JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 20, No. 4, DECEMBER 1976

Function-Space Quasi-Newton Algorithms for Optimal Control Problems with Bounded Controls and Singular Arcs[1]

E. R. EDGE[2] AND W. F. POWERS[3]

Communicated by T. N. Edelbaum

Abstract. Two existing function-space quasi-Newton algorithms, the Davidon algorithm and the projected gradient algorithm, are modified so that they may handle directly control-variable inequality constraints. A third quasi-Newton-type algorithm, developed by Broyden, is extended to optimal control problems. The Broyden algorithm is further modified so that it may handle directly control-variable inequality constraints. From a computational viewpoint, dyadic operator implementation of quasi-Newton methods is shown to be superior to the integral kernel representation. The quasi-Newton methods, along with the steepest descent method and two conjugate gradient algorithms, are simulated on three relatively simple (yet representative) bounded control problems, two of which possess singular subarcs. Overall, the Broyden algorithm was found to be superior. The most notable result of the simulations was the clear superiority of the Broyden and Davidon algorithms in producing a sharp singular control subarc.

Key Words. Optimal control, numerical methods, computing methods, gradient methods, quasi-Newton algorithms, bounded control problems, singular arcs.

1. Introduction

In the past few years, numerous quasi-Newton type algorithms for the solution of parameter optimization problems have been extended from Euclidean spaces to infinite-dimensional, real Hilbert spaces. Just as in

[1] This research was supported by the National Science Foundation under Grant Nos. GK-30115 and ENG 74-21618 and by the National Aeronautics and Space Administration under Contract No. NAS 9-12872.

[2] Control Systems Analyst, TRW Systems Group, Redondo Beach, California.

[3] Professor, Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan.

Euclidean space, the primary advantage in Hilbert space is the accelerated rate of convergence due to the building of second-order information while requiring only function and gradient evaluations. The primary difficulty in $L_2[t_0, t_f]$ lies in representing the infinite-dimensional H operator. Two representations, the integral kernel representation and dyadic operator representation, have been suggested and successfully implemented. The integral kernel representation requires a fixed, although large, amount of storage. The dyadic operator representation requires increasing amounts of storage as a function of the number of iterations. It will be shown that, even with this apparent disadvantage, the dyadic operator representation is superior to the integral kernel representation.

Except for the conjugate gradient and gradient methods, existing function-space methods cannot handle directly control-variable inequality constraints. Thus, applications to optimal control problems have primarily dealt with the classical Bolza problem. Since most realistic problems contain control-variable inequality constraints, it is desirable to be able to handle them directly in a computation scheme. In attempting to solve such problems, a new function-space algorithm has been generated, and two existing quasi-Newton type algorithms have been modified to allow them to handle directly the bounded control problem. The modification of the algorithms was strongly influenced by the work of Pagurek and Woodside (Ref. 1) on extending the conjugate gradient method to include bounded controls. The methods modified include the Davidon (Ref. 2), Broyden (Ref. 3), and projected gradient (Ref. 4) algorithms.

2. Algorithms

In this section, the various algorithms will be formally stated for the problem of minimizing a real functional $J(u)$, where u may be either finite-dimensional or infinite-dimensional. With u finite-dimensional, the formulas are applicable to the standard unconstrained parameter optimization problem. In a later section, the appropriate modifications for application to optimal control problems with bounded control variables will be presented.

In the listing below, each algorithm requires the specification of a starting vector u_0. The Davidon, projected gradient, and Broyden algorithms require the specification of a positive-definite, self-adjoint linear operator H_0. Also, $\langle a, b\rangle$ and $a\rangle\langle b$ will be used to denote the inner and outer dyadic products (Refs. 5–6), respectively, on the given Hilbert space. Note that, if the space is n-dimensional Euclidean space, then

$$\langle a, b\rangle = a^T b \qquad \text{and} \qquad a\rangle\langle b = ab^T,$$

where a, b are n-dimensional vectors. The inner and outer products for the optimal control problem will be defined in the next section.

Let $g(u)$ denote the gradient of J, and define the update formula by

$$u_{i+1} = u_i + \alpha_i d_i, \tag{1}$$

where d_i is the search direction vector and α_i is a scalar parameter defined by a one-dimensional search technique which minimizes J with respect to α.

Gradient Algorithm (G)

(a) Calculate the search direction $d_i = -g(u_i)$.

(b) Use Eq. (1) to calculate u_{i+1} and return to Step (a).

Conjugate Gradient Algorithm 1 (CG1, Ref. 7)

(a) Calculate the search direction[4]

$$d_i = -g(u_i) + \beta_{i-1} d_{i-1}, \tag{2}$$

where

$$\beta_{i-1} = \langle g_i, g_i \rangle / \langle g_{i-1}, g_{i-1} \rangle. \tag{3}$$

(b) Use Eq. (1) to calculate u_{i+1} and return to Step (a).

Conjugate Gradient Algorithm 2 (CG2, Ref. 8)

On the first iteration $i = 0$, (CG2) $\equiv$ (CG1). However, for $i \geq 1$, define

$$\beta_{i-1} = \langle g_i, g_i - g_{i-1} \rangle / \langle g_{i-1}, g_{i-1} \rangle \tag{4}$$

and proceed as in (CG1).

Davidon Algorithm (DAV, Refs. 2 and 9)

(a) Calculate the search direction

$$d_i = -H_i g(u_i). \tag{5}$$

(b) Use Eq. (1) to calculate u_{i+1}.

(c) Calculate

$$s_i = u_{i+1} - u_i, \tag{6}$$

$$y_i = g(u_{i+1}) - g(u_i). \tag{7}$$

(d) Update H according to the following formula:

$$H_{i+1} = H_i + (s_i \langle s_i) / \langle s_i, y_i \rangle - (H_i y_i \langle H_i y_i) / \langle y_i, H_i y_i \rangle. \tag{8}$$

(e) Return to Step (a).

[4] On the first iterate $i = 0$, define $d_i = -g(u_i)$.

Projected Gradient Algorithm (PGA, Ref. 4)
The same as DAV, except for Step (d), where H is updated according to the formula

$$H_{i+1} = H_i - (H_i y_i)\langle H_i y_i)/\langle y_i, H_i y_i\rangle. \tag{9}$$

Broyden Algorithm (BRD, Ref. 3)
The same as DAV except for Step (d), where H is updated according to the formula

$$H_{i+1} = H_i + [1 + \langle y_i, H_i y_i\rangle/\langle s_i, y_i\rangle][(s_i\rangle\langle s_i)/\langle s_i, y_i\rangle] - (s_i\rangle\langle H_i y_i)/\langle s_i, y_i\rangle$$
$$- (H_i y_i\rangle\langle s_i)/\langle s_i, y_i\rangle. \tag{10}$$

It has been shown by Dixon (Ref. 10) that, if an *exact* one-dimensional search is employed, then algorithms DAV and BRD produce the same search directions. However, in the simulations of this paper and Ref. 11, the BRD algorithm tended to give a slightly smaller performance index than the DAV algorithm in the same CPU time. This, of course, is due to the nonexactness of the search.

3. Extension to Optimal Control Problems

A motivating way of viewing the quasi-Newton methods is as a class of algorithms between the first-order (Ref. 12) and second-order (Refs. 12–15) optimal control gradient methods. The goal of a quasi-Newton algorithm is to build information about the second-variation operator without computing it explicitly, i.e., based upon gradient information only. As noted previously, in n-dimensional space, the algorithms are used to minimize a scalar-valued function $J(u)$, where u is an n-dimensional vector, the inner product is

$$\langle s, y\rangle \equiv s^T y,$$

the dyadic operator is

$$s\rangle\langle y \equiv s y^T,$$

and the H operator is an $n \times n$ matrix of scalars. Implementation of the algorithms on this type of problem is well documented in the literature. All of the algorithms described, with the exception of the (BRD) algorithm, have also been generalized to optimal control problems where g is the gradient of a functional. The primary difficulty in implementing the quasi-Newton type algorithms on optimal control problems lies in representing the infinite-dimensional H-operator.

In L_2-space, the inner product is

$$\langle s, y\rangle \equiv \int_{t_0}^{t_f} s^T y \, dt,$$

and the dyadic operator (Refs. 5–6) is

$$(s)\langle y)u \equiv \langle y, u\rangle s.$$

However, there simply is no convenient way to represent H. One way to overcome this difficulty is presented in Ref. 4 by Lasdon, where it is observed that only $H_i g_i$ (not H_i itself) is needed to compute d_i. This is also true for the Broyden algorithm. To implement the Broyden algorithm, where g is the gradient of the functional, and u, s, and y are time functions, we proceed as follows:

(i) H_0 is taken to be any positive-definite, self-adjoint operator.

(ii) H_i in Eq. (10) is expressed as a sum back to H_0. We operate on the resultant expression for H_i with g_i to obtain the following search direction:

$$d_i = -H_0 g_i - \sum_{j=0}^{i-1} [(1+\langle y_j, H_j y_j\rangle/\langle s_j, y_j\rangle)(\langle s_j, g_i\rangle/\langle s_j, y_j\rangle)s_j - (\langle H_j y_j, g_i\rangle/\langle s_j, y_j\rangle)s_j - (\langle s_j, g_i\rangle/\langle s_j, y_j\rangle)H_j y_j]. \tag{11}$$

Equation (11) requires the computation of inner products of the functions $H_i y_i$, s_i, y_i, and operating with H_0 ($H_0 = I$ being the simplest choice). The functions $(s_0, \ldots, s_{i-1})$ are available from past iterations. To compute the functions $H_i y_i$, we need only replace $-g_i$ by y_i in Eq. (11), i.e., H_i operating on y_i instead of $-g_i$. Then, for the case $i-1$,

$$H_{i-1}y_{i-1} = H_0 y_{i-1} + \sum_{j=0}^{i-2} [(1+\langle y_j, H_j y_j\rangle/\langle s_j, y_j\rangle)(\langle s_j, y_{i-1}\rangle/\langle s_j, y_j\rangle)s_j - (\langle H_j y_j, y_{i-1}\rangle/\langle s_j, y_j\rangle)s_j - (\langle s_j, y_{i-1}\rangle/\langle s_j, y_j\rangle)H_j y_j]. \tag{12}$$

Thus, $H_{i-1}y_{i-1}$ can be computed in a way requiring only inner products and operation with $H_0 = I$, as was the case for $-H_i g_i$. Note that $2i+4$ time functions must be stored after the ith iteration in order to compute the $(i+1)$th iteration, i.e.,

$$\begin{aligned} &(s_0, \ldots, s_i), && i+1 \text{ functions},\\ &(H_0 y_0, \ldots, H_{i-1} y_{i-1}), && i \text{ functions},\\ &g_i, u_{i+1}, y_{i-1}, && 3 \text{ functions}. \end{aligned} \tag{13}$$

Another approach, recently suggested by Oi, Sayama, and Takamatsu (Ref. 16), and earlier recognized by Ladd (Ref. 17), is the integral kernel representation. Consider the rank-one dyadic operator

$$(u\rangle\langle v)w = u(t)\langle v, w\rangle = u(t)\int_{t_0}^{t_f} v(s)w(s)\,ds$$

$$= \int_{t_0}^{t_f} u(t)v(s)w(s)\,ds = \int_{t_0}^{t_f} K_{uv}(t, s)w(s)\,ds, \tag{14}$$

where

$$K_{uv}(t, s) = u(t)v(s)$$

is a rank-one integral kernel operator and $u, v, w \in L_2[t_0, t_f]$. This leads to the explicit representation of H and the rank-one correction dyadic operators as integral kernels. $(u\rangle\langle v)$ is represented as

$$K_{uv}(t, s) = u(t)v(s).$$

H_i is represented as $H_i(t, s)$.

This formulation allows the direct application of Eqs. (8)–(10), where

$$d_i = -\int_{t_0}^{t_f} H_i(t, s)g_i(s)\,ds. \tag{15}$$

Only one function of two variables $H_i(t, s)$ along with four functions of one variable $u_i, u_{i+1}, g_i, g_{i+1}$ must be stored to compute H_{i+1} and the $(i+1)$th iteration. Computational aspects of the two representations will be discussed later.

4. Bounded Controls

We shall now define the basic optimal control problem and then discuss the problem of implementing the quasi-Newton algorithms. The interpretation of the above formulas and operations is more motivating in an optimal control setting.

The optimal control problem of interest is a Bolza problem with control constraints as follows:

$$\text{minimize} \qquad J(u) = \phi(x_f) + \int_{t_0}^{t_f} L(t, x, u)\,dt, \tag{16}$$

$$\begin{aligned} \text{subject to} \qquad & \dot{x} = f(t, x, u), \qquad x(t_0) = x_0, \qquad x \equiv k\text{-vector}, \\ & |u^i| \le c^i, \qquad i = 1, \ldots, n, \qquad u \equiv n\text{-vector}, \\ & t_0, t_f \text{ specified}. \end{aligned} \tag{17}$$

If terminal conditions are present, they are included in the term $\phi(x_f)$ in Eq. (11) by the method of penalty functions. Also, the algorithms apply to problems with free t_f, but the examples of this paper are for fixed t_f. Since only gradient and function evaluations are required for the quasi-Newton methods, we shall first outline the gradient method for optimal control problems, and then discuss the modifications for a quasi-Newton method.

In all of the algorithms, the following equations are required:

$$H = L + \lambda^T f(t, x, u), \tag{18}$$

$$\dot{\lambda} = -\partial H/\partial x, \qquad \lambda(t_f) = \partial\phi/\partial x_f, \tag{19}$$

$$g(u) = \partial H/\partial u. \tag{20}$$

The function H above is the Hamiltonian, which is not to be confused with the operator H_i of the algorithms, and $g(u) = \partial H/\partial u$ is the function space gradient. The usual implementation of the standard gradient method is shown in Fig. 1, where

$$u_{i+1}(t) = u_i(t) - \alpha_i(\partial H/\partial u). \tag{21}$$

Note that the subscripts indicate the iteration number for the respective vectors; this allows less cumbersome writing of the quasi-Newton formulas.

The optimal control for the problem defined by Eqs. (16)–(17) will, in general, consist of a sequence of component intervals with interior control ($|u^i| < c^i$) and bounded control ($|u^i| = c^i$). On each subarc, the following conditions must be satisfied:

$$u^i = c^i \Rightarrow \partial H/\partial u^i \leq 0, \tag{22}$$

$$-c^i < u^i < c^i \Rightarrow \partial H/\partial u^i = 0, \tag{23}$$

$$u^i = -c^i \Rightarrow \partial H/\partial u^i \geq 0. \tag{24}$$

We shall now discuss how bounded control variables are treated directly in the standard gradient method, since the same basic idea is employed in the quasi-Newton methods.

As new controls are generated by varying α in Eq. (20), they may violate the inequality

$$|u^i| \leq c^i.$$

On these intervals, u^i is truncated such that, if $u^i > c^i$, u^i is set equal to c^i and, if $u^i < -c^i$, u^i is set equal to $-c^i$. After truncation the cost associated with the given α is calculated. In this way, the saturation region may change from iteration to iteration, and costs are only computed for realizable controls.

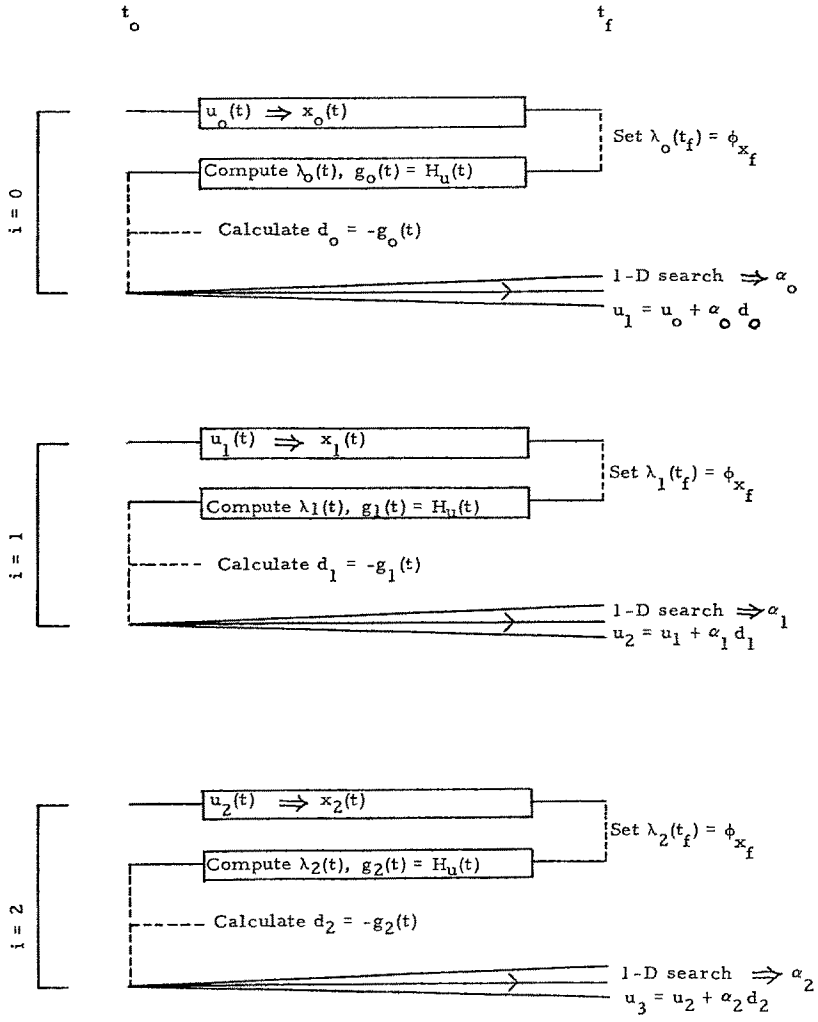

Fig. 1. Flow of the standard gradient method.

The implementation of the quasi-Newton type algorithm on unbounded control problems is shown in Figs. 2–3. As in Fig. 1, the subscripts indicate the iteration number, and Eq. (8) implies the Davidon algorithm, Eq. (9) implies the projected gradient algorithm, and Eqs. (11)–(12) imply the Broyden algorithm.

As the iteration proceeds, the number of functions stored increases. The computation time per iteration will also increase because of more inner product evaluations in the updating formulas for Hy and d. To overcome

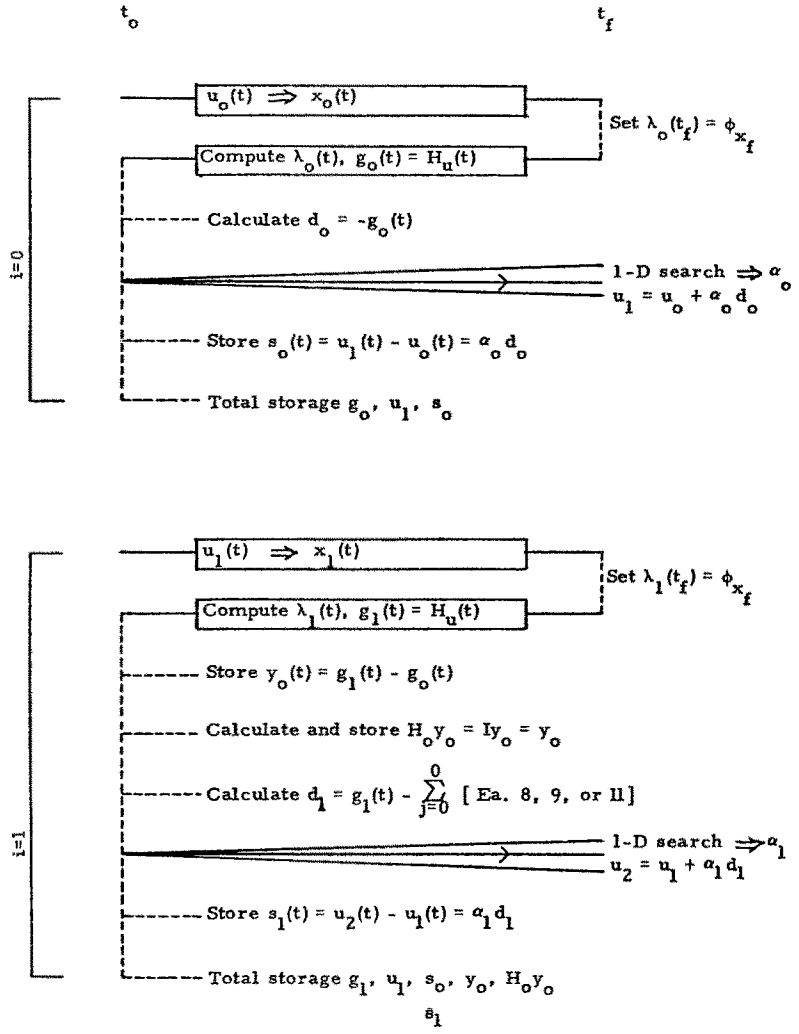

Fig. 2. Flow of the function-space quasi-Newton algorithms for $H_0 = I$ and $i = 0, 1$.

this difficulty, the algorithm is restarted with a pure gradient step when $i = q$, where q is some predetermined integer. This problem will be discussed in a later section.

To apply the quasi-Newton type algorithms to the bounded control problem, a modification to the updating formula is required. In the interior portion of the control, we wish to build second-order information, while second-order information on the bounded portion of the control is of little use. Thus, the quasi-Newton formulas should concentrate on the interior

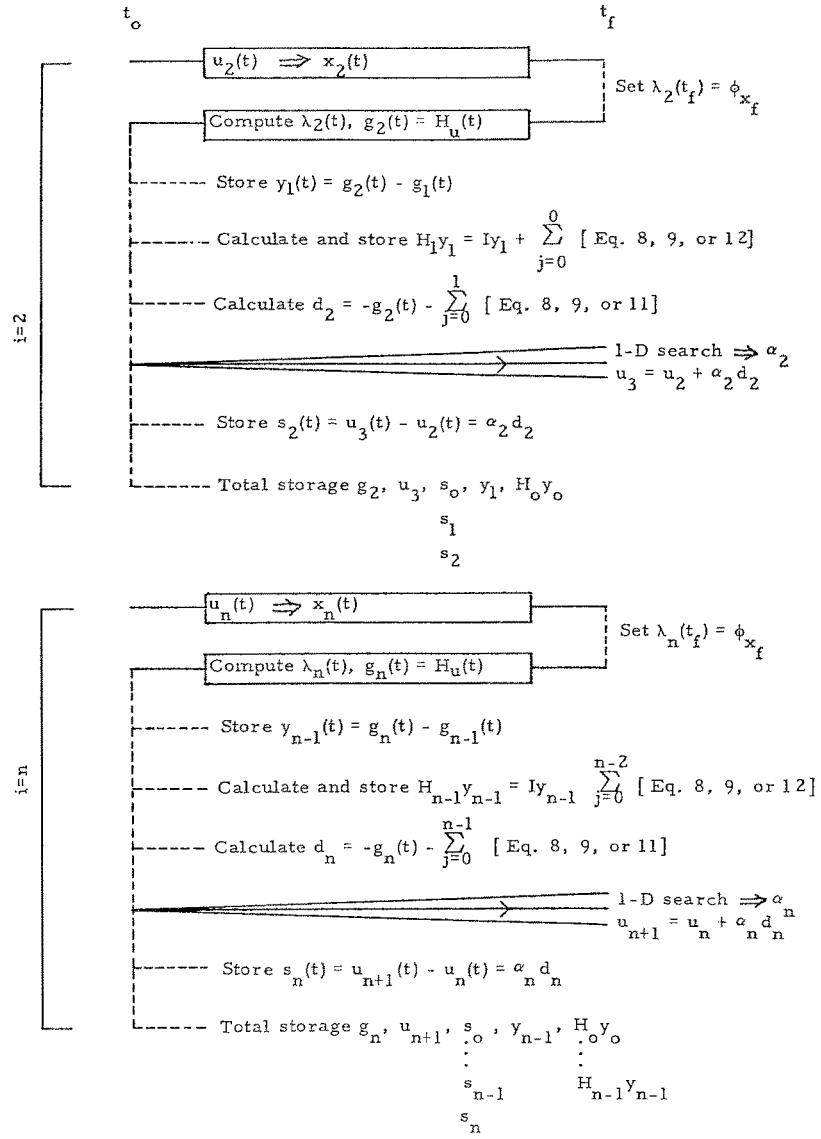

Fig. 3. Flow of the function-space quasi-Newton algorithms for $H_0 = I$ and $i = 2, \ldots, n, \ldots$.

controls, and a standard gradient formula can be used on the bounded portion of the control.

As with the gradient algorithm, as new controls are generated, they are truncated before calculating the associated cost. A saturation function $w_i(t)$,

identical to that of Pagurek and Woodside (Ref. 1), is defined. This saturation function is set equal to zero when the control is on the boundary and is set equal to unity on the interior. The saturation function is then used in the following way to compensate for our lack of freedom in choosing the control on the boundary. Instead of using g, y, Hy in the formulas for calculating the search direction and updating Hy, we use wg, wy, wHy. We know that $g = 0$ on the interior portion of the optimal control, and this is where we wish to build second-order information. On the region of saturation, $g \neq 0$ (in general) and the y's (or Δg's) should not contribute to the inner products in the updating formulas. It is not necessary to apply the saturation function to s because, on the saturation region, $s = \Delta u$ will already be zero.

The quasi-Newton algorithm for bounded control problems differs from the algorithm for unbounded control problems in the following ways:

(i) As the one-dimensional search seeks the best α, the associated controls are truncated before the associated cost is calculated.

(ii) A saturation function $w_i(t)$ is generated after each iteration.

(iii) wg, wy, wHy are used in the updating formulas and in the calculation of the search direction.

(iv) On the bounded portion of the control, the search direction is chosen to be $-\partial H/\partial u$. This allows the control to become interior if Eqs. (22) and (24) are not satisfied.

5. Computer Implementation Considerations

Dyadic and Kernel Representations. Since dyadic and integral kernel representations will generate the same search directions, given the same H_0, the choice of representations must be determined by practical considerations, such as storage required, number of inner products required, and previous computational experience. First, consider the total amount of storage required by each representation.

Numerically, functions of one variable are stored pointwise at m mesh points on $t \in [t_0, t_f]$. $H(t, s)$, a function of two variables, will be stored at $m \times m$ grid points on $t, s \in [t_o, t_f]$. In order to calculate intermediate points required for inner product evaluations, some form of interpolation is used. Let n equal the number of controls, i be the iteration number, and q be the reset parameter. We have:

$$\begin{aligned}
\text{dyadic storage} &= (2i+4)n \text{ time functions} \\
&= (2i+4)nm \text{ storage locations;} \\
\text{kernel storage} &= (4n \text{ time functions}) \text{ and } H(t, s) \\
&= 4nm + n(n+1)m^2/2 \text{ storage locations.}
\end{aligned}$$

The amount of storage required for dyadic representation is a function of i and does increase from iteration to iteration; however, the fixed storage required by the kernel representation is a function of $(nm)^2$ and is very large. The breakeven point, where the storage required by each representation is equal, occurs at

$$i = (n+1)m/4 \tag{25}$$

Thus for a typical mesh ($m = 100$) and only one control ($n = 1$), the breakeven point is the 50th iteration; for two controls ($n = 2$), the breakeven point is the 75th iteration. Computational experience (Refs. 4 and 18) indicates that to restart with a gradient step every 3 to 5 iterations will, for many problems, improve the convergence rate. In these cases, the dyadic representation is superior, because of the much smaller amount of storage required.

The quasi-Newton algorithms have been implemented on a relatively high-dimension space-shuttle ascent, trajectory optimization problem. During the initial testing of the program on the University of Michigan IBM 360/67 virtual memory computer, all storage was done in fast core memory. However, it was found that core storage was exceeded when the program was first run on the Johnson Space Center UNIVAC 1108 computer, which is not a virtual memory machine. To overcome this difficulty, drum storage was used to store the large amount of information needed by the quasi-Newton algorithms. This reduced the amount of core storage required allowing the program to fit on the UNIVAC 1108 computer. Upon running the modified program on the IBM computer, a considerable savings was realized in reduced virtual memory charges. It was also found that no significant increase in the amount of CPU time was incurred. There are two reasons for this.

(i) On high-dimension problems with relatively long integration intervals, only a small percentage of the CPU time is involved in the calculation of the search direction. Most of the CPU time is spent integrating the equations of motion. On each iteration, a forward integration and a backward integration are required to determine the gradient and a number of cost evaluations also requiring forward integrations are performed by the one-dimensional search.

(ii) The updating equation for $H_i y_i$ and the equation for d_i are summations which require inner products of the stored functions in the same sequence as they were generated and stored. Assume that $H_{i-1}y_{i-1}$ and d_i are to be calculated. $H_0 y_0$ through $H_{i-2}y_{i-2}$ are stored in a file as shown in Fig. 4.

At the end of the last iteration, the file has been rewound. The updating equations for $H_{i-1}y_{i-1}$ will read $H_0y_0, H_1y_1, \ldots, H_{i-2}y_{i-2}$ in order, calculate

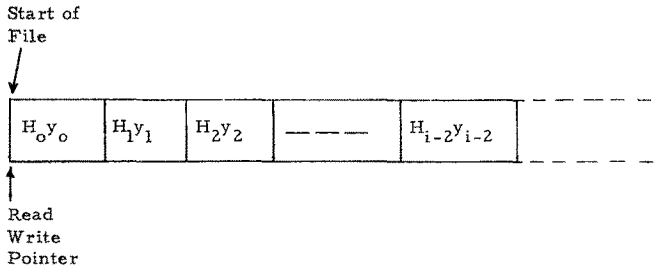

Fig. 4. File storage arrangement.

$H_{i-1}y_{i-1}$, then write $H_{i-1}y_{i-1}$ onto the file and rewind. Concurrently, the equation for d_i employs the Hy functions. The files in which Hy and s are stored need only be rewound once on a given iteration, and no forward or back spacing is required. Even if tape is employed as the storage medium (as opposed to fast-core storage), the increase in computer time would be small. When drum storage is used, the increase in computer time is insignificant. Thus, there is no need to restart with a gradient step because of limited storage.

Another measure of the desirability of one representation versus the other is the number of inner products required as a function of the iteration number. On each iteration, the following inner products must be calculated when implementing the dyadic representation:

$$\begin{aligned}
&\langle s_j, y_j\rangle, && (i-1)n,\\
&\langle y_j, H_jy_j\rangle, && (i-1)n,\\
&\langle s_j, g_i\rangle, && n\sum_{K=1}^{i} K,\\
&\langle H_jy_j, g_i\rangle, && n\sum_{K=1}^{i} K,\\
&\langle s_j, y_{i-1}\rangle, && n\sum_{K=1}^{i-1} K,\\
&\langle H_jy_j, y_{i-1}\rangle, && n\sum_{K=1}^{i-1} K.
\end{aligned} \tag{26}$$

Using the identity

$$\sum_{K=1}^{i} K = i(i+1)/2,$$

we have

$$\langle \text{dyadic} \rangle = (i-1)n + (i-1)n + ni(i+1) + n(i-1)i$$
$$= 2n(i^2+i-1) \qquad \text{for } i>1. \tag{27}$$

The integral kernel representation requires the following inner product evaluations:

$$\begin{aligned} &H_i y_i, && n^2 mi, \\ &\text{correction to } H_i, && 2ni, \\ &d_i = H_i g_i, && n^2 mi. \end{aligned} \tag{28}$$

Therefore,

$$\langle \text{kernel} \rangle = 2n^2 mi + 2ni = 2in(nm+1). \tag{29}$$

The breakeven point, where the number of inner products required by each representation is equal, occurs when

$$\begin{aligned} 2n(i^2+i-1) &= 2in(nm+1), \\ i^2 - nmi - 1 &= 0, \\ i &\approx nm. \end{aligned} \tag{30}$$

Thus, for a typical mesh ($m=100$) and only one control ($n=1$), the breakeven point is the 100th iteration; for two controls ($n=2$), the breakeven point is the 200th iteration. Now, assume that each method is restarted with a gradient step every qth iteration. The number of inner products required by the integral kernel representation is not changed. The number of inner products required by the dyadic representation becomes

$$\langle \text{dyadic} \rangle = 2n(q^2+q+1)i/q, \qquad \text{for } i = q, 2q, 3q, \ldots. \tag{31}$$

The number of inner products required by each representation increases linearly in i; therefore, consider the ratio

$$\langle \text{kernel} \rangle / \langle \text{dyadic} \rangle = [2in(nm+1)]/[2n(q^2+q-1)(i/q)], \tag{32}$$

or

$$\langle \text{kernel} \rangle = [nm + 1/(q+1(1/q))] \langle \text{dyadic} \rangle. \tag{33}$$

For $m=100$, $n=1$, we have the results shown in Table 1.

Table 1

q	⟨kernels⟩/⟨dyadic⟩
2	40.4
4	21.3
6	14.8
8	11.4
10	9.3
25	3.9

Even for $q = 25$, the integral kernel representation will require almost four times as many inner product evaluations as the dyadic representation per iteration. Note that all calculations are for one control ($n = 1$). For larger n or a finer mesh ($m > 100$), the dyadic representation becomes even more attractive.

Inner Product Calculations. In $L_2[t_0, t_f]$, the inner product is a quadrature:

$$\langle u, v\rangle = \int_{t_0}^{t_f} u^T v \, dt, \tag{34}$$

where u and v are stored pointwise. If it is assumed that the stored functions are linear between storage locations, the evaluation of the inner product reduces to a summation. Consider the interval t_1 to t_2, let $T = t - t_1$ and $\Delta t = t_2 - t_1$; then, on $[t_1, t_2]$,

$$u(t) = at + b, \qquad v(t) = \alpha t + \beta,$$

where

$$\begin{aligned} a &= (u_2 - u_1)/\Delta t, & \alpha &= (v_2 - v_1)/\Delta t \\ b &= u_1, & \beta &= v_1. \end{aligned} \tag{35}$$

The inner product of the functions between t_1 and t_2 is

$$\begin{aligned} \langle u, v\rangle_{t_1,t_2} &= \int_0^{\Delta t} (at + b)(\alpha t + \beta)\, dt \\ &= \int_0^{\Delta t} [a\alpha t^2 + (\alpha\beta + \alpha b)t + b\beta]\, dt \\ &= (a\alpha/3)\,\Delta t^3 + [(a\beta + \alpha b)/2]\,\Delta t^2 + (b\beta)\,\Delta t, \end{aligned} \tag{36}$$

and the total inner product is

$$\langle u, v\rangle_{t_0,t_f} = \sum_{i=0}^{n-1} \langle u, v\rangle_{t_i,t_{i+1}}. \tag{37}$$

It was found that this method of evaluating inner products is considerably faster than higher-order quadrature formulas and that the convergence rates of the algorithms do not suffer.

6. Numerical Study

In order to obtain some idea of the relative merit of the modified quasi-Newton algorithms, a controlled study was conducted. Three relatively simple bounded-control problems with known analytical solutions were chosen. Two of the three problems have both singular and nonsingular subarcs. The third problem has an optimal bang–bang control with seven switches.

In addition to the three quasi-Newton algorithms, the problems were also solved using gradient and conjugate gradient algorithms. Terminal constraints were handled with quadratic penalty terms. A quadratic curve fit is used in the one-dimensional search. In all cases, a gradient step is taken every sixth iteration ($q=6$). Only the gradient, conjugate gradient, and Broyden algorithms are illustrated. In all three cases, the control and gradient functions were stored every 0.01 sec and were assumed to be piecewise linear.

Problem 6.1. This problem is taken from Ref. 19:
minimize,

$$J=\frac{1}{2}\int_0^{2.985}(x_2^2-x_1^2)\,dt,$$

subject to

$$\dot{x}_1=x_2, \qquad x_{10}=0, \qquad x_{1f}=0.065,$$
$$\dot{x}_2=u, \qquad |u|\leq 1, \qquad x_{20}=1, \qquad x_{2f}=-1.336.$$

The optimal solution is

$$u=\begin{cases}-\sin t, & t\in[0,3\pi/4),\\ -1, & t\in(3\pi/4,2.085].\end{cases}$$

The optimal cost is $J^*=0.022386$. The optimal control is shown in Fig. 5. The augmented cost functional is

$$J'=10(x_{1f}-0.065)^2+10(x_{2f}+1.336)^2+J.$$

This problem has a discontinuous optimal control consisting of a singular arc followed by a nonsingular arc (Fig. 5). On this problem, all of the algorithms performed considerably better than the gradient method, as

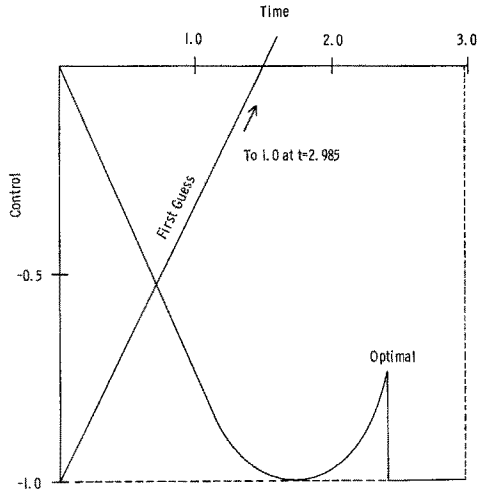

Fig. 5. Initial control estimate and the optimal control for Problem 6.1.

noted in Fig. 6 and Table 2. The Broyden algorithm obtained the least cost and approximates the optimal control more closely than the other methods (see Figs. 7–9). Note that, on this problem, the conjugate gradient method also had very good performance.

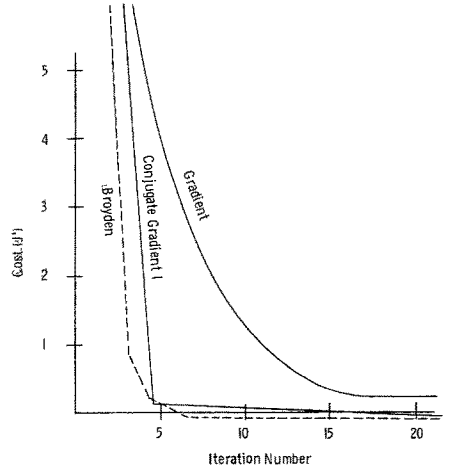

Fig. 6. Performance index vs iteration number for Problem 6.1.

Table 2. Final cost for Example 6.1.

Method	Final cost	Number of iterations	CPU (sec)
Broyden	−0.00294	48	78.4
Davidon	−0.00289	63	95.6
Conjugate gradient 1	−0.00244	72	75.1
Conjugate gradient 2	−0.00241	67	81.2
Projected gradient	−0.00100	306	456.1
Gradient	0.03686	385	366.7

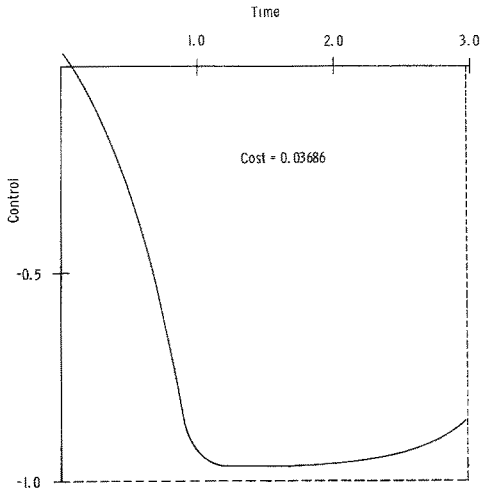

Fig. 7. Control obtained by the gradient method for Problem 6.1.

Problem 6.2. This problem is taken from Ref. 1:
minimize

$$J=\int_0^2 x^2\,dt,$$

subject to

$$\dot{x}=u, \qquad |u|\le 1, \qquad x_0=1.0, \qquad x_2=0.5.$$

The optimal solution is

$$u=\begin{cases}-1, & t\in[0,1),\\ 0, & t\in(1,3/2),\\ +1, & t\in(3/2,2].\end{cases}$$

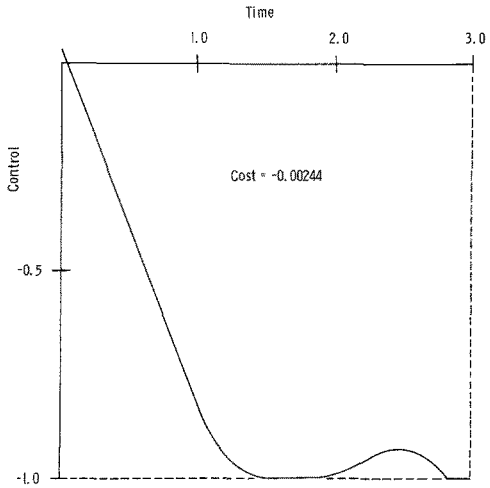

Fig. 8. Control obtained by the conjugate gradient method for Problem 6.1.

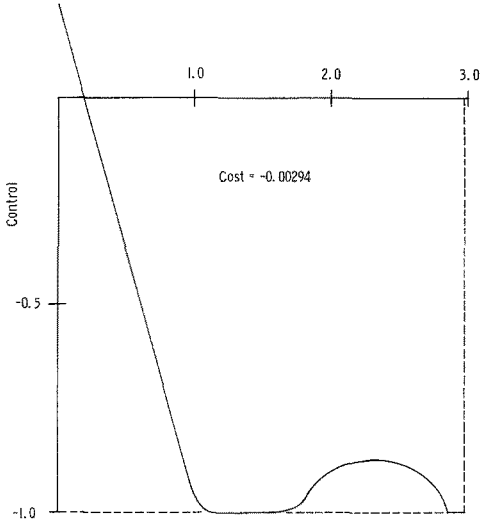

Fig. 9. Control obtained by the Broyden method for Problem 6.1.

The optimal cost is $J^* = 0.375$. The optimal control is shown in Fig. 10. The augmented cost functional is

$$J' = 50(x_f - \tfrac{1}{2})^2 + \int_0^2 x^2 \, dt.$$

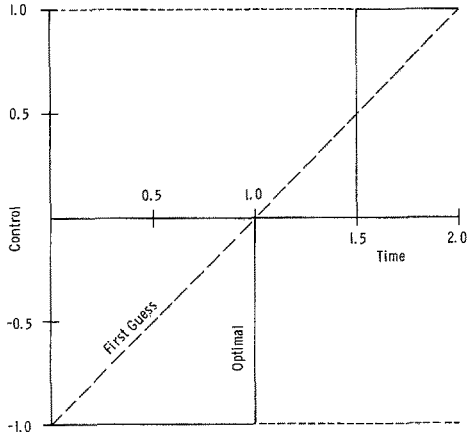

Fig. 10. Initial control estimate and optimal control for Problem 6.2.

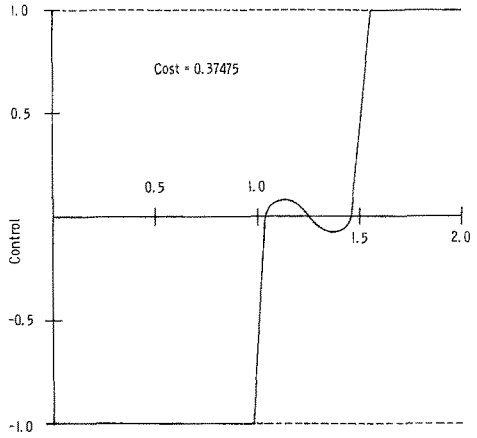

Fig. 11. Control obtained by the Broyden method for Problem 6.2.

This problem has a discontinuous optimal control consisting of three subarcs, one of which is singular. Note that the Broyden method (Fig. 11) shows *action* in the neighborhood of the singular arc, whereas the other methods do not (Figs. 12–13). It is also interesting to compare Fig. 11 with Fig. 2 of Ref. 1, which illustrates the optimal control for a second-order method requiring considerably more computation than the Broyden

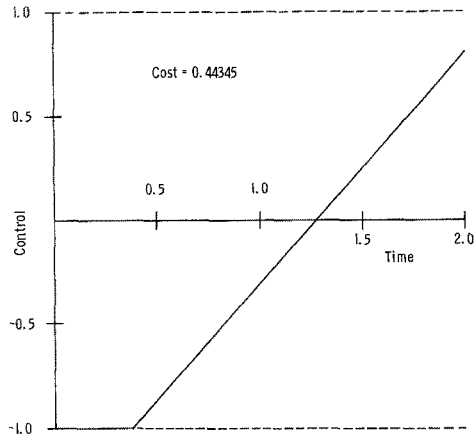

Fig. 12. Control obtained by the gradient method for Problem 6.2.

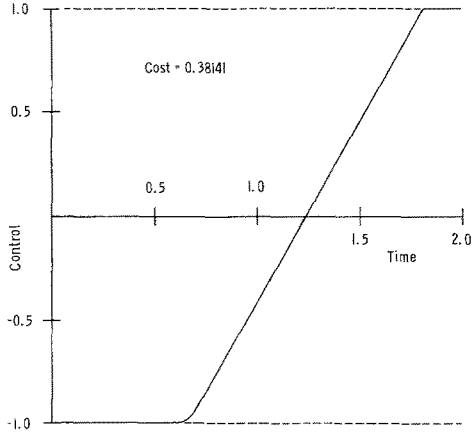

Fig. 13. Control obtained by the conjugate gradient method for Problem 6.2.

method. The controls in the two figures are similar and obtain approximately the same value of the performance index. This indicates that the Broyden method is approximating the second variation operator, while requiring only first-order information. The performance of the methods is presented in Fig. 14 and Table 3.

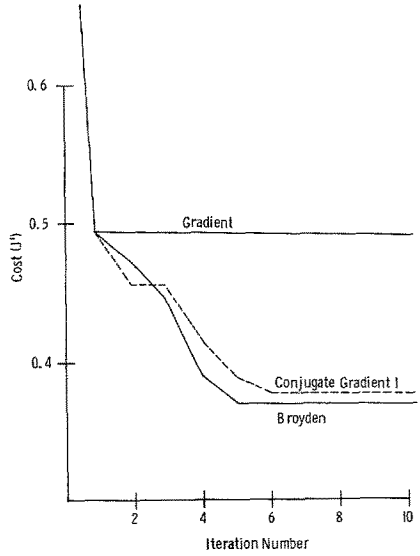

Fig. 14. Performance index vs iteration number for Problem 6.2.

Table 3. Final cost for Example 6.2.

Method	Final cost	Number of iterations	CPU (sec)
Broyden	0.37475	55	75.1
Davidon	0.37479	63	91.2
Conjugate gradient 1	0.38012	65	79.6
Conjugate gradient 2	0.38141	68	74.7
Projected gradient	0.43672	283	388.2
Gradient	0.44345	390	310.2

Problem 6.3. This problem is taken from Ref. 20: minimize

$$J = x_f^T x_f, \qquad t_0 = 0, \qquad t_f = 4.2, \qquad |u| \leq 1,$$

subject to

$$\dot{x}_1 = -0.5x_1 + 5x_2, \qquad x_{10} = 10,$$
$$\dot{x}_2 = -5x_1 - 0.5x_2 + u, \qquad x_{20} = 10,$$
$$\dot{x}_3 = -0.6x_3 + 10x_4, \qquad x_{30} = 10,$$
$$\dot{x}_4 = -10x_3 - 0.6x_4 + u, \qquad x_{40} = 10.$$

The optimal cost is $J^* = 0.996$. The optimal control is shown in Fig. 15.

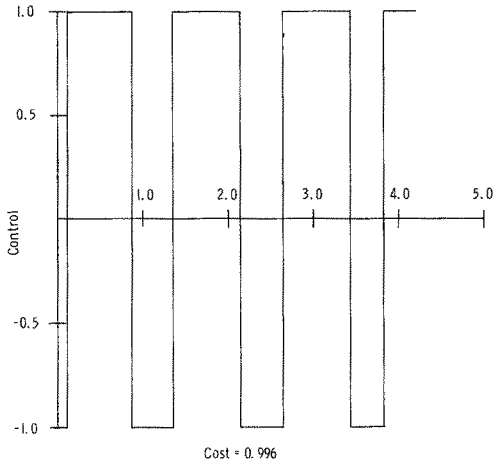

Fig. 15. Optimal control for Problem 6.3.

The optimal control for this problem (Fig. 15) has no interior subarcs. For the simulations, the initial control is $U_0(t) = 0$ and the associated cost is $J(U_0) = 4.293$. For all of the algorithms, major reduction in cost is realized only on the first iteration, which is a steepest descent or gradient step; in this iteration, the control changes from totally interior to totally bang–bang (Fig. 16). The final cost is $J(U_f) = 1.0138$. Since there are no interior subarcs after the first iteration, all of the algorithms reduce to essentially the gradient method, i.e., the various inner products required by the conjugate gradient and quasi-Newton methods are computed only on interior subarcs.

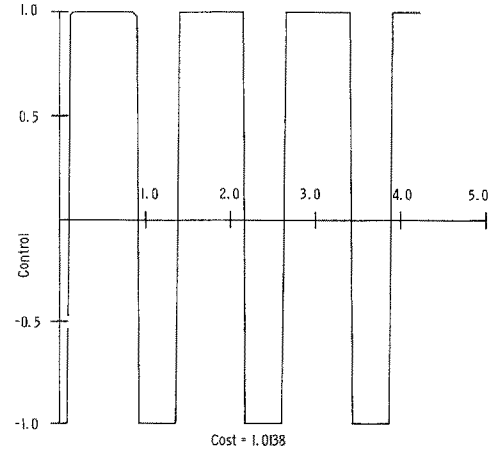

Fig. 16. Control obtained by all methods for Problem 6.3.

7. Summary

A parameter optimization scheme due to Broyden is extended to a form which is applicable to bounded optimal control problems. Several other quasi-Newton algorithms are also extended to apply to bounded control problems. These quasi-Newton algorithms, along with two conjugate-gradient algorithms and a pure gradient algorithm, are applied to two simple optimal control problems which contain both singular and non singular subarcs. The examples indicate that the Broyden and Davidon algorithms approximate the singular subarc portions of the trajectories more accurately than the gradient and conjugate gradient methods. A bang–bang control example problem was also simulated. The methods performed well on this problem, yielding a control history close to the optimal.

An alternate way of treating the bounded control is to transform the problem into an unconstrained problem, e.g., if $|u| \leq 1$, then define $u = \sin \tilde{u}$. The three examples in Section 6 were simulated with this artifice, and it was found that the direct method had a more rapid rate of convergence and produced sharper control histories. The $u = \sin \tilde{u}$ transformation also introduces oscillations about bounded control subarcs in the $\tilde{u}$-space. Because of this oscillatory nature, the artifice appears to be less effective if the solution tends to be bang–bang with a number of switches (e.g., Problem 6.3).

The choice of integral kernel or dyadic operator representation of the quasi-Newton algorithms is shown to depend not only upon storage considerations but also upon the number of inner products required. The dyadic representation appears to be superior, since less storage is required and far fewer inner product evaluations are necessary.

References

1. PAGUREK, B., and WOODSIDE, C. M., *The Conjugate Gradient Method for Optimal Control Problems with Bounded Control Variables*, Automatica, Vol. 4, Nos. 5/6, 1968.
2. HORWITZ, L. B., and SARACHIK, P. E., *Davidon's Method in Hilbert Space*, SIAM Journal on Applied Mathematics, Vol. 16, No. 4, 1968.
3. BROYDEN, C. G., *The Convergence of a Class of Double-Rank Minimization Algorithms, 2, The New Algorithm*, Journal of the Institute of Mathematics and Applications, Vol. 6, No. 3, 1970.
4. LASDON, L. S., *Conjugate Direction Methods for Optimal Control*, IEEE Transactions on Automatic Control, Vol. AC-15, No. 2, 1970.
5. PORTER, W. A., *Modern Foundations of Systems Engineering*, The Macmillan Company, New York, New York, 1966.
6. FRIEDMAN, B., *Principles and Techniques of Applied Mathematics*, John Wiley and Sons, New York, New York, 1966.

7. LASDON, L. S., MITTER, S. K., and WAREN, A. D., *The Conjugate Gradient Method for Optimal Control Problems*, IEEE Transactions on Automatic Control, Vol. AC-12, No. 2, 1967.
8. KLESSIG, R., and POLAK, E., *Efficient Implementations of the Polak–Ribière Conjugate Gradient Algorithm*, SIAM Journal on Control, Vol. 10, No. 3, 1972.
9. TRIPATHI, S. S., and NARENDRA, K. S., *Optimization Using Conjugate Gradient Methods*, IEEE Transactions on Automatic Control, Vol. AC-15, No. 2, 1970.
10. DIXON, L. C. U., *Variable Metric Algorithms; Necessary and Sufficient Conditions for Identical Behavior and Sufficient Conditions for Identical Behavior on Nonquadratic Functions*, Nottingham Optimization Centre, Nottingham, England, Report No. NOC-TR26, 1971.
11. POWERS, W. F., *A Crude-Search Davidon-Type Technique with Application to Shuttle Optimization*, AIAA Journal of Spacecraft and Rockets, Vol. 10, No. 11, 1973.
12. BRYSON, A. E. JR., and HO, Y. C., *Applied Optimal Control*, Blaisdell Publishing Company, Waltham, Massachusetts, 1969.
13. KELLEY, H. J., UZZELL, B. R., and MCKAY, S. S., *Rocket Trajectory Optimization by a Second-Order Numerical Technique*, AIAA Journal, Vol. 7, No. 5, 1969.
14. MITTER, S. K., *Successive Approximation Methods for the Solution of Optimal Control Problems*, Automatica, Vol. 3, Nos. 3/4, 1966.
15. JACOBSON, D. H., and MAYNE, D. Q., *Differential Dynamic Programming*, American Elsevier Publishing Company, New York, New York, 1970.
16. OI, K., SAYAMA, H., and TAKAMATSU, T., *Computational Schemes of the Davidon–Fletcher–Powell Method in Infinite-Dimensional Space*, Journal of Optimization Theory and Applications, Vol. 12, No. 5, 1973.
17. LADD, H. O., *A Unified Theory for Constrained Minimization on Hilbert Space*, Massachusetts Institute of Technology, PhD Thesis, 1968.
18. PIERSON, B. L., and RAJTORA, S. G., *Computational Experience with the Davidon Method Applied to Optimal Control Problems*, IEEE Transactions on System Science and Cybernetics, Vol. SSC-6, pp. 240–242, 1970.
19. MCDANELL, J. P., and POWERS, W. F., *Necessary Conditions for the Joining of Optimal Singular and Nonsingular Subarcs*, SIAM Journal on Control, Vol. 9, No. 2, 1971.
20. JACOBSON, D. H., *Differential Dynamic Programming Methods for Solving Bang–Bang Control Problems*, IEEE Transactions on Automatic Control, Vol. AC-13, No. 6, 1968.