The Control of Nonlinear Systems VI. Computational Use of Nonlinear Programming

J. V. FLYNN, JR. and LEON LAPIDUS

Princeton University, Princeton, New Jersey

In previous papers in this series (*12*) the main theme has been the use of various computational techniques for solving optimal control problems. Another paper (*17*) also pointed out the utility of mapping certain control problems into an equivalent linear programming (LP) format and using easily available computer programs to solve the original control problems.

In the present paper we wish to point out the computational feasibility of mapping nonlinear control problems into a nonlinear programming (NLP) format and solving the resulting new problem in control space. As shown the method is quite versatile with one computer program being able to handle the control problem with or without control and state constraints. The defect in the method is the amount of computer storage required. Because the problem is solved all at one time, the number of variables associated with the number of time steps may become excessive. Nevertheless the method is convenient, quick, handles a wide variety of cases and is quite competitive with most other methods. As such it makes a most inviting computational device.

DEFINITIONS AND LITERATURE

Table 1 presents a specification of the optimal control problem (*4*, *13*) and the NLP problem (*11*). In the former case there are many possible approaches to the solution of the stated problem. These include the maximum principle, dynamic programming, and whole classes of iterative techniques (*3*, *5*, *8*, *13* to *15*). At this time we do not wish to detail the defects and advantages to each procedure; such items as the difficulty of including the constraints of Equations (4) and (5), the need to solve two-point boundary value problems, and dimensionality considerations are, however, a few of the serious problems which may be encountered, with these methods.

When $I(\mathbf{z})$ in (6) is linear in the z_i, the problem becomes one of LP and the work of Lesser and Lapidus (*17*) illustrated the feasibility of solving the time optimal control problem via the LP format. It should be noted that it is now possible to handle as many as 200,000 unknowns and 1,024 equations within a 32K computer.

TABLE 1. DEFINITIONS OF OPTIMAL CONTROL AND NONLINEAR PROGRAMMING PROBLEMS

Optimal Control			Nonlinear Programming		
$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$	(system)	(1)	$I(\mathbf{z}) = I(z_1, \ldots, z_j, \ldots z_m)$	(index)	(6)
$\mathbf{x}(t_o) = \mathbf{x}_{(o)}$	(initial)	(2)	$C_i(\mathbf{z}) = \sum_{j=1}^{m} n_{ij} z_j - b_i = 0$	(constraint) $i = 1, \ldots, k$	(7*a*)
$I = \phi[\mathbf{x}(t_f)]$	(index)	(3)	$C_i(\mathbf{z}) = \sum_{j=1}^{m} n_{ij} z_j - b_i \leq 0$	(constraint) $i = k+1, k+2, \ldots$	(7*b*)
$\mathbf{C}(\mathbf{x}) \leq \boldsymbol{\beta}(t)$	(constraint)	(4)	$\sum_{j=1}^{m} n_{ij}^2 = 1$		
$\mathbf{C}(\mathbf{u}) \leq \boldsymbol{\alpha}(t)$	(constraint)	(5)			
Find $\mathbf{u}(t)$ which minimizes $I = \phi[\mathbf{x}(t_f)]$ subject to satisfying (*1*), (*4*), and (5), from a starting condition of (2). $\mathbf{x}(t)$ = state vector with n elements $\mathbf{u}(t)$ = control vector with r elements			Find $\mathbf{z}$ which minimizes $I(\mathbf{z})$ subject to satisfying linear equality (7*a*) and inequality (7*b*) constraints. $\mathbf{z}$ = vector with m elements		

Ho (*9*), Mel'ts (*21*), Ringlee (*22*), and Rosen (*23* to *25*) have examined the theory of the use of NLP to solve the optimal control problem. Cannon and Eaton (*2*), Fujisawa and Yasada (*6*), and Tracz and Bernholtz (*27*) have also looked into the special case where quadratic programming (the index is quadratic in the z_i) can be used. Mangasarian (*18*) has also developed extended

J. V. Flynn, Jr. is with E. I. DuPont Company, Wilmington, Delaware.

Kuhn-Tucker conditions in NLP (*19*) which then connect to the necessary conditions in the maximum principle (*13*) to indicate when a global optimum has been achieved. This work is a fine contribution which overlays both the optimal control and mathematical programming theories.

MAPPING OF THE CONTROL PROBLEM INTO A NLP PROBLEM

At this point we investigate the means of mapping the control problem into an equivalent NLP problem. The basis in the mapping resides in the linear autonomous version of Equation (1), namely,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{8}$$

where **A** and **B** are constant matrices. We may solve this linear nonhomogeneous equation and, assuming the control is piece-by-piece constant over a time period Δt, obtain the discrete-time solution (*13*)

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \quad k = 0, 1, 2 \ldots \tag{9}$$

where $\mathbf{\Phi} = \exp(\mathbf{A}\Delta t)$, $\mathbf{D} = \{\int_o^{\Delta t} \mathbf{\Phi}(\Delta t - \lambda)d\lambda\}\mathbf{B}$ and k indicates the time increment of Δt from $t = t_o$. If $\Delta t =$ constant and (9) is applied recursively there finally results

$$\mathbf{x}(k) = \mathbf{\Phi}^k\mathbf{x}(0) + \sum_{j=0}^{k-1} \mathbf{\Phi}^{k-1}\mathbf{D}\mathbf{u}(j) \tag{10}$$

Note that this equation relates the state vector at the end of the k time period to the initial state $\mathbf{x}_o$ and all the controls used to this point, that is, $\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(k-1)$. Thus Equation (10) may be used to eliminate the state at any time in terms of $\mathbf{x}_o$ and the controls. As such the control problem may be mapped into the NLP format in which the z_k are merely the discrete controls with the state terms eliminated by means of (10). If state constraints are involved in the original control problem these can also be converted into only control (or z_k) constraints. As a result of the use of (10), we may therefore convert the linear control problem into the form of (6) and (7) where the z_k are the controls and m represents the number of control periods.

Note that this allows us to eliminate the state variable $\mathbf{x}(k)$ in terms of $\mathbf{u}(k)$ and to then solve the NLP problem in control space. The logic behind this approach is that control space is usually of lower dimension than state space (*20*) and thus the number of unknown variables, z_k, is the smallest.

When the system equations are nonlinear, it is not possible to start with Equation (8). However, if one linearizes the nonlinear system equation by admitting small perturbations in the state and control then (1) can be written as

$$\delta\dot{\mathbf{x}}(t) = \mathbf{F}\,\delta\mathbf{x}(t) + \mathbf{G}\,\delta\mathbf{u}(t) \tag{11}$$

where $\mathbf{F} = \mathbf{f}_\mathbf{x}$ and $\mathbf{G} = \mathbf{f}_\mathbf{u}$ and the linearization has been carried out about a reference trajectory of $\mathbf{x}(t)$ and $\mathbf{u}(t)$. With the restrictions mentioned shortly, Equation (11) may be viewed as equivalent to (8) and the elimination of the state [now $\delta\mathbf{x}(t)$] in terms of only the control [now $\delta\mathbf{u}(t)$] can be accomplished exactly as before. The complications which now arise, however, are threefold. First, the variables must be expressed in the variational domain of $\delta\mathbf{x}$ and $\delta\mathbf{u}$ rather than in the original variable domain of $\mathbf{x}$ and $\mathbf{u}$. Therefore, the performance index and all constraints must be converted into this same domain. As a result the control problem, when the system equations are nonlinear, must be broken up into a series of linear control problems. This immediately causes the second computational problem in the sense that the NLP solution is now a subproblem in an overall iterative procedure. At each iterative step the linearization must be carried out, the conversion to the NLP format made and the NLP solved. This yields a correction $\delta\mathbf{u}$ to the previous $\mathbf{u}(t)$; this is continued until the correction approaches zero. The third problem deals with the constraints. On one hand there are the normal control problem constraints that restrict the state and control variables. These may be handled within the NLP procedure and do not cause any unusual difficulties. On the other hand, however, are the constraints on the control variations, $\delta\mathbf{u}(t)$, which must be introduced to assure that the linearization is maintained. Whereas the normal constraints are only important when one of the linear solutions reaches a constraint, the control variation constrants must be imposed continuously to guarantee a valid linearization.

The actual means of solving the NLP problem is of considerable importance in itself. There exist an endless number of different algorithms which one may use, but in the present work two were chosen. These are the projected-gradient method of Rosen (*23*) and the conjugate-gradient method of Goldfarb and Lapidus (*7*). We shall forgo any descriptions of the algorithms themselves since the references given present sufficient details. Of interest, however, is that Rosen's algorithm is a typical gradient method which exhibits slow convergence in the neighborhood of the optimum. Goldfarb and Lapidus' algorithm by contrast includes second-order terms and exhibits the feature of quadratic convergence. As a consequence one would probably expect this latter method to converge faster than Rosen's method on problems where the second-order information can be used.

COMPUTATIONAL RESULTS

To illustrate some of the features of the computational solution via this NLP approach we shall detail some calculations on a tubular reactor problem. These results are abstracted from a large number of other results on this problem; other problems have also been investigated including the Ho-Brentani (*10*) control-constrained case, the linear absorber problem (*13*) with 6 state variables and with state-constraints added and the singular reactor problem of Siebenthal and Aris (*26*). In all cases the results were equivalent to those to be described.

If we consider the reaction $A \rightarrow B \rightarrow C$ in a tubular plug-flow reactor then the relevant equations (mass balances) are

$$\begin{aligned} \dot{x}(t) &= k_1 x \\ \dot{y}(t) &= k_1 x - k_2 y \end{aligned} \tag{12}$$

where x and y represent the mole fraction of A and B, respectively at a point in the reactor corresponding to a residence time t. k_1 and k_2 are given by

$$k_i(T) = G_i \exp[-E_i/RT(t)] \quad i = 1, 2 \tag{13}$$

It is desired to maximize the yield of product B, $y(t_f)$, for a given feed $x(t_o) = x_o$ and $y(t_o) = y_o$ by selecting the control or temperature $T(t)$. The parameters used in this example are those given in the literature (*16*) and we note merely that $x_o = 0.95$, $y_o = 0.05$ and $t_f = 8$ min. Because of the physical problem it is apparent that the constraints $0 \leq x, y \leq 1.0$ must be imposed on the system.

As a first step in converting this problem to one suitable for the NLP format we linearize the system equations about a nominal trajectory, i.e., we go from the x, y, and T domain to the variational domain of δx, δy and δT in terms of perturbations around the time-dependent trajec-

tory indicated by $\bar{x}$, $\bar{y}$ and $\bar{T}$. For Equations (12) and (13) this approach yields

$$\delta\dot{x} = f_1\,\delta x + g_1\,\delta T$$

$$\delta\dot{y} = f_2\,\delta x + f_3\,\delta y + g_2\,\delta T \tag{14}$$

where

$$f_1 = -\bar{k}_1$$

$$f_2 = \bar{k}_1$$

$$f_3 = -\bar{k}_2$$

$$g_1 = -\bar{k}_1 \frac{E_1}{R\bar{T}^2}\bar{x}$$

$$g_2 = \bar{k}_1 \frac{E_1}{R\bar{T}^2}\bar{x} - \bar{k}_2 \frac{E_2}{R\bar{T}^2}\bar{y} \tag{15}$$

with the overbar indicating that items are to be evaluated along the nominal trajectory. Now Equation (14) is a time-varying but linear equation representing the variation around the assumed trajectory. The performance index which originally was merely

$$I = y(t_f)$$

can also be expressed as

$$\delta I = \delta y(t_f)$$

and we can now map the control problem into a NLP problem in control variation space.

For the explicit computational results to be described shortly, cases of a 1 min. and ½ min. time-steps were used. With $t_f = 8$ min. this means that there were 8 and 16 NLP variables (corresponding to δT), respectively

$$z_k = \delta u_k = \delta T_k$$

$$k = 0, \ldots, 7$$

or $k = 0, \ldots, 15$

In all cases the state variables were monitored to make sure that they remained in the range $0 \leq x, y \leq 1.0$. Boundaries, however, had to be put on z_k to keep the

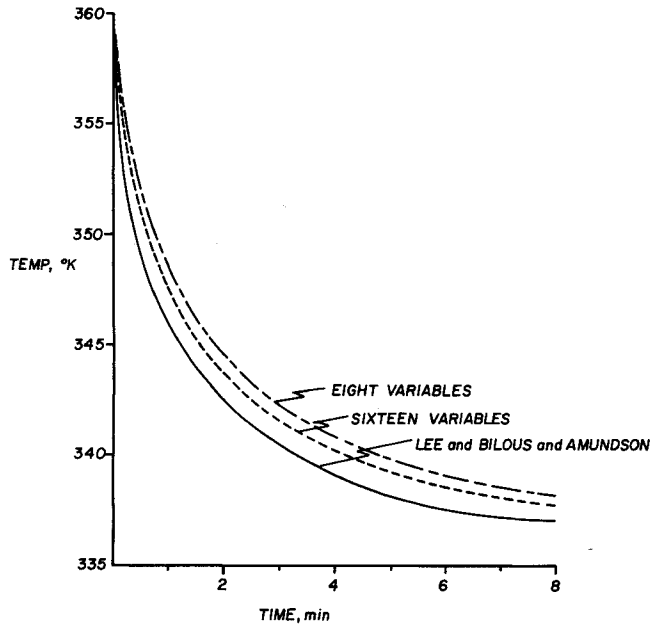

Fig. 1. Temperature °K. vs. time for the optimal temperature in a tubular reactor problem. Initial nominal profile is linear between 350 and 330°K.

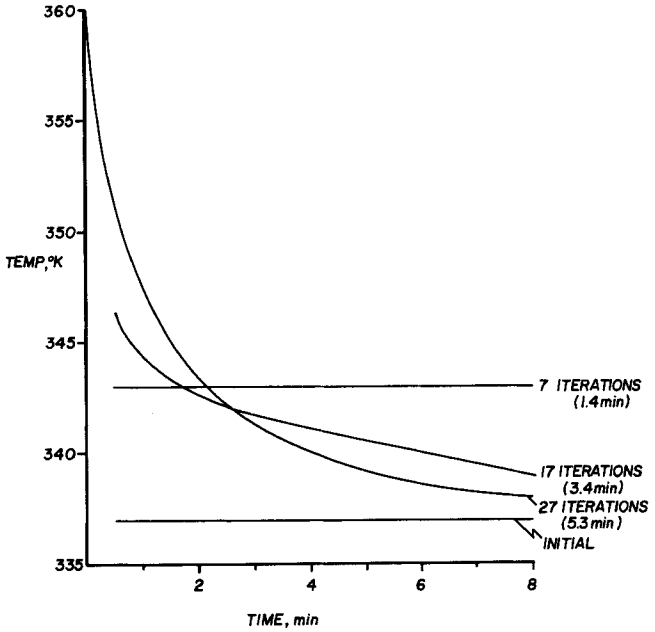

Fig. 2. Temperature vs. time profiles for the optimal temperature in a tubular reactor problem. Initial nominal profile is constant at 337°K.

linearization accurate. A linear initial nominal temperature profile was chosen from 350 to 330°K. and the bound on δT_k was restricted to 1 degree, $|\delta T_k| \leq 1$.

On this basis the problem becomes one of starting with the given initial nominal trajectory and using the projected gradient algorithm to yield the optimum change in temperature, δT_k. This optimum change is added to the original profile and a new state trajectory determined. Now the system equations will have new coefficients and the NLP procedure used again to yield a new optimum change in temperature. This is continued until no further changes can be made, the result being taken as the optimal results. The projected-gradient algorithm was used in the solution of the NLP problem.

Figure 1 shows the results of this study as well as the results of other researchers (*1, 16*). As can be seen the results approach those of other researchers as the number of time steps increases. This is due to a more accurate approximation of the true system by the piece-by-piece constant model studied here.

In order to check these results, two further computational runs were made. In the first case a different initial nominal control was used (a constant temperature profile of 337°K.); in the second case the allowable control variations were relaxed from $|\delta T_k| \leq 1$ to $|\delta T_k| \leq 3$. In both cases the end result of the iterations was virtually the same as in Figure 1. Thus one may conclude that the optimum reached is probably a global optimum and that the linearizations used were not violated during the course of the iterations. Figure 2 shows the change in control as the linearizations or iterations proceed from the starting nominal profile of a constant temperature of 337°K. While not shown the performance index exhibited a monotonic increase to the optimum value of $y(t_f) = 0.6758$.

To further study the usefulness of the NLP approach, constraints were placed on the state variables. The constraints were

$$0.5\,x + y \leq 0.8$$

or, in the variational domain,

$$0.5\,\delta x + \delta y \leq 0$$

These state variation constraints may be mapped into con-

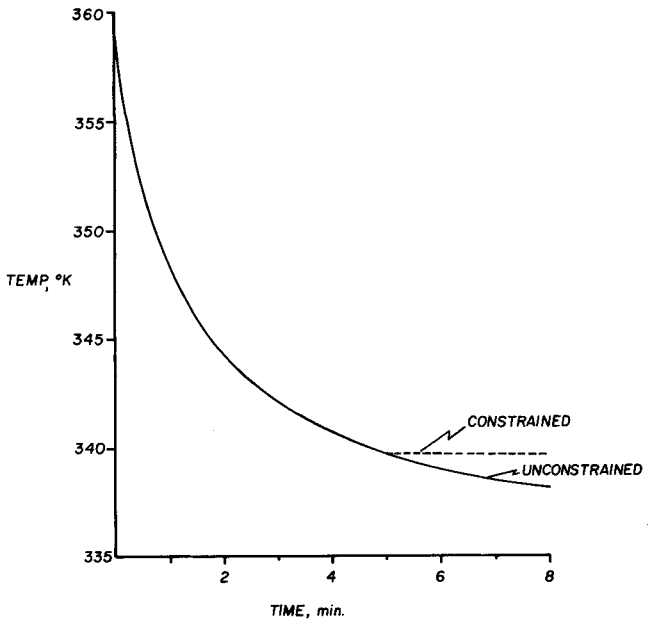

Fig. 3. Temperature vs. time for the state constrained optimal temperature profile in a tubular reactor problem.

trol space resulting in essentially the same program as used before. Now, however, it is required that when a constraint is met the variations are restricted from crossing the boundary and violating the constraint.

Figure 3 shows the effect of the state constraint on the temperature profile. This profile is significantly changed and, while not shown, the state trajectory meets the constraint but never violates it. Of specific interest is that the constrained problem only required about 10% more computer time than the unconstrained case.

As a final point of interest we turn briefly to a comparison of the projected and conjugate gradient algorithms for solving the NLP problem. Because the performance index is linear in the above example, maximize the yield, the two algorithms behave identically. If however we solve the control problem given by Ho and Brentani (*10*) where $I = \mathbf{x}(t_f)'\mathbf{x}(t_f)$ then the algorithms reveal some interesting differences in terms of efficiencies or rate of convergence. As a measure of efficiency one may consider the numerical value of the performance index vs. the

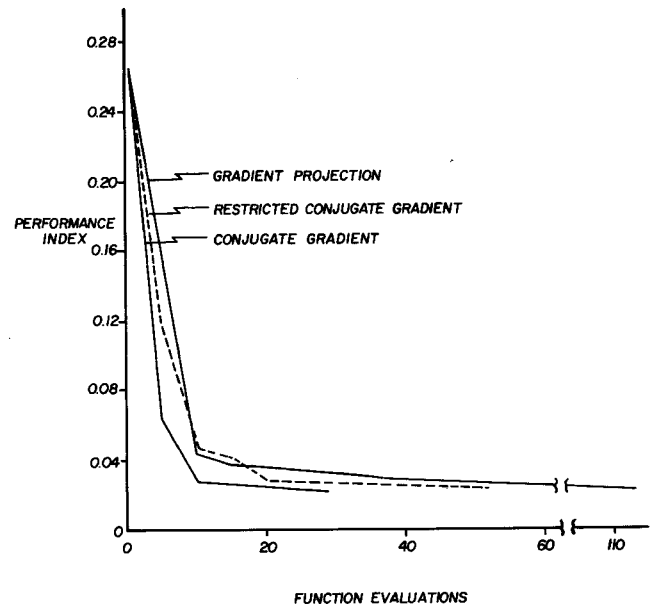

Fig. 4. Comparison of performance index vs. the number of function evaluations for the Ho-Brentani problem using two algorithms, 10 time-steps.

number of function evaluations in both algorithms. A large number of function evaluations means a greater amount of computing time for the solution. For the control constrained case, the projected gradient algorithm needed 113 function evaluations to reach the optimum value of the performance index, and this required 49 sec. of execution time on an IBM 7094 computer.

Shown in Figure 4 are the progress of two runs using the conjugate gradient algorithm. These two cases differ in the allowable step length in the algorithm; the restricted case results allowed a maximum step length of 1.0 while the second case allowed any step length calculated by the algorithm. In the first case 52 function evaluations and 13 sec. of execution time were required whereas in the unrestricted second case only 28 function evaluations and 9 sec. of execution time were required. Figure 4 shows at a glance the vast improvement in efficiency of the conjugate gradient over the projected gradient algorithm. It is of interest to point out that it is near the optimum condition where the major improvement occurs as seen by the short tails of the conjugate gradient results.

ACKNOWLEDGMENT

The authors are grateful to the National Science Foundation which under Grant NSF 6K-460 provided funds for this research. Numerical results were obtained using the facilities of the Princeton University Center supported in part by National Science Foundation Grant NSF-6P-579.

LITERATURE CITED

1. Bilous, O., and N. Amundson, *Chem. Eng. Sci.*, **5**, 81, 115 (1956).
2. Cannon, M., and J. Eaton, *J. Soc. Ind. Appl. Math. Control*, **4**, 34 (1966).
3. Denham, W. F., and A. E. Bryson, *AIAA Jrn.*, **2**, 25 (1964).
4. Dobell, H., and Y. C. Ho, *Trans. Auto. Control*, **12**, 4 (1967).
5. Fiacco, A. V., and G. P. McCormick, *Manag. Sci.*, **10**, 360 (1964).
6. Fujisawa, T. and Y. Yasada, *JMAA*, **19**, 586 (1967).
7. Goldfarb, D. and L. Lapidus, *Ind. Eng. Chem. Fundamentals*, **7**, 142 (1968).
8. Goldstein, A. A., *J. Soc. Ind. Appl. Math Control*, **3**, 142 (1965).
9. Ho, Y. C., *JMAA*, **5**, 216 (1962).
10. ——— and P. G. Brentani, *J. Soc. Ind. Appl. Math. Control*, **1**, 319 (1963).
11. Kim, M., *Inst. Soc. Am. Trans.*, **5**, 93 (1966).
12. Lapidus, L., and co-workers, *AIChE J.*, **13**, 101, 108, 114, 973, 982 (1967).
13. ———, and R. Luus, "Optimal Control of Engineering Processes," Blaisdell Publ. Co., (1967).
14. Lasdon, L. S., S. K. Miter, and A. D. Waren, *Auto. Control*, **12**, 132 (1967).
15. ——— A. D. Waren, and R. K. Rice, *ibid.*, **12**, 388 (1967).
16. Lee, E. S., *Ind. Eng. Chem. Fundamentals*, **3**, 373 (1964).
17. Lesser, H., and L. Lapidus, *AIChE J.* **12**, 143 (1966).
18. Mangasarian, O. L., *J. Soc. Ind. Appl. Math. Control*, **4**, 139 (1966).
19. ——— and Fromovitz, S., *JMAA*, **17**, 37 (1967).
20. McReynolds, S. R., *ibid.*, **19**, 565 (1967).
21. Mel'ts, I. O., *Auto. Remote Control*, **10**, No. 1, 68 (1968).
22. Ringlee, R. J., *Auto. Control*, **10**, 28 (1965).
23. Rosen, J. B., *J. Soc. Ind. Appl. Math.* **8**, 181 (1960); **9** 514 (1961).
24. ——— Proceedings IBM Scientific Computing Symposium (1966).
25. ——— *J. Soc. Ind. Appl. Math. Control*, **4**, 223 (1966).
26. Siebenthal, C. D., and R. Aris, *Chem. Eng. Sci.*, **19**, 729, 747 (1964).
27. Tracz, G. S., and B. Bernholtz, Personal Communication.