{-1}$
$0.34158578 \times 10^{-13}$	$-0.67464734 \times 10^{-1}$	$-0.67573219 \times 10^{-1}$
0.11892548×10^{1}	0.11872676×10^{1}	0.11872618×10^{1}
0.11892548×10^{1}	0.11872676×10^{1}	0.11872618×10^{1}
$0.31085705 \times 10^{-9}$	$-0.80664348 \times 10^{-1}$	$-0.80796073 \times 10^{-1}$
$0.31\text{-}85705 \times 10^{-9}$	$-0.80664348 \times 10^{-1}$	$-0.80796073 \times 10^{-1}$
0.59462739×10^{0}	0.59363380×10^{0}	0.59363091×10^{0}
0.59462738×10^{0}	0.59363380×10^{0}	0.59363091×10^{0}
$c(s_0)$	$c(s_1)$	$c(s_2)$
0.35487401×10^{0}	0.36352709×10^{0}	0.36357584×10^{0}

Table 2. The optimal control sequence.

u_0	$= -0.21288059 \times 10^{-2}$
u_1	$= -0.23653399 \times 10^{-2}$
u_2	$= -0.26281554 \times 10^{-2}$
u_3	$= -0.29201728 \times 10^{-2}$
u_4	$= -0.32446364 \times 10^{-2}$
u_5	$= -0.36051516 \times 10^{-2}$
u_6	$= -0.40057241 \times 10^{-2}$
u_7	$= -0.44508046 \times 10^{-2}$
u_8	$= -0.49453384 \times 10^{-2}$
u_9	$= -0.54948205 \times 10^{-2}$
u_{10}	$= -0.61053560 \times 10^{-2}$
u_{11}	$= -0.67837290 \times 10^{-2}$
u_{12}	$= -0.75374768 \times 10^{-2}$
u_{13}	$= -0.83749742 \times 10^{-2}$
u_{14}	$= -0.93055267 \times 10^{-2}$
u_{15}	$= -0.10339474 \times 10^{-1}$
u_{16}	$= -0.11488305 \times 10^{-1}$
u_{17}	$= -0.12764784 \times 10^{-1}$
u_{18}	$= -0.14183093 \times 10^{-1}$
u_{19}	$= -0.15758992 \times 10^{-1}$
u_{20}	$= -0.17509992 \times 10^{-1}$
u_{21}	$= -0.194555546 \times 10^{-1}$
u_{22}	$= -0.21617274 \times 10^{-1}$
u_{23}	$= -0.24019194 \times 10^{-1}$
u_{24}	$= -0.26687993 \times 10^{-1}$
u_{25}	$= -0.29653326 \times 10^{-1}$
u_{26}	$= -0.32948140 \times 10^{-1}$
u_{27}	$= -0.36609045 \times 10^{-1}$
u_{28}	$= -0.40676714 \times 10^{-1}$
u_{29}	$= -0.45196350 \times 10^{-1}$
u_{30}	$= -0.50218165 \times 10^{-1}$
u_{31}	$= -0.55797960 \times 10^{-1}$
u_{32}	$= -0.61997730 \times 10^{-1}$
u_{33}	$= -0.68986360 \times 10^{-1}$
u_{34}	$= -0.76540388 \times 10^{-1}$
u_{35}	$= -0.85044856 \times 10^{-1}$
u_{36}	$= -0.94494253 \times 10^{-1}$
u_{37}	$= -0.10499356 \times 10^{0}$
u_{38}	$= -0.11665941 \times 10^{0}$
u_{39}	$= -0.12962140 \times 10^{0}$
u_{40}	$= -0.14402349 \times 10^{0}$
u_{41}	$= -0.16002560 \times 10^{0}$
u_{42}	$= -0.17780535 \times 10^{0}$
u_{43}	$= -0.19755999 \times 10^{0}$
u_{44}	$= -0.21950844 \times 10^{0}$
u_{45}	$= -0.24389356 \times 10^{0}$
u_{46}	$= -0.27098443 \times 10^{0}$
u_{47}	$= -0.30107871 \times 10^{0}$
u_{48}	$= -0.33450454 \times 10^{0}$
u_{49}	$= -0.371621269 \times 10^{0}$

FLOW CHART

EXITS

(1) Undriven response inside target set Ω. Optimal controls = 0

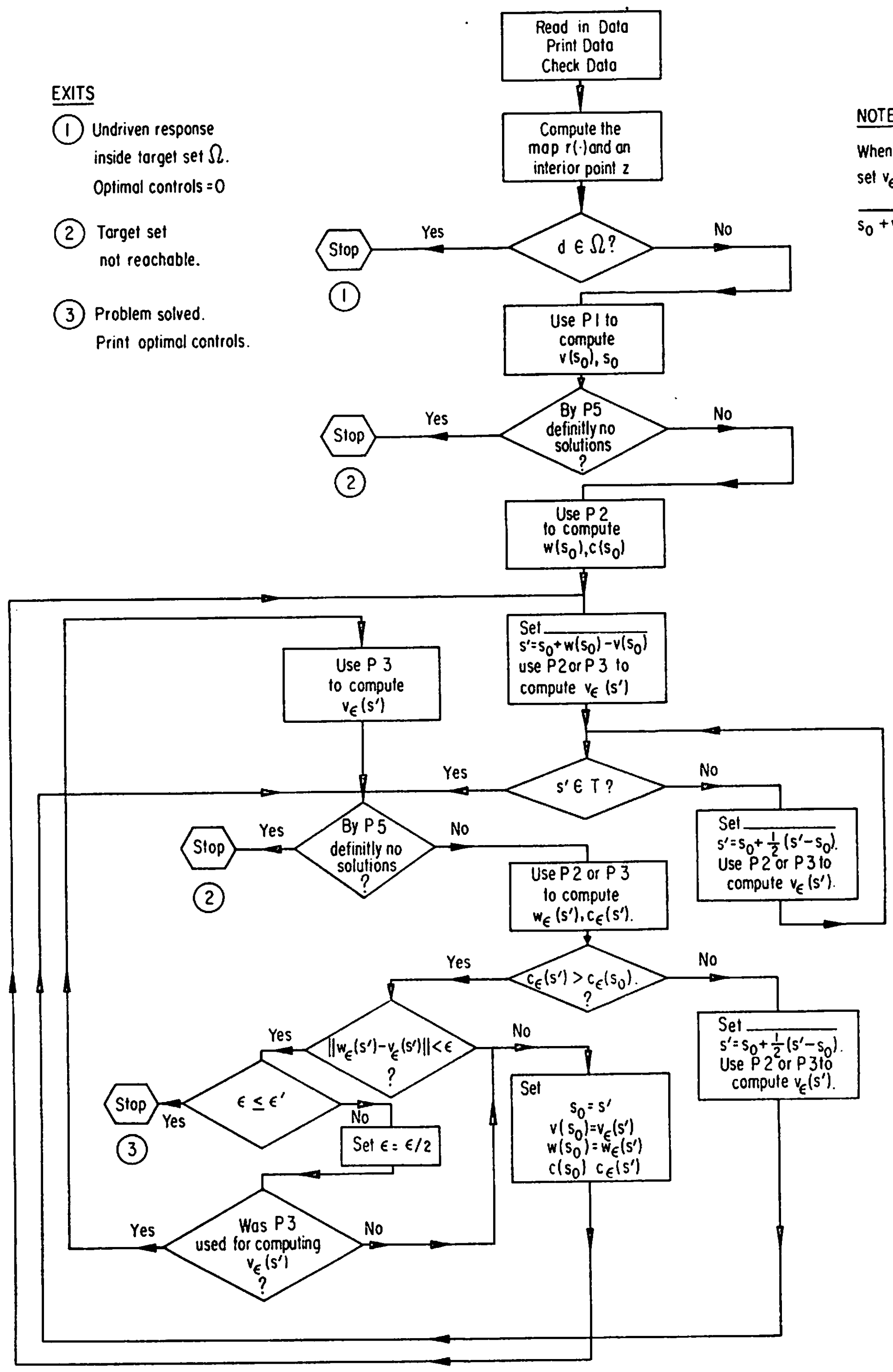
(2) Target set not reachable.

(3) Problem solved. Print optimal controls.

NOTES

When $v(s)$ is not on edge set $v_\epsilon(s) = v(s), w_\epsilon(s) = w(s), c_\epsilon(s) = c(s)$.

$$\overline{s_0 + w(s_0) - v(s_0)} = \frac{s_0 + w(s_0) - v(s_0)}{\|s_0 + w(s_0) - v(s_0)\|}$$



Read in Data
Print Data
Check Data
Compute the map r(·) and an interior point z
d ∈ Ω?
Yes
No
Stop
1
Use P1 to compute v(s0), s0
By P5 definitly no solutions ?
Yes
No
Stop
2
Use P2 to compute w(s0), c(s0)
Set s' = s0 + w(s0) - v(s0) use P2 or P3 to compute vε(s')
Use P3 to compute vε(s')
s' ∈ T ?
Yes
No
Set s' = s0 + ½(s' - s0). Use P2 or P3 to compute vε(s').
By P5 definitly no solutions ?
Yes
No
Stop
2
Use P2 or P3 to compute wε(s'), cε(s').
cε(s') > cε(s0). ?
Yes
No
Set s' = s0 + ½(s' - s0). Use P2 or P3 to compute vε(s').
||wε(s') - vε(s')|| < ε ?
Yes
No
ε ≤ ε'
Yes
No
Stop
3
Set ε = ε/2
Set s0 = s' v(s0) = vε(s') w(s0) = wε(s') c(s0) cε(s')
Was P3 used for computing vε(s') ?
Yes
No

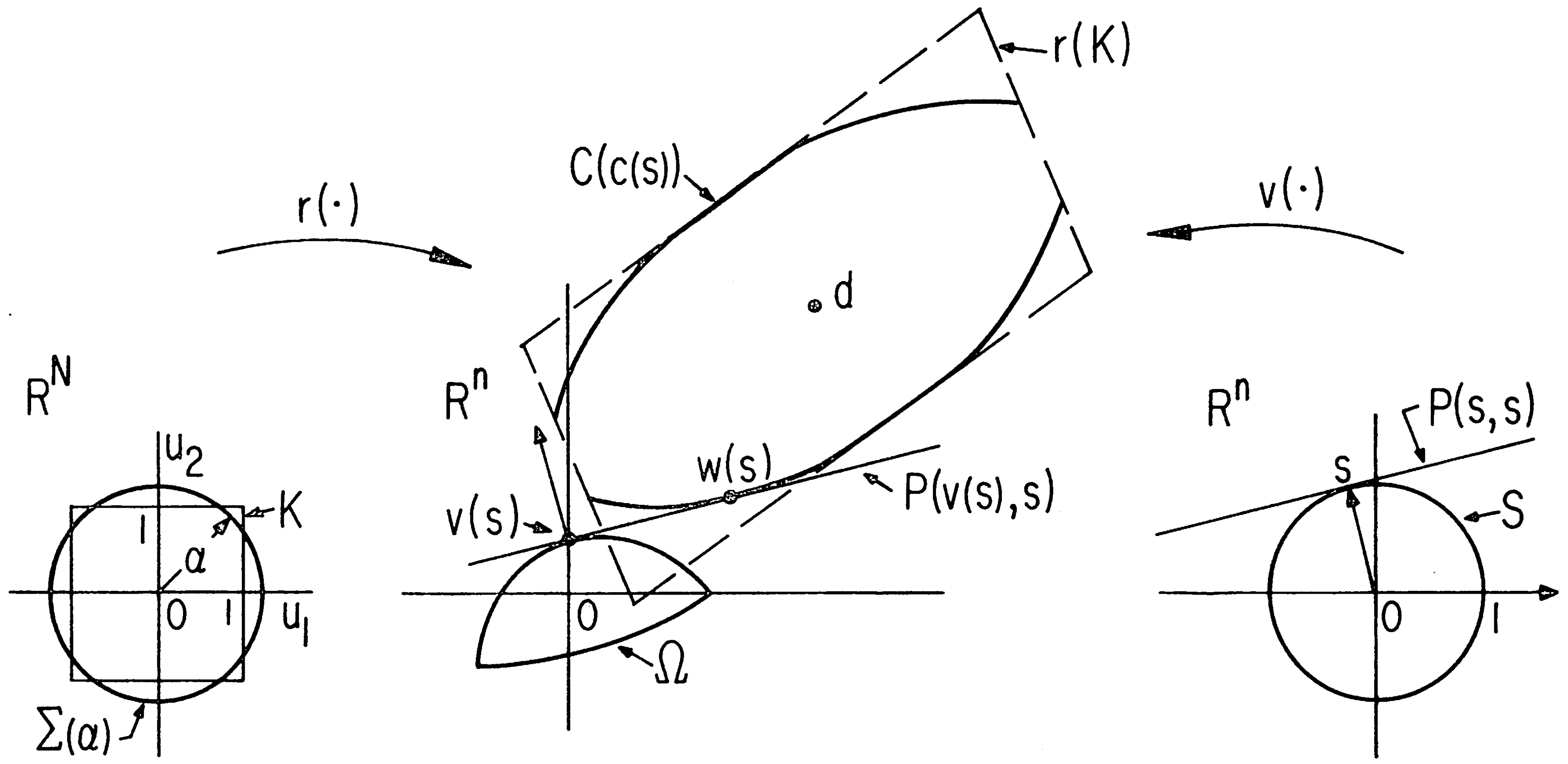


Fig. 1. The geometry of the problem.