difference distribution presented in Figure 3 of reference 3. This figure shows that the analytical solution is applicable when the axial diffusion of vorticity overwhelms the convective transport and significant changes in the axial velocity occur before the fluid enters the conduit. The good agreement between these two solutions at this Reynolds number limit is considered to be evidence in support of the accuracy of the numerical solutions at other Reynolds numbers since the same numerical scheme was employed in all cases.

NOTATION

$g(z)$ = function as defined by Equation (43)
$I_n(\lambda)$ = modified Bessel function of first kind of order n
$J_n(\lambda)$ = Bessel function of first kind of order n
$K_n(\lambda)$ = modified Bessel function of second kind of order n
N_{Re} = Reynolds number
r = radial distance/radius of tube
U = axial component of velocity/average velocity in tube
V = radial component of velocity/average velocity in tube
z = axial distance/radius of tube

Greek Letters

ξ = variable as defined by Equation (5)
ψ = dimensionless stream function
ω = dimensionless component of vorticity vector

LITERATURE CITED

1. Courant, R., and D. Hilbert, "Methods of Mathematical Physics," Vol. I, p. 77 ff., Interscience, New York (1953).
2. Illingworth, C. R., "Laminar Boundary Layers," L. Rosenhead, ed., p. 163 ff., Oxford Press, London (1963).
3. Vrentas, J. S., J. L. Duda, and K. G. Bargeron, *A.I.Ch.E. J.*, **12**, No. 5, 837-844 (Sept., 1966).

Manuscript received February 17, 1966; revision received June 13, 1966; paper accepted June 14, 1966.

The Control of Nonlinear Systems. Part I:

Direct Search on the Performance Index

LEON LAPIDUS and REIN LUUS

Princeton University, Princeton, New Jersey

A straightforward and simple algorithm is presented for obtaining the optimal control of nonlinear systems. This algorithm has many of the advantages of dynamic programming but not the disadvantage of excessive computer storage. A number of numerical examples are presented to show the versatility of the method.

In the recent chemical engineering literature there has been an overabundance of papers on the development of computer algorithms suitable for control of nonlinear systems. Each of these papers has used some form of dynamic programming (*2, 10*) or the maximum principle (*4, 5, 7, 12, 15*). Of the two methods the former would seem to appeal more to engineers, since dynamic programming has the very important advantage that trajectory control and state constraints may be included in a simple fashion.

Unfortunately dynamic programming tends to flounder on the curse of dimensionality in which computer storage requirement rises exponentially with the number of state variables. Thus a three-state variable problem may require more high-speed storage than is currently available in the largest digital computers. At the same time it should be pointed out that considerable care must be used in both dynamic programming and the maximum principle or the results of an optimal control calculation may be in error (*8*).

In this paper we wish to develop a simple search technique for solving the optimal control problem, which retains many of the excellent features of dynamic programming but which eliminates the storage disadvantage. The primary idea of the method is to trade off computer storage for computing time, an almost necessary requirement for developing practical methods of solution of nonlinear problems.

The explicit method exploited here fits into the category of solution defined by Bellman as an approximation in policy space (*2*) and by Dreyfus as an approximation to the solution (*6*). Rather than requiring the use of derivatives to improve the control function (*5, 12*), the method is an application of finding a minimum along one or more coordinate directions at a time by using a direct search. Converse (*3, 4*) was the first to use this method in control problems but his procedure was a specific and special form of the algorithm used here. As a result his work has not received the attention that it warrants.

To illustrate the basic details of the algorithm the famous cross-current extraction system with recycle, a sequence of nonisothermal CSTR's and a plate type of gas absorber system will be analyzed and computational results will be presented.

PROBLEM DEFINITION

We consider in this paper systems which are described by the set of nonlinear differential equations and associated boundary conditions:

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t)] \quad (1)$$

$$\mathbf{x}(0) \quad \text{given} \quad (2)$$

Alternately we may conceive of a discretized form of Equation (1)

Rein Luus is at the University of Toronto, Toronto, Canada.

$$\mathbf{x}(k+1) = \mathbf{h}[\mathbf{x}(k), \mathbf{u}(k)] \quad k = 0, 1, 2, \ldots \quad (1a)$$

with the boundary conditions of Equation (2). In these equations $\mathbf{x}(t)$ is the state vector and $\mathbf{u}(t)$ is the control vector. It follows that, given some explicit form for $\mathbf{u}(t)$ or $\mathbf{u}(k)$, Equation (1) or (1*a*) may be used to generate a unique trajectory in $\mathbf{x} - t$ space with the trajectory originating at $\mathbf{x}(0)$.

For our discussion we shall assume a problem in which $\mathbf{u}(t)$ is sought between $t = 0$ and $t = t_f$ (where t_f is a fixed final time) or, alternately, $\mathbf{u}(k)$ is sought only between $k = 0$ and $k = P - 1$ (P stages of control). This assumption may be easily relaxed but it would only serve to complicate our discussion.

As a measure of the goodness of the selection of $\mathbf{u}(t)$ the scalar performance index

$$I[\mathbf{x}(0),t_f] = \int_0^{t_f} J[\mathbf{x}(t), \mathbf{u}(t)]dt \quad (3)$$

is introduced. In the form shown this index is determined as some integrated value of the state and control over the entire control time. The functionality on the left-hand side is taken to show that the index is a function of the initial state of the system and the overall time of control. Actually this performance index may take any one of a number of alternate forms; that is, it may involve only the final values of the state $\mathbf{x}(t_f)$, a weighted set of the final values $\mathbf{c}'\mathbf{x}(t_f)$, or a combination of $\mathbf{c}'\mathbf{x}(t_f)$ and the integral of Equation (3) (the prime indicates the vector transpose). The actual explicit form is not of great concern here but the reader may consult certain references on this point (*11, 13, 15*).

Now we can state the problem under investigation, namely, to select that $\mathbf{u}(t)$ which will extremize the performance index subject to the constraints of the system equation and the boundary conditions. In addition this $\mathbf{u}(t)$ may only be taken from a finite set if the control is bounded and there may be state constraints which bound the possible trajectories.

While we have Equations (1) to (3) in hand we may immediately note the rough features of the technique to be discussed. Thus if we guess the function $\mathbf{u}(t)$, call it $\mathbf{u}^{(0)}(t)$, it follows that by starting with the known $\mathbf{x}(0)$ the system equations may be integrated to $t = t_f$. At this point the value of $I[\mathbf{x}(0), t_f]$, call it $I^{(0)}[\mathbf{x}(0), t_f]$, can also be evaluated. Now, except by chance, the $\mathbf{u}^{(0)}(t)$ chosen is not the optimal $\mathbf{u}(t)$, call it $\mathbf{u}^o(t)$, since $I^{(0)}[\mathbf{x}(0), t_f]$ is not the extreme value. Thus we will suggest a means of finding a new $\mathbf{u}(t)$, $\mathbf{u}^{(1)}(t)$, which will be better than the first choice. In the limit, $\mathbf{u}^{(j)}(t)$ is assumed to converge to that $\mathbf{u}^o(t)$ which extremizes $I[\mathbf{x}(0), t_f]$; this, then, is the optimal control function.

We note that the initial conditions and the system equations are unaffected in this algorithm but that the performance index is the measure of the convergence of the process. For this reason we call the method, direct search on the performance index.

THE DIRECT SEARCH ALGORITHM

Let us start our discussion of the algorithm by rewriting the performance index in the form

$$I[\mathbf{x}(0),t_f] = \int_0^{t_f} J[\mathbf{x}(t),u_1(t),u_2(t), \ldots, u_r(t)]dt \quad (4)$$

where we have specified that $\mathbf{u}(t)$ contains r elements. We desire to find that $\mathbf{u}(t)$ which extremizes $I[\mathbf{x}(0),t_f]$.

Let us assume certain functions for $u_1(t)$, $u_2(t)$, . . ., $u_r(t)$ and call these $u_1^{(0)}(t)$, $u_2^{(0)}(t)$, . . ., $u_r^{(0)}(t)$ to indicate that they are the starting functions. Obviously once these values are specified it is possible to substitute into $J[\mathbf{x}(t),u_1^{(0)}(t), \ldots, u_r^{(0)}(t)]$, to use the system equations to generate the $\mathbf{x}(t)$, and to calculate $I[\mathbf{x}(0),t_f]$ from Equation (4). Since this value of the performance index is obtained from the initially assumed set of $\mathbf{u}^{(0)}(t)$, we shall call it $I^{(0)}[\mathbf{x}(0),t_f]$. Next we allow $u_1(t)$ to vary and we find that $u_1(t)$, which minimizes (obviously maximizes could be used just as easily)

$$I[\mathbf{x}(0),t_f] = \int_0^{t_f} J[\mathbf{x}(t), u_1(t), u_2^{(0)}(t), u_3^{(0)}(t), \ldots, u_r^{(0)}(t)]dt \quad (5)$$

with t_f fixed. In other words, we fix $u_2(t), \ldots, u_r(t)$ at the assumed values $u_2^{(0)}(t), \ldots, u_r^{(0)}(t)$ and only vary $u_1(t)$ until that value is found which minimizes $I[\mathbf{x}(0), t_f]$. It is, of course, assumed that as $u_1(t)$ varies, $\mathbf{x}(t)$ also varies. We call the resulting index $I^{(1)}[\mathbf{x}(0),t_f]$ with control vector $\mathbf{u}'(t) = [u_1^{(1)}(t), u_2^{(0)}(t), \ldots, u_r^{(0)}(t)]$. At this point note that

$$I^{(1)}[\mathbf{x}(0),t_f] \leq I^{(0)}[\mathbf{x}(0),t_f] \quad (6)$$

since at worst the same $u_1(t)$, namely, $u_1^{(0)}(t) = u_1^{(1)}(t)$, would be retained. This procedure is now continued by finding that $u_2(t)$, which minimizes

$$I[\mathbf{x}(0),t_f] = \int_0^{t_f} J[\mathbf{x}(t),u_1^{(1)}(t),u_2(t), u_3^{(0)}(t), \ldots, u_r^{(0)}(t)]dt \quad (7)$$

to yield

$I^{(2)}[\mathbf{x}(0),t_f]$ and $\mathbf{u}'(t)$

$$[u_1^{(1)}(t)u_2^{(2)}(t)u_3^{(0)}(t) \ldots u_r^{(0)}(t)].$$

As before

$$I^{(2)}[\mathbf{x}(0),t_f] \leq I^{(1)}[\mathbf{x}(0),t_f] \quad (8)$$

Obviously, this algorithm can be continued in the same fashion, searching on each control and yielding monotonic convergence to $I^o[\mathbf{x}(0),t_f]$, the minimum value of $I[\mathbf{x}(0),t_f]$. However, the question of whether this yields a global minimum or a local minimum is not known. A test of this point is to consider a number of widely different initial decision vectors $\mathbf{u}^{(0)}(t)$ and to see if they all converge to the same value of the performance index. If they do converge to the same value, global optimization is probably assured.

When the minimum value of the performance index is found, one complete optimal trajectory is immediately available. Furthermore, the computer storage required for a multidimensional problem is quite small, since only the last $I^{(j)}[\mathbf{x}(0),t_f]$ and the last control function $\mathbf{u}^{(j)}(t)$ have to be retained. The extension to a variable t_f can be carried out with no difficulty and furthermore the control and state constraints can be incorporated. This last point is easily seen in the material to follow.

There are a number of interesting variations on this method which, while not quite as primitive, still retain the advantages of essentially negligible storage and of computer programming simplicity. The major disadvantage is the possibly large computing times required. One such method is *pair-interchange* as labeled by Jensen (*9*); we shall illustrate it with the discrete system

$$\mathbf{x}(k+1) = \mathbf{h}[\mathbf{x}(k), u(k)] \quad (9)$$

$$\mathbf{x}(0) \quad \text{prescribed}$$

with $u(k)$ as a scalar for simplicity. In the pair-interchange method, the first step is to guess a nominal (assumed) control sequence, $u^{(0)}(k)$, $k = 0, 1, 2, \ldots, P-1$, each of which must lie at any one of Q values (the control variable in the k stage is discretized into Q values). The system equations are now evaluated with this assumed control sequence and the performance index $I^{(0)}[\mathbf{x}(0),P]$ results.

Now we fix the controls $u(2), u(3), \ldots, u(P-1)$ at the initially assumed $u^{(0)}(2), u^{(0)}(3), \ldots, u^{(0)}(P-1)$,

respectively, and consider variations in $u(0)$ and $u(1)$. Each of these latter control values can assume any of Q values. Taken in combination, they can assume Q^2 values. Let us assume each of the Q^2 pairs and for each pair solve the system equations and evaluate the performance index. If $Q = 15$, $Q^2 = 225$ and 225 trajectories must be evaluated. Of these 225 trajectories, the one which yields the minimum value of the performance index, $I^{(1)}[\mathbf{x}(0),P]$, is chosen. Of course

$$I^{(1)}[\mathbf{x}(0),P] \leq I^{(0)}[\mathbf{x}(0),P] \qquad (10)$$

since, as before, the worst case is to retain the pair $u^{(0)}(0)$ and $u^{(0)}(1)$. The control sequence is now $u^{(1)}(0)$, $u^{(1)}(1)$, $u^{(0)}(2), \ldots, u^{(0)}(P-1)$.

Next consider that the control sequence is fixed, except for the second and third stages, at $u^{(1)}(0)$, $u^{(0)}(3), \ldots, u^{(0)}(P-1)$. Interchange $u(1)$ and $u(2)$ to all Q^2 pairs and solve the system equations and performance index for all pairs. The best performance index is $I^{(2)}[\mathbf{x}(0),P]$ where

$$I^{(2)}[\mathbf{x}(0),P] \leq I^{(1)}[\mathbf{x}(0),P] \qquad (11)$$

and the control sequence is $u^{(1)}(0)$, $u^{(2)}(1)$, $u^{(2)}(2)$, $u^{(0)}(3), \ldots, u^{(0)}(P-1)$. Obviously this pair-interchange can be continued until $u(P-1)$ is reached and then the entire sequence can be repeated and continued until the performance index becomes constant within some error bound. When this occurs no further improvement in the performance index is possible. Clearly an extremum has been reached, since any variation in the established control policy does not improve the performance index. Note that this method provides monotonic convergence, that the storage required is negligible, and that the computer program used to implement the method is quite simple.

If constraints on the control variables exist, the Q allowable mesh points are suitably restricted; thus constrained control presents no difficulty. If $r > 1$, more than one control variable is involved; after searching on the first control variable over the range $0 \leq k \leq P-1$, the second control variable is searched in the same fashion over the same range, etc. Thus in the multiple control vector case there is actually a series of subsearch cycles. On this basis it is also apparent that the method will have its greatest utility when the system involves a large number of state variables but a small number of control variables.

If constraints on the state variables exist, this is equivalent to the problem encountered in dynamic programming. By specifying that no control be used which yields a state which is outside of the constraints, nonadmissible states are eliminated and the constraints are handled.

More general interchange methods can immediately be suggested. For example, a triplet interchange could be used in which $u(k-1)$, $u(k)$, $u(k+1)$ are scanned simultaneously. Possibly some random procedure could be used whereby members of the control sequence are randomly changed. Details of these more sophisticated procedures will not be given here, since they are self-apparent.

In summary, we note that by trading off computer storage for computer time an algorithm of extremely simple form results. This algorithm is quite easy to program for a digital computer, it requires a small amount of storage, and it may handle control and state constraints. The nonlinearity of the system equations and the dimensions of the state vector are of minor importance. Other advantages to this method are that one may learn how sensitive the performance index is to the control policy, and the technique has direct application to on-line experimental plant optimization since the performance index converges monotonically. The defects of the method are the large (possibly) computer time involved and the question of the limit of the convergence process. This latter point is of interest, since the method may also find interesting application in terms of yielding an initial starting point for more complicated algorithms (*5, 11, 12*) which involve the use of first or second derivatives.

This matter of large computing time deserves further discussion. Since the only way to solve problems of the type we are discussing is by iterative methods, the current procedure probably does not lead to excessive computer time. When compared with the methods of dynamic programming and the maximum principle, in their fundamental form, the time is excessive. However, since it is almost impossible to use these two methods for any problem other than simple ones, such a comparison has no real meaning.

Finally, it should be pointed out that while the algorithm has been detailed as proceeding in a forward time direction, it could equally be used in a backward time direction. In this latter context the comparison to the standard form of dynamic programming becomes evident, although the dual principle of dynamic programming (*11*) allows either a forward or backward pass.

NUMERICAL EXAMPLES

The direct search method suggested will be illustrated with three different system models. These systems exhibit a progressive increase in order of mathematical complexity. Analogous results have been obtained for a variety of other and more complex systems but, to minimize the amount of discussion, only three of the total will be used.

Example 1

The first system of interest is the now famous three-stage, cross-current extraction system with recycle (Figure 1). This system has been the source of a number of papers but we only mention the most recent ones by Rosen (*16*) and by Lovland (*14*). In both of these communications the proper dynamic programming algorithm is developed for this system. In the present work we quote results obtained by Baker (*1*), using direct search on the performance index.

In terms of Figure 1, the steady state material balance for the n^{th} stage is

$$\frac{Q_1 + Q_2}{Q_1}(x_{n+1} - x_n) = \frac{w_n}{Q_1} y_n = v_n\, y_n \qquad (12)$$

The equilibrium between the output raffinate and extract phases is given by

$$y_n = f(x_n) \qquad (13)$$

and a profit or performance index is defined as

$$P_n = \frac{Q_1 + Q_2}{Q_1}(x_{n+1} - x_n)\left(1 - \frac{C}{f(x_n)}\right) \qquad (14)$$

where C is a relative cost factor. The optimal problem of interest is: Given x_f, Q_1, and Q_2, select the allocations v_1

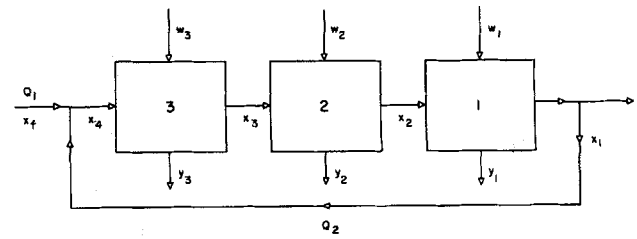

Fig. 1. Cross-current extraction system with recycle.

TABLE 1. CROSS-CURRENT EXTRACTION SYSTEM WITH RECYCLE

	Run I Accuracy = 5%	Run II Accuracy = 5%	Run III Accuracy = 1%
	$10^{-10} \leq v \leq 2.0$	$10^{-10} \leq v \leq 2.0$	$10^{-2} \leq v \leq 2.0$
	$x_3^{(0)} = 0.2000$	$x_3^{(0)} = 0.2000$	$x_3^{(0)} = 0.2000$
	$x_2^{(0)} = 0.1333$	$x_2^{(0)} = 0.0800$	$x_2^{(0)} = 0.1333$
	$x_1^{(0)} = 0.0667$	$x_1^{(0)} = 0.0400$	$x_1^{(0)} = 0.0667$
No. of cycles	9	9	8
Computation time, sec.	29	24	19
Optimal x_i's	$x_3 = 0.1185$	0.1186	0.1189
	$x_2 = 0.0724$	0.0736	0.0798
	$x_1 = 0.0500$	0.0510	0.0534
Optimal v_i's	$v_3 = 0.6040$	0.5840	0.4916
	$v_2 = 0.3620$	0.3620	0.4100
	$v_1 = 0.2660$	0.2800	0.3085
Optimal profits	$p_3 = 0.0130$	0.0138	0.0156
	$p_2 = 0.0395$	0.0409	0.0480
	$p_1 = 0.1016$	0.1017	0.1017

$= w_1/Q_1$, v_2, and v_3 such that the cumulative profit

$$\sum_{1}^{3} p_n = P \qquad (15)$$

is maximized subject to the constraints of Equations (12) and (13). In each stage of the system there is one state variable x_n and one control variable v_n. Note that the stages are numbered backward to conform to the literature.

In all of the computations to be described Equation (13) is represented as a third-degree polynomial fit to the tabulated equilibrium data of Rosen (*16*). In addition, a one-dimensional search routine was used to scan a permissible range of the v_n. In this routine an upper and lower bound is specified for v_n; the routine finds the midpoint of this range and then searches for a point where the profit is higher. The amount of variation allowed in the search is one-tenth the range of the bounds. When the best point is found with this step size, the step size is divided by ten and the neighborhood of the first optimum is examined. This is continued until ten size reductions have been made.

The specific version of the algorithm is:

1. Guess a set of allocations, $v_3^{(0)}$, $v_2^{(0)}$, and $v_1^{(0)}$.
2. Holding $v_2^{(0)}$ and $v_1^{(0)}$ fixed we proceed to scan a set of possible v_3. For *each possible* v_3 the new outputs x_2 and x_1 are calculated with x_f as a first guess for x_4. But once x_1 is calculated, a new x_4 may be calculated by using the material balance of the recycle and feed streams:

$$x_4 = (Q_1\ x_f + Q_2\ x_1)/(Q_1 + Q_2) \qquad (16)$$

Since this value of x_4 will yield a new value of x_3 the calculation of x_2 and x_1 is repeated. Finally the entire cycle is continued for the *one value of* v_3 until x_3 becomes constant. The total profit is then calculated for this v_3.
3. When this is repeated for all the possible v_3, there will result a $v_3^{(1)}$ which yields the maximum total profit associated with $v_2^{(0)}$ and $v_1^{(0)}$.
4. Holding $v_3^{(1)}$ and $v_1^{(0)}$ fixed we scan v_2 to yield $v_2^{(1)}$; in turn v_1 is scanned to yield $v_1^{(1)}$.
5. At this point there exist $v_3^{(1)}$, $v_2^{(1)}$, and $v_1^{(1)}$ and a return to step 2 is made. This overall cycle is repeated until no further change in the value of the profit is observed. As can be seen, the algorithm is essentially identical to that discussed previously and negligible computer storage is required.

To illustrate the results of this algorithm, we choose the following parameter values: $x_f = 0.20$, $C = 0.05$, $Q_1 = 1.0$, and $Q_2 = 1.0$. For comparison two preliminary runs were made. The first corresponded to $Q_2 = 0$ (no recycle); the use of dynamic programming in a straightforward manner to solve for the optimal allocation required 5.5 sec. on an IBM 7094 computer. This computational run then serves as a measure of comparison for the recycle case in terms of computation time. The second run was to use dynamic programming as detailed by Rosen with $Q_2 = 1.0$ (with recycle); this showed that the present search method and interpolation formula for the equilibrium data gives results analogous but not exactly equal to those of Rosen. This slight difference is undoubtedly due to the present use of a third-degree polynomial to fit the equilibrium data.

With these preliminaries the three runs shown in Table 1 were made. The accuracy refers to the tolerance used in step 2 on recalculating x_4 and the bounds on v_i's are also given. In the actual calculations, it proved simpler to guess initially for the x_i's rather than for the v_i's. Which is guessed is immaterial, since the material balance allows the recovery of the other item. Thus in Table 1 the initial estimates are shown on x_3, x_2, and x_1.

These runs illustrate the effects of different initial guesses or starting points in the iteration and different tolerances on accuracy and bounds of the v_i's. In each case the results are essentially equivalent to those obtained with the dynamic programming algorithm but with a four- to five-fold increase in computation time. This illustrates the trading of storage for computer time associated with the current algorithm.

Example 2

As an example of a more complex system we choose the nonisothermal reactor series (see Figure 2) with chemical reaction steps

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

k_1 and k_2 are temperature-dependent rate constants of the form

$$k_i = G_i e^{-E_i/RT}$$

Since the reactions are first order

$$-(dA/dt) = k_1\ A = r_A$$

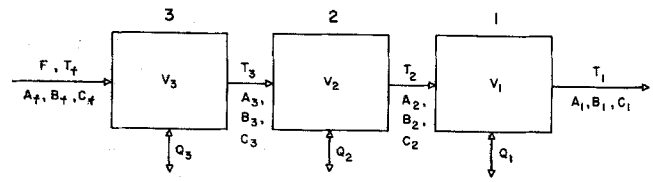

Fig. 2. Series of nonisothermal reactors.

TABLE 2. NONISOTHERMAL REACTORS

Run 1: $300° \leq T_n \leq 500°$ $\quad 10^{-10} \leq A_n \leq 1.0$

Number of cycles to convergence = 6
Computation time = 9 min., 44 sec.

	Stage n	Input A_{n+1}	Output A_n	Input B_{n+1}	Output B_n	Input T_{n+1}	Output T_n
Initial guesses	1	0.1667	—	0.3333	—	350.0	—
	2	0.3333	0.1667	0.1667	0.3333	350.0	350.0
	3	0.5000	0.3333	0.0	0.1667	350.0	350.0

	Stage n	Input A_{n+1}	Output A_n	Input B_{n+1}	Output B_n	Input T_{n+1}	Output T_n
Final iteration	1	0.0710	0.0243	0.3766	0.4093	300.0	300.0
	2	0.2635	0.0710	0.2022	0.3766	340.2	300.0
	3	0.5000	0.2635	0.0	0.2022	350.0	340.2

	D_n	Q_n	Cumulative profit
1	384.9	—0.0158	3.086
2	540.9	—0.1133	20.23
3	5.172	—7.128	40.63

$$-(dB/dt) = -k_1 A + k_2 B = r_B$$

Thus at steady state, the material and heat balances for stage n are

$$A_n = A_{n+1}/(1 + D_n k_1) \qquad (17)$$

$$B_n = (B_{n+1} + D_n k_1 (A_{n+1} + B_{n+1}))/(1 + k_2 D_n)(1 + k_1 D_n) \qquad (18)$$

$$C_p\rho(T_{n+1} - T_n) - D_n \sum_1^2 r_i \Delta H_i = D_n Q_n \qquad (19)$$

where $D_n = v_n/F$ is holding time of the n^{th} stage and Q_n is heat added or removed by a cooling coil. All other symbols are self-evident. In particular, the following parameters have been used here:

$G_1 = 0.535 \times 10^{11}$ min.$^{-1}$ $\quad \Delta H_1 = 100.0$ cal./mole
$G_2 = 0.461 \times 10^{18}$ min.$^{-1}$ $\quad \Delta H_2 = 100.0$ cal./mole
$E_1 = 1.8 \times 10^4$ cal./mole $\quad C_p\rho = 1.0$ cal./deg.
$E_2 = 3.0 \times 10^4$ cal./mole

The profit or performance index for stage n is chosen to be

$$p_n = M_1(B_n - B_{n+1}) - M_2|Q_n| - M_3 D_n \qquad (20)$$

namely, as the difference between the value of the intermediate component B and the cost of the heat addition or removal and the holding time. As a simplification, the absolute value of Q is used in the profit function, implying that the cost of adding or removing heat is the same. M_1, M_2, and M_3 are cost factors chosen as $M_1 = 100.0$, $M_2 = 0.0005$, and $M_3 = 0.0005$.

In the explicit calculations all the input and output concentrations and temperatures are selected initially; this is analogous to selecting the control variables, since once the feed concentrations A_f and B_f and the feed temperature T_f are given, either set of information can be calculated from the other. At the same time it can be seen that if the A_{n+1}, B_{n+1}, and T_{n+1} are known as inputs to stage n (see Figure 2) and the outputs A_n and T_n are selected, Equations (17) to (19) can be used to calculate D_n and Q_n, the control variables. On this basis it seems natural to set up a range of A_n and T_n and, when working on a single stage, to scan the range of these two variables to determine the best set of D_n and Q_n. When a single stage is being considered its effect on the lower or downstream stages can be easily ascertained, since for each D_n and Q_n there will be a corresponding A_n, B_n, and T_n to the lower $(n-1)$ stage; but if this corresponds to the feed to stage $n-1$ and D_{n-1} and Q_{n-1} are fixed values, it is simple to calculate A_{n-1}, B_{n-1}, and T_{n-1}, etc.

Table 2 shows the result of a numerical calculation on the current system. In this calculation the A_n and T_n are limited by $10^{-10} \leq A_n \leq 1.0$ and 300°C. $\leq T_n \leq$ 500°C. It can be seen that convergence occurred in six cycles and that the optimal temperatures in stages 2 and 1 are on the lower boundary of 300°C. In another calculation (not shown here) the lower limit on T_n was dropped to 200°C. In this case lower optimal temperatures in stages 2 and 1 were achieved in five cycles. These lower temperatures allowed a subsequent higher profit, namely, of 44.25 rather than 40.63 as in Table 2.

The results shown indicate that convergence occurs in a relatively small number of cycles but that the computation time is relatively high. Once again we see the exchange of computer storage requirements for computer time. In the present case there is no way to specify whether the converged optima are really the true optima and not local conditions. However, results not tabulated here with different starting conditions all yield the equivalent optimum and thus a true maximum condition has probably been achieved.

Example 3

The third system is a six-plate absorption column as detailed by Lapidus (*10*) (see Figure 3) and implemented by Jensen (9). In this system we are concerned with dynamic control, as distinct from examples 1 and 2, which were steady state optimization problems. Stages in time will now replace the physical stages of the two previous examples.

The material balance equation for the gas absorber can be represented by

$$\dot{\mathbf{x}}(t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \qquad (21)$$

where $\mathbf{x}(t)$ is a vector with six elements corresponding to the liquid compositions on each plate and $\mathbf{u}(t)$ is a vector with two elements corresponding to the liquid and gas feed compositions

$$\mathbf{A} = \begin{bmatrix} -1.173 & 0.6341 & & & & \\ 0.5390 & \cdot & \cdot & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & 0.6341 \\ & & & & 0.5390 & -1.173 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} 0.5390 & 0 \\ 0 & 0 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & 0 \\ 0 & 0.6341 \end{bmatrix}$$

In the present example [detailed further by Baker (*1*)] we select two control variables, D_n and Q_n; three state variables, A_n, B_n, and T_n; and seek to maximize the cumulative profit $\sum_1^3 p_n$.

The control problem of interest here is given the initial condition

$$\mathbf{x}(0)' = [-0.0306 \quad -0.0568 \quad -0.0788 \quad -0.0977 \quad -0.1138 \quad -0.1273]$$

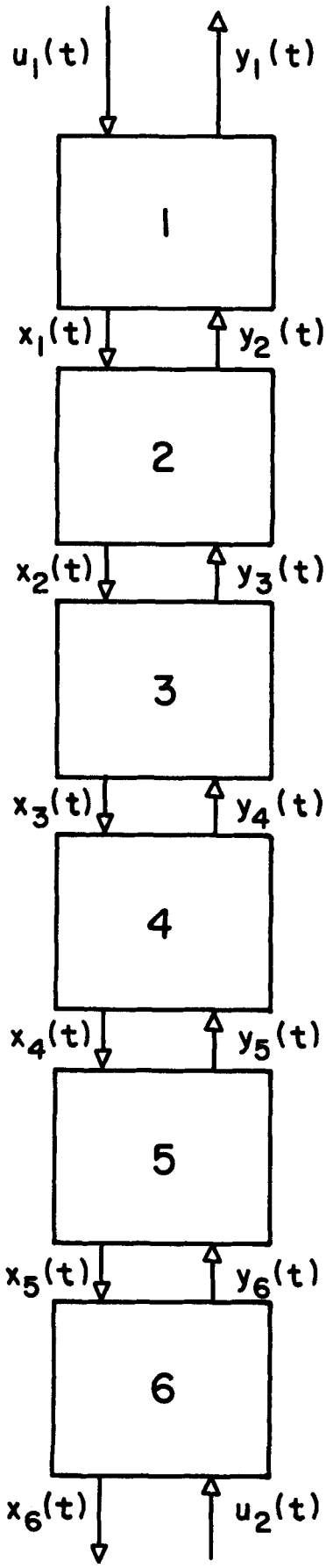

Fig. 3. Gas absorber.

corresponding to

$$\mathbf{u}(0)' = [0 \quad -0.1389]$$

to find that control function $\mathbf{u}(t)$ such that the system goes from $\mathbf{x}(0)$ to $\mathbf{x}(t_f)$ while minimizing the performance index

$$I[\mathbf{x}(0), t_f] = \int_0^{t_f} [\mathbf{x}(t)' \ \mathbf{Q}\, \mathbf{x}(t) + \mathbf{u}(t)' \ \mathbf{R}\, \mathbf{u}(t)]dt \tag{22}$$

$\mathbf{Q}$ and $\mathbf{R}$ are suitable constant weighting matrices for $\mathbf{x}(t)$ and $\mathbf{u}(t)$; thus the system equations are linear while the performance index is nonlinear; in fact it is quadratic in $\mathbf{x}(t)$ and $\mathbf{u}(t)$. The negative numbers for $\mathbf{x}(0)$ and $\mathbf{u}(0)$ are due to a normalization which has been made such that $\mathbf{x}(t_f) = \mathbf{0}$ and $\mathbf{u}(t_f) = \mathbf{0}$.

For comparison of the results of the present work we also can quote the work of Lapidus (*10*) in which an optimal value of the performance index of $I^o[\mathbf{x}(0), 15] = 0.04167$ was obtained for $\mathbf{Q} = \mathbf{I}$ and $\mathbf{R} = \mathbf{0}$. These results were obtained by making use of the linearity of the system equations, a simplification which we will not take advantage of here.

In accordance with the strategy set down for the direct search on the performance index, we first select or choose a nominal set of the control variables $\mathbf{u}^{(0)}(t) = \begin{bmatrix} u_1^{(0)}(t) \\ u_2^{(0)}(t) \end{bmatrix}$; once these values are chosen the system Equations (21) are integrated numerically with a fourth-order Runge-Kutta-Gill subroutine for first-order ordinary differential equations. Control $u_1(t)$ is allowed to assume any of Q_1 discrete values and control $u_2(t)$ is allowed to assume any of Q_2 discrete values in this calculation; these are held constant for the sampling period, that is, for a period of time equivalent to the discrete staging in time to be used. Thus while we write the system equations in the continuous form above, in actual use we really use a discrete form, that is, because of the method of integration, which does not use the benefit of linearity, Equation (21) is really used in the form

$$\mathbf{x}(k+1) = \mathbf{h}[\mathbf{x}(k), \ \mathbf{u}(k)] \tag{23}$$

In the same sense, because the applied control is held constant for the sampling period, the integral in Equation (21) is replaced by an equivalent sum over P terms. Alternately we may replace Equations (22) and (23) by

$$\mathbf{x}(k+1) = \mathbf{x}(k) + [\mathbf{A}\ \mathbf{x}(k) + \mathbf{B}\ \mathbf{u}(k)]T \tag{24}$$

and

$$x_7(k+1) = x_7(k) + [\mathbf{x}(k+1)' \ \mathbf{Q}\ \mathbf{x}(k+1) + \mathbf{u}'(k)\ \mathbf{R}\ \mathbf{u}(k)]T \tag{25}$$

where the new variable $x_7(k)$ represents the cumulative sum of the performance index and T is the sampling period. Now we seek to minimize $x_7(P)$, which serves as the performance index for the calculation.

The actual calculation proceeds exactly as outlined previously for the pair-interchange algorithm. First, with $u_2(k)$ held at its nominal value $u_2^{(0)}(k)$, $u_1^{(0)}(0)$ and $u_1^{(0)}(1)$ are scanned to find the best set from among the allowable Q_1^2 set of controls. Best here means that set which yields the smallest value of $x_7(P)$. Second, $u_1^{(1)}(0)$ is fixed at its newly improved value and $u_1^{(0)}(1)$ and $u_1^{(0)}(2)$ are adjusted in the same manner. Third, this is continued for all the double combinations of $u_1^{(0)}(k)$; then with $u_1(k)$ fixed at the improved values the procedure is repeated with $u_2^{(0)}(k)$ by scanning Q_2^2 values at each step. This completes one computation cycle. The entire cycle is then repeated until finally some preselected criterion such as no further changes in $x_7(P)$ is met.

To illustrate the results of such a calculation we select first the case of $\mathbf{Q} = \mathbf{I}$, $\mathbf{R} = 0$, and choose a sampling period of $T = 1$ min., an integration time of 0.1 min. in the Runge-Kutta-Gill routine and use $P =$ fifteen total stages. Now we select the nominal policy

$$u_1^{(0)}(k) = 0$$
$$u_2^{(0)}(k) = -0.100 \qquad k = 0, 1, 2, \ldots, 14$$

and allow the discrete controls

$u_1(k)$	$u_2(k)$
0.0	−0.100
0.005	−0.105
0.010	−0.110
0.020	−0.120
0.030	−0.130
0.040	−0.140
0.050	−0.160
0.070	−0.180
0.090	−0.240
0.110	−0.280
0.150	−0.330
0.200	−0.350
0.220	−0.370
0.250	−0.400
	−0.480

Note that $Q_1 = 14$ and $Q_2 = 15$ and we refer to this as a (14, 15) allowable control mesh. For this sequence the results of the pair-interchange method on the performance index are shown in the first part of Table 3. In five cycles this method has converged to a value quite close to that obtained by Lapidus, namely, 0.04684 and 0.04617, respectively.

The defect in the present method is, of course, the large amount of computing time required. For the current problem the pair-interchange method requires about fifteen times as much computer time to achieve the answer as does the method of Lapidus (*10*). However, this must be tempered by pointing out that no use was made here of the linearity of the system equations, and in fact a nonlinear system of the same dimension would probably require about the same calculation time.

To proceed further with this calculation we now raise the question of the effect of the number of points in the allowable control mesh. In other words, what effect does the quality of discretizing of the control have on the final result. In the second part of Table 3 we show some further results for (4, 9), (6, 11), and (13, 14) allowable control meshes with the deleted values taken out roughly from the entire range of the (14, 15) case. We see that as a more refined control is allowed, the performance index does approach the value of 0.04617.

TABLE 3. DIRECT SEARCH ON ABSORPTION SYSTEM

$\mathbf{Q} = \mathbf{I}, \mathbf{R} = 0$

Computation cycle	Performance index
0	0.15671
1	0.06958
2	0.04928
3	0.04713
4	0.04686
5	0.04684

Allowable control mesh	Final performance index
(4, 9)	0.0623
(6, 11)	0.0569
(13, 14)	0.0470
(14, 15)	0.0468

Equivalent results were also obtained for the case $\mathbf{Q} = \mathbf{I}$, $\mathbf{R} = \mathbf{I}$, with an optimal performance index obtained after five cycles with a (11, 15) allowable control mesh. While we have concentrated here on the performance index, it should be pointed out that the resulting control and state trajectories are essentially identical to those given by Lapidus.

In summary, it can be seen that for each of the three problems discussed here the direct search on the performance index yields convergence to an optimum. This optimum is, in certain of the cases, the global optimum. The method involved allows for easy computer programming and can handle control and state constraints.

ACKNOWLEDGMENT

The authors wish to acknowledge the work carried out by B. Baker and Dr. Timothy Jensen, who furnished much of the computational results presented in this paper, as well as aiding in the formulation of many of the concepts.

In addition, the National Science Foundation supported part of this work under Grant GP-506. The computer time was furnished by Princeton University, supported in part by the National Science Foundation Grant NSF GP-579.

NOTATION

A = chemical species
$\mathbf{A}$ = coefficient matrix of dimension $n \times n$
B = chemical species
$\mathbf{B}$ = control coefficient matrix of dimension $n \times r$
$\mathbf{c}$ = constant vector of n elements
C = relative cost factor between solvent and feed
C_p = heat capacity
D_n = holding time of the n^{th} stage
E_i = heat of reaction of i^{th} chemical reaction
f = equilibrium functionality between raffinate and extract concentrations
$\mathbf{f}$ = nonlinear vector function of n variables
F = flow rate
G_i = pre-exponential term for i^{th} reaction
$\mathbf{h}$ = nonlinear vector function of n variables
H_i = enthalpy of i^{th} reaction
I = performance index
J = integrand for performance index
k = time step index
k_i = rate constant for i^{th} reaction
M_i = i^{th} cost factor
p_n = profit associated with the n^{th} stage
P = total profit
P = number of time stages
Q = extent of discretization of control values
Q_1 = flow rate of feed
Q_2 = flow rate of recycle stream
Q_n = heat added to the n^{th} stage
$\mathbf{Q}$ = weighting matrix in performance index
$\mathbf{R}$ = weighting matrix in performance index
t = time
t_f = final time
T_n = temperature of n^{th} stage
T = sampling time
$\mathbf{u}$ = control vector of r variables
u = scalar control function
v_n = normalized solvent flow rate into n^{th} stage; volume of n^{th} stage
w_n = solvent flow rate into the n^{th} stage
x_n = raffinate concentration leaving n^{th} stage
$\mathbf{x}$ = state vector of n variables

y_n = extract concentration leaving n^{th} stage
ρ = density

LITERATURE CITED

1. Baker, B., B.S. thesis, Princeton Univ., N. J. (1965).
2. Bellman, R., "Applied Dynamic Programming," Princeton Univ. Press, N. J. (1962).
3. Converse, A. O., *Chem. Eng. Progr. Symposium Ser. No. 46*, **59**, 127 (1963).
4. ———, and C. I. Huber, *Ind. Eng. Chem. Fundamentals*, **4**, 475 (1965).
5. Denn, Morton, and Rutherford Aris, *ibid.*, **4**, No. 7, 213, 248 (1965).
6. Dreyfus, S., *RAND Rept. P-2605-1* (Aug., 1963).
7. Fan, L., and C. Wang, "The Discrete Maximum Principle," Wiley, New York (1964).
8. Horn, F., and R. Jackson, *Ind. Eng. Chem. Fundamentals*, **4**, 110 (1965).
9. Jensen, Timothy, Ph.D. dissertation, Princeton Univ., N. J. (1964).
10. Lapidus, Leon, et al., *A.I.Ch.E. J.*, **7**, 288 (1961).
11. Lapidus, Leon, and Rein Luus, "Optimal Control of Chemical Engineering Systems," Blaisdell Publishing Co., New York (1967).
12. Lee, E. S., *A.I.Ch.E. J.*, **10**, 309 (1964).
13. Leitmann, G., ed., "Optimization Techniques," Academic Press, New York (1962).
14. Lovland, J., *Chem. Eng. Sci.*, **19**, 843 (1964).
15. Pontryagin, L. S., et al., "The Mathematical Theory of Optimal Processes," Interscience, New York (1962).
16. Rosen, E. M., *Chem. Eng. Sci.*, **19**, 999 (1964).

Manuscript received March 8, 1966; revision received July 11, 1966; paper presented July 11, 1966.

The Control of Nonlinear Systems.

Part II: Convergence by Combined First and Second Variations

REIN LUUS and LEON LAPIDUS

Princeton University, Princeton, New Jersey

It is shown that the first variation method can be used to provide an excellent starting condition for the second variation control of nonlinear systems. Numerical examples illustrate the basic features of the methods.

In Part I (*10*) we presented a simple and straightforward algorithm for determining the optimal control trajectory of nonlinear systems subject to control and state constraints. The basic concept involved a direct search on the control function. In the present paper we outline a more rational method for choosing the variations in the control function. The performance index is expanded to first- or second-order terms in the control vector; the direction in which the control vector is driven and its magnitude are then chosen so as to extremize the change in performance index at each step.

A method suitable for determining the optimal control trajectory which currently is in much favor is the *gradient* or *first variation* procedure (*9*). The logic in this method is easy to follow and programming on a digital computer is straightforward. A certain step is always taken in the *direction* of the most rapid change of the function to be extremized, and if the optimum region is overstepped, the step size is halved and the procedure is continued until some criterion is satisfied. However, the simplicity of the formulation is offset by two serious disadvantages. First, it is found in practice that a very large number of such steps or iterations must be taken before convergence occurs. Second, and more important, the trajectory *approaches* the optimum, but *does not actually reach it.* In some cases, after as many as 50 to 150 iterations, the trajectory is still far from the optimum and the rate of convergence becomes too slow to warrant further iterations.

To overcome these drawbacks of the first variation method Bryson et al. (*4, 12*), Kelley et al. (*7*), Merriam (*13, 14*) Kopp and McGill (*8*), and Jaswinski (*6*) have developed an iteration scheme which incorporates into the formulation not only the gradient of the function, but also the *curvature* or magnitude of the function. This formulation is called the *second variation method,* since second-order terms are used in the development. Convergence to the optimal trajectory now occurs in a quadratic fashion.

There are certain disadvantages of the second variation method which remove some of the theoretical attractiveness it may have. First, and most important, the initially assumed trajectory must be sufficiently close to the optimal trajectory for convergence to be obtained. Second, the number of equations to be integrated is considerably greater than required for the first variation method. Third, the equations themselves are more complicated. Hence, the programming of the iteration scheme with the required equations can be quite complicated. Finally, instability can arise from bad starting values, that is, from an insufficiently good guess for the initial trajectory.

In this paper we wish to show how the uncertainty of the choice of the initial trajectory can be removed by using the first variation method for the first few iterations and then changing over to the second variation method. This combination provides rapid convergence to the optimum from almost any realistic starting trajectory.

To illustrate the advantages of using the combined first and second variation methods, the control of a continuous stirred-tank reactor (CSTR) will be analyzed. In particular, we shall consider the control of the CSTR at a naturally unstable steady state. This choice will make the selection of the initial control policy very crucial if the second variation method is used alone. With the combined method, however, the choice of the initial control policy can be relaxed considerably.

Rein Luus is at the University of Toronto, Toronto, Canada.