THE OPTIMIZATION OF CHEMICAL REACTORS

by

CHEE-GEN WAN

B. S., National Taiwan University, 1959

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Chemical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1963

Approved by:

Liang-tseng Fan

Major Professor

TABLE OF CONTENTS

INTRODUCTION

In recent years, engineers have become more and more concerned with optimization. One of the reasons for this concern is that intensive competition in the industries in general and in the chemical process industries in particular makes it more necessary than ever that equipment and systems be designed and operated at peak performance. Even marginal savings can be extremely vital in this competitive environment (1).

A simple optimization problem corresponds to the findings of the extreme value of a function in calculus (2). This can be accomplished by differentiation. As the problem becomes more involved, partial differentiation and the calculus of variation (3) may have to be used. More often than not, optimal problems in engineering and industry cannot be solved by direct applications of these conventional mathematical methods. A variety of approaches more sophisticated than the conventional methods has been proposed to solve complex problems. Among them are Dynamic Programming (4) and the Maximum Principle (5).

Using the method of dynamic programming, Aris has solved a number of problems dealing with chemical reactor design (6); Aris and co-workers have solved a cross-current extraction problem (7); Rudd has investigated a reliability problem in chemical system design (8) and the optimal use of limited resources (9). Other applications of dynamic programming to chemical engineering problems can be found in the works of Roberts (10,11), Dranoff et al. (12), and Mitten et al. (13). Katz's algorithm of the discrete maximum principle has been successfully applied to several chemical processes in the works of Ahn et al. (14) and Wang (15).

In this report a brief review of these two approaches and their applications to several chemical reactor systems is presented.

DYNAMIC PROGRAMMING

The basic notion of the principle of optimality from which dynamic programming algorithm is derived is introduced by considering a simple example. The problem is formulated according to the dynamic programming algorithm. Two techniques frequently used to reduce complexities of problems -- the use of the Lagrange Multiplier and the ratio of the state variables -- are then introduced by again using examples.

Basic Problem--Minimum Length of Directed Network

In Figure 1, the circles represent the nodes, and the numbers on the lines which connect nodes denote the distances between the nodee. Suppose that the problem is to find the shortest path from node 1 to node 6 with an additional restriction that the passages of a particle should be allowed along any path only in the directions of increasing nodal index (16). Common sense tells us that we can enumerate all possible paths between node 1 and node 6, and then pick the shortest one. For convenience the number in the parentheses is used to denote the node number.

path (1)—(2)—(4)—(6) = 13

path (1)—(2)—(5)—(6) = 7

path (1)—(2)—(4)—(5)—(6) = 13

path (1)—(2)—(3)—(5)—(6) = 8

path (1)—(2)—(3)—(4)—(6) = 10

path (1)—(2)—(3)—(4)—(5)—(6) = 10

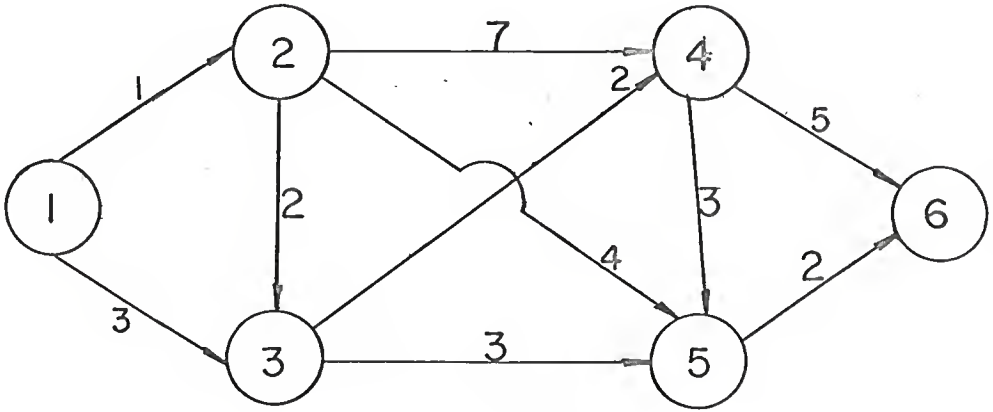

Fig. 1. Diagram of directed net works.

path (1)—(3)—(4)—(6) = 10

path (1)—(3)—(4)—(5)—(6) = 10

path (1)—(3)—(5)—(6) = 8

The shortest path is the second one listed, i.e., (1)—(2)—(5)—(6), whose total length is 7.

But enumeration of all the possibilities would be prohibitive in a problem of large size, employing dynamic programming treatment, we work backwards as follows:

from (5) to (6) = 2

from (4) to (6) = 5 either by way of (4) to (6) directly or (4)—(5)—(6). To go from (3) to (6) there are two choices, either by way of (4) or (5). If we proceed by way of (4), the distance from (3) to (6) is the sum of (3) to (4) and (4) to (6), that is, 2 + 5 = 7. This is true for both paths: (3)—(4)—(6) or (3)—(4)—(5)—(6). If we proceed by way of (5), we have: (3)—(5)—(6) = 5. Thus, the shortest path from (3) to (6) is: (3)—(5)—(6) = 5. By the same argument there are three alternatives leading from (2) to (6), namely, (2)—(4)—(6) = 12, (2)—(3)—(5)—(6) = 7 and (2)—(5)—(6) = 6. It follows that the shortest path from (2) to (6) (2)—(5)—(6) = 6. Starting from (1), we can go either by way of (2) or (3) to reach (6). The distance from (1) to (3) is 3 and the shortest distance from (3) to (6) is (3)—(5)—(6) = 5. As soon as the particle reaches (3), it must follow the shortest path from (3) leading to (6) in order to make the overall path the shortest. Therefore, the shortest path from (1) to (6) by way of (3) is: (1)—(3)—(5)—(6) = 8 and from (1) to (6) by way of (2) is (1)—(2)—(5)—(6) = 7. The choice of the shortest path from (1) to (6) will naturally be (1)—(2)—(5)—(6) = 7.

The main difference in the two methods is that in the first, the process of enumerating all the paths, we do not have any idea which path will eventually be the answer. We are only exhausting all the possible cases. However, the second method allows us to leave out all the impossible cases. For example, (2)—(5)—(6) = 6 is always the shortest path between (2) and (6) and, therefore, every time we proceed by way of (2), we discard all other paths branching out from (2).

What is illustrated above represents a very simple version of Bellman's Principle of Optimality (4) which states that "An optimal policy has the property that whatever the initial state and initial decisions are the remaining decisions must constitute an optimal polity with regard to the state resulting from the first decision." Dynamic programming is the algorithm based on this principle. The advantage of this approach is to reduce a problem with large dimensions to a problem of solving simultaneous equations each with small dimensions. Then the use of a high speed computer becomes particularly suitable.

Before describing in detail the formal algorithm of dynamic programming, a basic terminology should be presented (9,15).

1. State, x: The condition of the entry to be transformed in the process; for instance, the concentration of the reacting substances in the reactor, or the position of the node at which the particle is currently situated.
2. Decision, or control variable, θ: A decision or control variable is defined as a way of manipulation made to transform the entries from one state to another; for example, the temperature or pressure in a reactor, or the decision of the path to follow in the net-work problem.

3. Transformation, T: The transformation operator describes the way in which a given decision or control action transforms the entries from one state to another, for example; the kinetic equations or material balance equations and others.

4. Interval profit, P: The interval profit is the profit or any physical quantity resulting from the transformation of the entries in the interval time or space between two decisions. It is a function of the state and the decision made. An example, is the increase in concentration of a desired product in one backmix chemical reactor of a multistage process. Another example is the distance between two nodes in the interval.

5. Stage, n: This represents the typical unit of a process or the typical time periods, or any real or abstract notion of intervals in which cer- transformations take place.

6. Objective function, O: This is summation of the interval profits, it is the net profit of the entire process; for example, the total distance of the paths from node 1 to 6. Another example is the change in the concentration of the desired product between the first and last stages of a multistage process.

   The optimal sequence of decisions is defined as that sequence of decisions which maximizes the objective function.

7. Constraints: These are the limitations imposed on the decisions; examples are, only the path in the direction of increasing nodal index is permitted in the network problem or the limitations on the tempera- ture that may be imposed in designing a reactor, or the limitations of relative cost of the product.

According to the dynamic programming algorithm the net-work problem may be reformulated as follows (16):

Let f_i be the shortest path from node i to 6. P_{ij} is the distance from node i to j, and the constraints demand that $j>i$ and $P_{ij} = \infty$ when node i and j are not connected. Also, it can be noted that the distance from node 6 to itself is zero.

$$0_6 = 0 \tag{1}$$

In general

$$f_i = \min_{j>i} (P_{ij} + f_j) \tag{2}$$

To find the optimal objective function, in this case the shortest distance, $0^{*+} = f_i$, we employ the equation (2) repeatedly.

$$f_5 = \min_{j>5} (P_{5j} + f_j) = \min (P_{56} + f_6) = 2 + 0 = 2$$

This is to say that, if the system were in state 5, an optimal continuation would be to proceed to node 6, with a corresponding increase in length of 2 units. This continuation is designated by $j_5 = 6$.

$$f_4 = \min_{j>4} (P_{4j} + f_j) = \min \begin{pmatrix} P_{45} + f_5 \\ P_{46} + f_6 \end{pmatrix} = \min \begin{pmatrix} 3 + 2 \\ 5 + 0 \end{pmatrix} = 5$$

This shows that both paths are equal in distance and, therefore, $j_4 = 5$ or 6.

$$f_3 = \min \begin{pmatrix} P_{34} + f_4 \\ P_{35} + f_5 \end{pmatrix} = \min \begin{pmatrix} 2 + 5 \\ 3 + 2 \end{pmatrix} = 5 \qquad j_3 = 5$$

+The star, *, indicates optimum conditions in general. For instance, θ^* is the optimal decision and 0^* is the optimum profit, i.e. the objective function.

$$f_2 = \text{Min}\begin{pmatrix} P_{23} + f_3 \\ P_{24} + f_4 \\ P_{25} + f_5 \end{pmatrix} = \text{Min}\begin{pmatrix} 2 + 5 \\ 7 + 5 \\ 4 + 2 \end{pmatrix} = 6 \qquad j_2 = 5$$

$$0^* = f_1 = \text{Min}\begin{pmatrix} P_{12} + f_2 \\ P_{13} + f_3 \end{pmatrix} = \text{Min}\begin{pmatrix} 1 + 6 \\ 3 + 5 \end{pmatrix} = 7, \qquad j_1 = 2$$

Thus the objective function or profit is 7. The track of path will be $j_1 = 2$, $j_2 = 5$, $j_5 = 6$, i.e., (1)—(2)—(5)—(6). It is a well organized algorithm, producing the same result as previously obtained.

The example cited above has the property of being discrete which is indispensable in the dynamic programming algorithm. Although the problems encountered in engineering are often of a continuous nature, the discrete interval can often be made small enough to approach the continuous nature of the problem. This can be accomplished with the use of modern high speed computers.

The Exploitation Problem

The problem of exploiting a natural resource which is available for a total of N years will be considered next (9). The state x_n is defined as the amount of the resource remaining for processing with n years remaining. Suppose that at each year a decision θ concerning the method of processing for the next year is made and thus, altogether N decisions are made. The condition of the resource remaining for further exploitation and the profit made during exploitation are dependent upon the decision made at the present stage. For example, an oil reservoir may be initially water flooded to increase the rate of oil removal and the profit rate; but this early flooding may disturb the reservoir and in the long run reduce the total amount of oil that can be removed.

Representing the sequence of years in the order shown in Figure 2, the interval profit P_n is the profit made during the n-th period and is a function of the state of the resource at the beginning of the n-th stage and the decision made at that stage, i.e., $P_n = P_n(x_{n+1};\theta_n)$. As previously mentioned, the condition of the resource remaining for further exploitation depends on the decision made and also on the previous state. The relationship between them may be denoted by T, the transformation operator. T may be a complicated mathematical equation or just common sense; it may be expressed by $x_n = T(x_{n+1};\theta_n)$. The objective function is the summation of the interval profits

$$O(x_{in};\ \theta_N;\ \theta_{N-1},\ldots,\theta_1) = \sum_{n=N}^{1} P_n(x_n;\ \theta_n)$$

The value of the objective function depends on the initial state, x_{in}, of the resource and a sequence of N decisions, $\theta_N, \theta_{N-1},\ldots,\theta_1$. The problem is to make a series of decisions which will maximize the objective function. Such decisions are called the optimal policies denoted by $\theta_N^*, \theta_{N-1}^*,\ldots, \theta_1^*$. Thus

$$O^*(x_{in}) \equiv O(x_{in};\ \theta_N^*;\ \theta_{N-1}^*;\ldots,\ \theta_1^*) \equiv f_N(x_{in}) = \max_{\{\theta_n\}}\left\{\sum_{n=N}^{1} P(x_n;\ \theta_n)\right\} \quad (3)$$

$f_n(x_{n+1})$ is the total profit obtained over the time period n to 0 if an optimal policy is used starting at stage n. Clearly, if we find ourselves at a state x_{n+1} at a certain time or a stage n, the best we can possibly do is to follow the optimal policy from the state x_{n+1} to the end and thus obtain a profit, $f_n(x_{n+1})$, over that time period. Applying the principle of optimality the dynamic programming algorithm may be written as

$$f_n(x_{n+1}) = \max_{\theta_n}\left\{P_n(x_{n+1};\ \theta_n) + f_{n-1}(x_n)\right\} \quad (4)$$

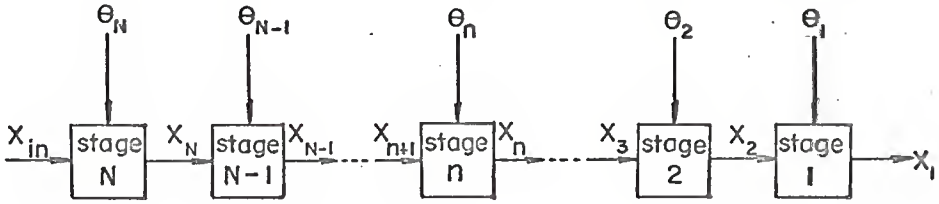

Fig. 2. Schematic diagram of multi-stage process.

$$x_n = T(x_{n+1};\ \theta_n) \tag{5}$$

$$0^*(x_{in}) \equiv 0^*(x_{N+1}) \equiv f_N(x_{N+1}) = \underset{\theta_N}{\text{Max}} \left\{ P_N(x_{N+1};\ \theta_N) + f_{N-1}(x_N) \right\}$$

If M different decisions are possible each year, the optimal policy and objective function, equation (3), may be reached by the repeated and simultaneous use of equations (4) and (5). The procedure starting from the last stage is illustrated as follows:

$$f_1(x_2) = \underset{\theta_1}{\text{Max}} \left\{ P_1(x_2;\ \theta_1) \right\} \tag{6}$$

$$x_1 = T(x_2;\ \theta_1) \tag{7}$$

First, for a selected value of x_2, the values of $P_1(x_2;\ \theta_1)$ are computed for M possible θ_1, which maximizes $P_1(x_2;\ \theta_1)$ is the optimal decision θ_1^* corresponding to the particular x_2 selected. The corresponding x_1 can be calculated by equation (7). Then the same calculation is repeated for all possible x_2. Eventually the so-called dynamic programming table may be constructed as shown in Table 1A. Proceeding to the second stage, equations (4) and (5) are written as

$$f_2(x_3) = \underset{\theta_2}{\text{Max}} \left\{ P_2(x_3;\ \theta_2) + f_1(x_2) \right\} \tag{8}$$

$$x_2 = T(x_3;\ \theta_2) \tag{9}$$

Again assigning a value to x_3, the values of $P_2(x_3;\ \theta_2)$ and x_2 are computed from equations (8) and (9) respectively. The value of $f_1(x_2)$ in Table 1A corresponding to the computed x_2 is added to $P_2(x_3;\ \theta_2)$. The optimum profit, $f_2(x_3)$, for the particular x_3 is the maximum among all the computed values

Table 1

Illustration of dynamic programming algorithm

Table 1.A

X_2	θ_1^*	$f_1(X_2)$	X_1
–	–	–	–
–	–	–	–
–	–	–	–
.	.	.	.
.	.	.	.
.	.	.	.

Table 1.B

X_3	θ_2^*	$f_2(X_3)$	X_2
–	–	–	–
–	–	–	–
–	–	–	–
.	.	.	.
.	.	.	.
.	.	.	.

Table 1.n-1

X_n	θ_{n-1}^*	$f_{n-1}(X_n)$	X_{n-1}
–	–	–	–
–	–	–	–
–	–	–	–
.	.	.	.
.	.	.	.
.	.	.	.

Table 1.n

X_{n+1}	θ_n^*	$f_n(X_{n+1})$	X_n
–	–	–	–
–	–	–	–
–	–	–	–
.	.	.	.
.	.	.	.
.	.	.	.

of $\left\{P_2(x_3;\ \theta_2) + f_1(x_2)\right\}$ for all possible θ_2. The succeeding θ_1^* and the x_1 correspondent to the computed x_2 are located in Table 1A. Table 1B is likewise constructed by repeating the above calculation for each possible x_3. In a similar fashion all N tables may be constructed sequentially. In general, an n-th stage table has the form of Table 1n. With a particular $x_{N+1} = x_{in}$ given, we proceed to Table N and find θ_N^* and x_N from it. Proceeding to the (N-1) stage, θ_{N-1}^* and x_{N-1} corresponding to the initial state, x_N are obtained, and this process can be continued until the entire sequence of the optimal decision, θ_N^*, $\theta_{N-1}^*,\ldots,\theta_1^*$, is extracted. This algorithm not only provides the answer to the particular $x_{N+1} = x_{in}$, but it also yields a series of solutions for a whole range of x_{in}, meaning that the dynamic programming tables contain solutions for M x N number of optimization problems. Knowing this is especially useful in chemical engineering problems. While the feed condition can hardly be kept at a steady state, we can determine the effect of putting on or taking off a few stages. Furthermore, the dynamic programming formulation not only allows a problem with high dimensionality to be replaced by a sequence of problems with lower dimensionality, but it also provides a straight forward computing algorithm. The device by which computational solution is brought about is through a system of recursive equations, which is well suited to high speed computer machine solution.

Another way of ordering the series for the stages for the problem just considered is presented in Figure 3. The algorithm is essentially the same as before, except that the direction is reversed in numbering the states and control variables. According to this scheme,

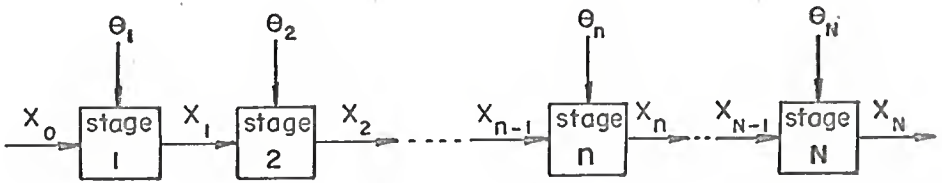

Fig. 3. Schematic diagram of multi-stage process.

$$0^* \equiv f_N(x_0) = \max_{\theta_1} \left\{ P_N(x_0;\ \theta_1) + f_{N-1}(x_1) \right\} \qquad (4a)$$

While the former way of numbering the system is more convenient for the purpose of a computer programming, each has its advantage. The latter enables us to compare dynamic programming with the maximum principle.

The dynamic programming algorithm, as has been shown, is convenient and powerful, but it is not a rote to be thoughtlessly applied to every problem. One of its best features is that it permits exercise of ingenuity in simplifying the complexities of a particular problem before applying the basic algorithm. Two schemes for reducing the complexities will be introduced by means of concrete examples.

Lagrange Multiplier

Quite often the use of the Lagrange Multiplier can reduce the dimensionality of a problem and enable the problem to be handled within the capacity of computers (17). Usually the Lagrange Multipliers are applied when there are inequality constraints or restrictions on the variables (17). There are two important rules in applying this technique (17): 1. The number of Lagrange Multipliers to be introduced must be equal to the number of constraining equations and 2. The Lagrange expression must be equal to the objective function plus the product of the Lagrange Multipliers and [illegible] the constraints. These constraints must be in the form of an equation which is equal to zero.

Suppose that the function $f(x,y,z) = x + 2y^2 + z^2$, subject to the constraint that $x + y + z - 1 = 0$, is to be minimized. The Lagrange expression (17) is:

$$w(x,y,z) = x + 2y^2 + z^2 + \lambda(x + y + z - 1)$$

Where λ is the Lagrange Multiplier. Consider λ as a new variable.

$$\frac{\partial W}{\partial x} = 1 + \lambda = 0$$

$$\frac{\partial W}{\partial y} = 4y + \lambda = 0$$

$$\frac{\partial W}{\partial z} = 2z + \lambda = 0$$

$$\frac{\partial W}{\partial \lambda} = x + y + z - 1 = 0$$

Solving these four equations simultaneously,

$$\lambda = -1, \quad x = 0.25, \quad y = 0.25, \quad z = 0.5$$

Thus, the minimum value of $f(x,y,z) \equiv x + 2y^2 + z^2 = (0.25) + 2(0.25)^2 + (0.5)^2 = 0.625$, under the constraint, $x + y + z - 1 = 0$.

Now the case with two types of resources represented by quantities x and y respectively will be considered (4). The quantities of these resources allocated to the n-th activity are designated by x_n and y_n respectively. If there exists an interval profit $P_n(x_n, y_n)$ which is the return from the n-th activity due to respective allocations of x_n and y_n, then the optimal problem is to maximize the objective function of 2N variables, i.e., the proper allocation of the resources for an efficient utilization. In other words, it is to maximize

$$O(x_1, x_2, \ldots, x_N; y_1, y_2, \ldots, y_N) = \sum_{n=1}^{N} P_n(x_n, y_n) \tag{10}$$

subject to the constraints

$$\begin{aligned} \sum_{n=1}^{N} x_n &= x \qquad x_n \geqq 0 \\ \sum_{n=1}^{N} y_n &= y \qquad y_n \geqq 0 \end{aligned} \tag{11}$$

This is a two-dimensional allocation process, independent of the value of N. Applying the dynamic programming algorithm,

$$O^*(x,y) \equiv f_N(x,y) = \underset{\{x,y\}}{\text{Max}} \; P_N(x_1, x_2, \ldots, x_N; y_1, y_2, \ldots, y_N)$$

x_n and y_n are values to be determined.

For N = 1

$$f_1(x,y) = P_1(x,y)$$

For $N \geq 2$, the following recurrence relation is obtained,

$$f(x,y) = \underset{0 \leq x_N \leq x}{\text{Max}} \; \underset{0 \leq y_N \leq y}{\text{Max}} \left[P_N(x_N, y_N) + f_{N-1}(x - x_N, y - y_N) \right]$$

The solution of this recurrence equation is quite involved and demands a large computer capacity. In the following treatment it will be shown how the Lagrange Multiplier can simplify the computation.

The following equation with constraints on x and y

$$O(x_1, \ldots, x_N; y_1, \ldots, y_N) = P_1(x_1, y_1) + P_2(x_2, y_2) + \ldots + P_N(x_N, y_N)$$

$$x_1 + x_2 + \ldots + x_N = x \quad , \quad x_n \geq 0$$

$$y_1 + y_2 + \ldots + y_N = y \quad , \quad y_n \geq 0$$

is replaced by the problem of maximizing the modified function

$$P_1(x_1, y_1) + P_2(x_2, y_2) + \ldots P_N(x_N, y_N) - \lambda \left[y_1 + y_2 + \ldots + y_N \right]$$

subject to the constraints

$$x_1 + x_2 + \ldots + x_N = x \quad , \quad x_n \geq 0 \tag{13}$$

$$y_n \geq 0 \tag{14}$$

For the moment consider λ as a fixed parameter and assume that

$$P_n(x_n, y_n)/y_n \to 0 \quad \text{as} \quad y_n \to \infty$$

The maximization over y_n can be done independently of the maximization over x_n (4). Thus

$$h_n(x_n, \lambda) = h_n(x_n) = \underset{y_n \geq 0}{\text{Max}} \left[P_n(x_n, y_n) - \lambda y_n \right]$$

The problem is transformed to that of maximizing the function

$$h_1(x_1) + h_2(x_2) + \ldots + h_N(x_N)$$

subject to the constraints of equations (13) and (14). Repeating the procedure by varying λ until it satisfies the restriction

$$\sum_{n=1}^{N} y_n(\lambda) = y$$

it may be found that the corresponding optimal policy is the solution for the problem.

Geometrically, the Lagrange Multiplier can be interpreted as the slope of the searching lines in the complex plane (4). As will be shown in the first example in the application section, the Lagrange Multiplier can also denote price or relative cost.

Ratio of State Variables

The term "homogeneous nature" implies that the transformation functions of the problem are linear with respect to each other (6). This linear property simplifies the optimization problem in that the ratio of

two state variables may be used as a new state variable, and thus the number of state variables is reduced (6).

As an example, the system of a first order consecutive reaction $A \rightarrow B \rightarrow C$, will be considered.

Referring to Figure 2 the concentrations of A and B which are denoted by x and y respectively, are the state variable and the temperature is the controlling variable. The n-th reactor has the volume V_n and the volumetric flow rate of the stream v is constant throughout the multistage system. It will be assumed that there is no density change in the process, which is essentially true for the liquid phase reactions. The holding time is $\theta_n = \frac{V_n}{V}$.

Since the reaction under consideration is a first order consecutive reaction represented by

$$A \xrightarrow{k(t)} B \xrightarrow{h(t)} C$$

and the kinetic equations for rate of appearance of A and B are

$$r_A = - k(t)x$$

$$r_B = k(t)x - h(t)y$$

the material balance around the n-th stage for each species is:

$$vx_{n+1} = vx_n + (-r_A) V_n$$

or

$$x_{n+1} = x_n + k(t_n) x_n \theta_n \qquad (15)$$

and

$$vy_{n+1} = vy_n + (-r_B)\, V_n$$

or

$$y_{n+1} = y_n + h(t_n)\, y_n - k(t_n)\, x_n\, \theta_n \tag{16}$$

The production of y for a definite feed condition is to be maximized by choosing the optimal series of decisions, θ_n and t_n. It can be seen that in order to describe the system completely the composition, x and y, must be individually specified at each stage. This is a two-dimensional problem which not only requires complex computational scheme but also demands a great deal of computer memories. However, as we examine closely, we can see that because of the linear transformation equations involved, the problem can be simplified by defining a new variable

$$e_n = \frac{x_n}{y_n} \tag{17}$$

Dividing equation (16) by equation (15)

$$\frac{y_{n+1}}{x_{n+1}} = \frac{y_n + \theta_n\left[h(t_n)\, y_n - k(t_n)\, y_n\right]}{x_n + x_n\, \theta_n\, k(t_n)}$$

$$e_{n+1} = \frac{e_n + \theta_n\left[e_n\, h(t_n) - k(t_n)\right]}{1 + \theta_n\, k(t_n)}$$

$$e_{n+1} = e_n \left\{\frac{1 + \theta_n\, h(t_n)}{1 + \theta_n\, k(t_n)}\right\} - \frac{\theta_n\, k(t_n)}{1 + \theta_n\, k(t_n)}$$

$$e_n = \left\{e_{n+1} + \frac{\theta_n\, k(t_n)}{1 + \theta_n\, k(t_n)}\right\}\left\{\frac{1 + \theta_n\, k(t_n)}{1 + \theta_n\, h(t_n)}\right\}$$

$$e_n = \left\{e_{n+1}\left[1 + \theta_n\, k(t_n)\right] + \theta_n\, k(t_n)\right\} / \left[1 + \theta_n\, h(t_n)\right] \tag{18}$$

This equation gives a functional relation for determining the output ratio e_n at any stage in terms of the feed ratio e_{n+1}. From equations (15) and (16)

$$\frac{y_n - y_{n+1}}{x_{n+1}} = \frac{\theta_n \left(k(t_n)\, x_n - h(t_n)\, y_n \right)}{x_n + x_n\, k(t_n)\, \theta_n}$$

$$= \frac{\theta_n\, k(t_n) - \theta_n\, h(t_n)\, e_n}{1 + \theta_n\, k(t_n)} \qquad (19)$$

Since the formation of B is to be maximized, the objective function is:

$$y_1 - y_{n+1} = \sum_{1}^{N} (y_n - y_{n+1}) \qquad (20)$$

and the optimum function, $f_N(x_{N+1};\, y_{N+1})$, is

$$f_N(x_{N+1};\, y_{N+1}) = \underset{\{t,\theta\}}{\text{Max}} \sum_{1}^{N} (y_n - y_{n+1}) \qquad (21)$$

Because of the linear nature of the equations, this must be proportional to x_{N+1} and depends only on the ratio $y_{N+1}/x_{N+1} = e_{N+1}$, i.e.

$$f_N(x_{N+1};\, y_{N+1}) \equiv x_{N+1}\, g_N(e_{N+1}) \qquad (22)$$

Now applying the dynamic programming algorithm to the function f_N

$$f_N(x_{N+1}; y_{N+1}) = \underset{t_N, \theta_N}{\text{Max}} \left\{ (y_N - y_{N+1}) + f_{N-1}(x_N;\, y_N) \right\} \qquad (23)$$

Combing equations (15), (19), (22) and (23)

$$g_N(e_{N+1}) = \underset{t_N, \theta_N}{\text{Max}} \left\{ \frac{y_N - y_{N+1}}{x_{N+1}} + \frac{1}{x_{N+1}}\; x_N\, g_{N-1}(e_N) \right\}$$

$$g_N(e_{N+1}) = \max_{t_N, \theta_N} \left\{ \frac{\theta_n k(t_n) - \theta_n h(t_n) e_n}{1 + \theta_n k(t_n)} + \frac{1}{1 + \theta_n k(t_n)} g_{N-1}(e_N) \right\} \quad (24)$$

This is an equation in e only, and can be solved recursively in conjunction with equation (18).

DISCRETE MAXIMUM PRINCIPLE

In this section, the general maximum principle algorithm for discrete systems with recycle is presented first (15). It is then reduced to a special form with no recycle.

A series of multi-stage ideal backmix reactors with recycle from the last stage to the first stage will be considered. They are schematically shown in Figure 4. If the superscript, n, indicates the stage number, x_i^n's, i = 1, 2,...,s, are the state variables leaving the n-th stage and θ_r^n, r = 1, 2,...,t, are the control variables at the n-th stage. If T_i is the transformation function for each different i, then

$$x_i^n = T_i^n(x_k^{n-1}; \theta_r^n) \quad (25)$$

$$i = 1, 2, \ldots, s$$

$$n = 1, 2, \ldots, N$$

where $T_i^n(x_k^{n-1}; \theta_r^n)$ is a short hand form for $T_i(x_1^{n-1}; x_2^{n-1}, \ldots, x_s^{n-1}; \theta_1^n, \theta_2^n, \ldots, \theta_t^n)$.

The feed enters the system at a rate q, while the recycle stream is fed back at a rate r. The mixing conditions are represented by

$$x_i^o = M_i^o(x_k^f; x_k^N) \quad , \quad i = 1, 2, \ldots, s \quad (26)$$

where x_k^f are the state variables in the feed.

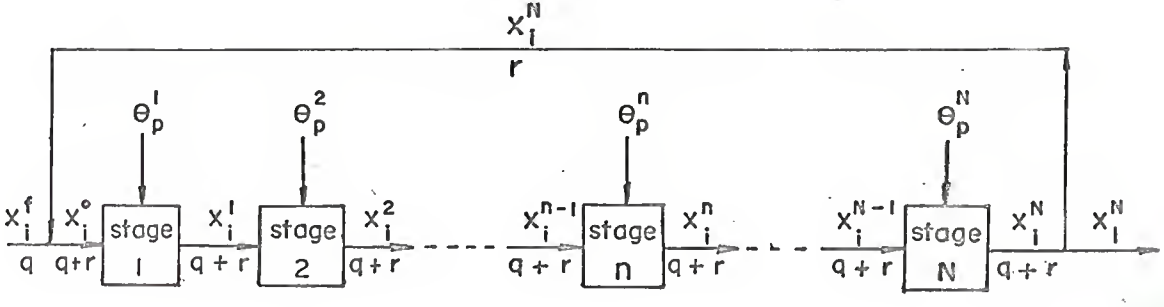

Fig.4. Schematic representation of multi-stage process with recyle.

The problem is to find a sequence of control decisions $\left\{\theta_r^n\right\}$, with or without the constraints, $\alpha_r^n \leq \theta_r^n \leq \beta_r^n$; n = 1, 2,...,N; r = 1, 2,...,t, which will maximize x_m^N.

The essence of the discrete algorithm for the maximum principle (5) is to introduce a new set of variables defined by

$$z_i^{n-1} = \sum_{j=1}^{s} \frac{\partial T_j^n(x_k^{n-1};\ \theta_r^n)}{\partial x_1^{n-1}}\ z_j^n \tag{27}$$

$$i = 1, 2,...,s$$

$$n = 1, 2,...,N$$

The Hamiltonian H^n is formed as follows:

$$H^n = \sum_{j=1}^{s} z_j^n\ T_j^n(x_k^{n-1};\ \theta_r^n) \tag{28}$$

It can be shown that (5) (15)

$$x_i^n = \frac{\partial H^n}{\partial z_i^n} \tag{29}$$

$$z_i^{n-1} = \frac{\partial H^n}{\partial x_1^{n-1}} \tag{30}$$

Then the optimization problem becomes that of finding a sequence of $\left\{\theta_r^n\right\}$ to satisfy the following conditions (5,15)

$$H^n = \sum_{j=1}^{s} z_j^n\ T_j^n(x_k^{n-1};\ \theta_r^n) = \text{Maximum} \tag{31}$$

with the boundary conditions (15)

$$x_i^0 = M_i^0(x_k^f;\ x_k^N) \tag{32}$$

$$z_i^N - \sum_{j=1}^{s} z_j^{\theta} \frac{\partial T_j(x_k^f;\, x_k^N)}{\partial x_i^N} = \delta_{im} \qquad (33)$$

$$n = 1, 2, \ldots, N$$

$$i = 1, 2, \ldots, x$$

$$\delta_{im} = \begin{cases} 1 & i = m \\ 0 & i \neq m \end{cases}$$

where δ_{im} is the Kronecker delta, i.e. $\delta_{im} = \begin{cases} 1 & i = m \\ 0 & i \neq m \end{cases}$.

If the minimizing sequence instead of the maximizing sequence of control actions is to be decided, the basic algorithm remains unchanged except that the equation $H^n = \max$ is replaced by $H^n = \min$.

Quite often the maxima sought in equation (31) are found to lie at stationary positions. This is equivalent to

$$\theta_r^n ; \sum_{j=1}^{s} z_j^n \frac{\partial T_j^n(x_k^{n-1};\, \theta_r^n)}{\partial \theta_r^n} = 0 \qquad (34)$$

If the θ's are under the constraints

$$\alpha^n \leq \theta_r^n \leq \beta^n$$

then, for each n, the maximum required by equation (31) may be found at some stationary place between α^n and β^n or at one of the end points, α^n or β^n. In this situation, the following equations must be used:

$$\theta_r^N : \sum_{j=1}^{s} z_j^n \frac{\partial T_j^n(x_k^{n-1};\, \theta_r^n)}{\partial \theta_r^n} = 0; \quad \alpha^n < \theta^n < \beta^n$$

$$\text{or} : \theta_r^n = \alpha^n$$

or : $\theta_r^n = \mathcal{B}^n$

whichever makes H^n the maximum.

This algorithm can seldom be solved analytically (15). Even in simple cases considerable ingenuity is needed. Therefore, only a very general working procedure for finding a numerical answer to a specific problem according to the algorithm is presented:

1. Propose a sequence θ_r^1 , θ_r^2,... θ_r^n
2. Propose a set of x_i^N. $i = 1, 2, \ldots, s$
3. With a given x_i^f, solve equations (26) and (25) forward and get a new set of x_i^N. Repeat the calculating using this new set of x_i^N until the subsequent x_i^N are close enough.
4. With these x_i^n and θ_r^n, solve the equations (27) and (33) in the z's backward from $n = N$ to $n = 1$.
5. With these x's and z's compute a sequence of θ_r^n from equation (31) and return to step 2.
6. The iteration procedure is terminated when the successive computed values of x_m^N are sufficiently close, possibly together with the condition that successive computed strings of θ are also close enough.

Quite often a modification in practice may save considerable work. Instead of taking θ computed in step 5 directly back to step 2, it may sometimes be desirable to regard the new θ as a mere indication of direction in θ-space and apply the methods appropriate to steepest descent calculations to choose the best length of step in this direction, and possibly even to modify the direction in the light of the iterative history.

For the process without product recycle, i.e., r = 0, equation (32) reduces to

$$x_i^0 = x_i^f \tag{36}$$

$$i = 1, 2, \ldots, s$$

and the basic algorithm reduces to

$$z_i^{n-1} = \sum_{j=1}^{s} \frac{\partial T_j^n(x_k^{n-1};\ \theta_r^n)}{\partial x_i^{n-1}}\, z_j^n \tag{27}$$

$$H^n = \sum_{j=1}^{s} z_j^n\, T_j^n(x_k^{n-1};\ \theta_r^n) \tag{28}$$

$$x_i^n = \frac{\partial H^n}{\partial z_i^n} \tag{29}$$

$$z_i^n = \frac{\partial H^n}{\partial x_i^{n-1}} \tag{30}$$

$$n = 1, 2, \ldots, N \quad r = 1, 2, \ldots, t \quad i = 1, 2, \ldots, s$$

θ_r^n are determined where H^n = max, with the boundary conditions

$$x_i^0 = x_i^f \tag{36}$$

$$z_i^N = \begin{cases} 1, & i=m \\ 0, & i \neq m \end{cases} \tag{37}$$

The general working procedure is the same as stated for the case with product recycle except that the step of guessing a set of x_i^N may be omitted saving considerable labor in matching the x_i^N by trial and error.

It is worthwhile to note that for each assigned value of x_1^N, the corresponding calculated values of x_1^n, n = 0, 1, 2,...,N-1 are the optimal state variables corresponding to the initial condition x_1^0 computed in each run of trial calculation.

If the final values of certain x's other than x_m are specified, the boundary conditions of equation (37) are changed to:

$$z_i^N = \begin{cases} 1, & i=m \\ 0, & i\neq m,p \end{cases} \tag{38}$$

$$x_p^N = b \tag{39}$$

Otherwise the algorithm is the same as before.

APPLICATIONS

Cross-current Elutriation of Sludge

The purpose of elutriation in the sludge treatment is to reduce the alkalinity of the sludge with wash water or wash solution (18). Here, the term "sludge" indicates the sludge in wet slurry form after the filtration process. The alkalinity of the sludge is usually expressed as mg/liter of calcium carbonate equivalent to the determined alkalinity. The elutriation can be carried out practically by repeated cross-current washings as shown in Figure 5 (18). The sludge is fed at a constant rate corresponding to liter of water per unit time and with alkalinity E_o mg/liter. In the tank, it is kept in suspension by air or mechanical agitation. After thorough washing, the sludge and wash water are separated. The out wet sludge solids and wash water should have the same alkalinity. The outlet sludge solids become the feed to the next tank. The wash water is available at the alkalinity w mg/liter. It is fed into the tank at the rate of R liter per unit time.

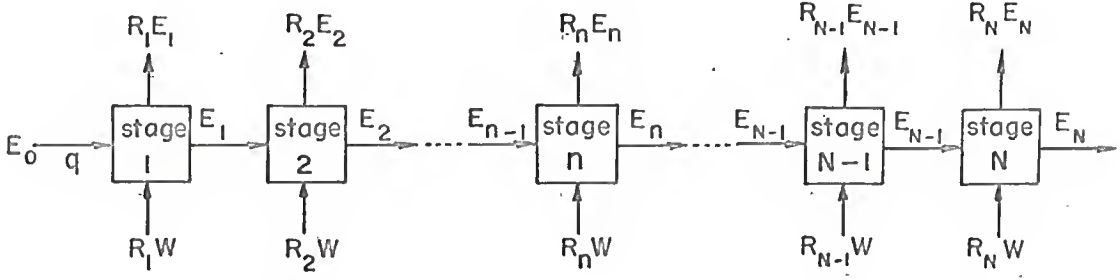

Fig. 5. Cross-current elutriation of sludge.

For a system with a finite number of stages and a definite input alkalinity E_0, two types of problems may arise:

1. With a fixed total amount of wash water Q_N, how can this amount of wash water be distributed amoung the stages in order to maximize the reduction in alkalinity, E_0-E_N, of the sludge?

2. If the cost of providing and pumping the wash water is a fraction, λ, of the value to be gained by reducing the alkalinity of the sludge, how should the flow rate of wash water to each stage be regulated to maximize the net profit, $E_0 - E_N - \lambda Q_N$?

It will be shown later that the solution of the second problem also yields the solution of the first and is, in fact, the easiest method of solving the problem. The realtive cost, λ, plays the role of the Lagrange multiplier (4,7).

It will be shown below that dynamic programming and the maximum principle algorithms yield the same solution for this particular problem. The optimal decision, as will be shown later, is to distribute the wash water evenly among all the stages.

For the convenience of formulating the problem, the following notations are defined:

$$E_0 \equiv a$$

$$E \equiv x \qquad (40)$$

It can be recognized that

$$\theta = \frac{R}{q}$$

The physical condition demands that

$$a \geq x_1 \geq \cdots \geq x_n \geq \cdots \geq x_N \geq w$$

Dynamic Programming Solution. The material balance around the n-th stage with respect to alkalinity yields

$$q E_{n-1} + R_n W = R_n E_n + q E_n$$

$$x_{n-1} + \theta_n W = \theta_n x_n + x_n$$

$$x_n = \frac{x_{n-1} + \theta_n W}{1 + \theta_n} \tag{41}$$

This represents the transformation function, $T(x_n; \theta_n)$.

Interval profit per unit volume of the wet sludge at the n-th stage is,

$$P_n = x_{n-1} - x_n - \lambda \theta_n$$

Therefore, according to equation (4a) and Figure 3,

$$0^* \equiv f_N(a) = \underset{\{\theta_n\}}{\text{Max}} \left\{ a - x_N - \lambda \sum_{n=1}^{N} \theta_n \right.$$

$$= \underset{\{\theta_n\}}{\text{Max}} \left\{ \sum_{n=1}^{N} (x_{n-1} - x_n - \lambda \theta_n) \right\}$$

$$= \underset{\theta_1}{\text{Max}} \left\{ P_N(a, \theta_1) + f_{N-1}(x_1) \right\}$$

$$f_N(a) = \underset{\theta_1}{\text{Max}} \left\{ a - x_1 - \lambda \theta_1 + f_{n-1}(x_1) \right\} \tag{42}$$

Clearly, if there is no stage existing, the profit will be zero, i.e.

$$f_0(a) = 0 \tag{43}$$

For the one-stage system, shown in Fig. 6A, equation (42) is reduced to,

$$f_1(a) = \underset{\theta_1}{\text{Max}} \left\{ a - x_1 - \lambda \theta_1 + f_0(x_1) \right\}$$

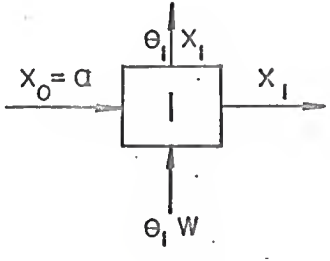

Fig.6A. Diagram of one-stage system.

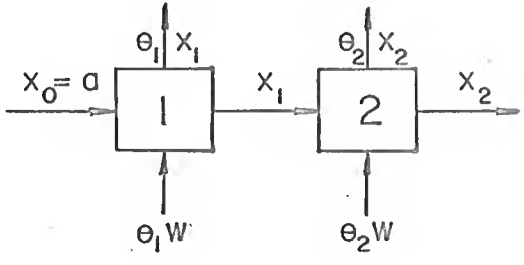

Fig.6B. Diagram of two-stage system.

Combining this with equations (43) and (41)

$$f_1(a) = \max_{\theta_1} \left\{ a - \frac{a + \theta_1 w}{1 + \theta_1} - \lambda\theta_1 \right\} \tag{44}$$

By straightforward differentiation, it yields the optimal decision θ_1 as

$$\theta_1 = \left(\frac{a - w}{\lambda}\right)^{\frac{1}{2}} - 1 \tag{45}$$

Inserting equation (45) into equations (41) and (44)

$$x_1 = \left[\lambda(a - w)\right]^{\frac{1}{2}} + w \tag{46}$$

$$f_1(a) = a - w + \lambda - 2\lambda^{\frac{1}{2}}(a - w)^{\frac{1}{2}} \tag{47}$$

For the optimal two-stage policy, referring to Figure 6B and employing equation (42),

$$f_2(a) = \max_{\theta_1} \left\{ a - x_1 - \lambda\theta_1 + f_1(x_1) \right\} \tag{48}$$

Combining this equation with equations (41) and (47)

$$f_2(a) = \max_{\theta_1} \left\{ a - x_1 - \lambda\theta_1 + x_1 + \lambda - w - 2\left[\lambda(x_1 - w)\right]^{\frac{1}{2}} \right\}$$

$$f_2(a) = \max_{\theta_1} \left\{ a + \lambda - w - \lambda\theta_1 - 2\left[\frac{\lambda(a - w)}{1 + \theta_1}\right]^{\frac{1}{2}} \right\} \tag{49}$$

By direct differentiation the optimum point is found as

$$\theta_1 = \left(\frac{a - w}{\lambda}\right)^{\frac{1}{3}} - 1 \tag{50}$$

$$x_1 = \left[\lambda(a - w)^2\right]^{\frac{1}{3}} + w \tag{51}$$

$$f_2(a) = a - w + 2\lambda - 3\lambda^{\frac{2}{3}} (a - w)^{\frac{1}{3}} \tag{52}$$

According to the principle of optimality the second tank should use the optimal one-stage policy with respect to its input x_1. Therefore, from equation (45)

$$\theta_2 = \left(\frac{x_1 - w}{\lambda}\right)^{\frac{1}{2}} - 1 \tag{45a}$$

Inserting equation (51) into the equation above

$$\theta_2 = \left(\frac{a - w}{\lambda}\right)^{\frac{1}{3}} - 1 \tag{53}$$

Comparing this with equation (50)

$$\theta_2 = \theta_1$$

Thus for the two-stage system, it has been proved that the optimal policy is to divide the wash water equally between the two stages.

In general for the N-stage system it can be deduced that

$$f_N(a) = a - w + N\lambda - (N+1)\lambda^{\frac{N}{N+1}} (a-w)^{\frac{1}{N+1}} \tag{54}$$

$$\theta_N = \left(\frac{a - w}{\lambda}\right)^{\frac{1}{N+1}} - 1 \tag{55}$$

$$\theta_n = \theta_{n-1} \qquad n = 1, 2, \ldots, N$$

The optimal policy is always to distribute the wash water evenly among all the stages.

Maximum Principle Solution. Thie problem is a simple case without recycle. Following the notation of the maximum principle algorithm and from equation (41)

$$x_1^n = T_1(x_k^{n-1};\ \theta^n) = \frac{x_1^{n-1} + \theta^n w}{1 + \theta^n} \tag{56}$$

Defining

$$x_2^n = T_2(x_k^{n-1};\ \theta^n) = x_1^{n-1} + x_1^{n-1} - x_1^n - \lambda\theta^n \tag{57}$$

it can be shown that $x_2^N = \sum_{n=1}^{N} (x_1^{n-1} - x_1^n - \lambda\theta^n)$ is the objective function to be maximized. The problem is then to maximize x_2^N with the boundary conditions

$$\left.\begin{aligned} x_1^0 &= a \\ x_2^0 &= 0 \end{aligned}\right\} \tag{58}$$

From equation (27)

$$z_1^{n-1} = \frac{z_1^n}{1 + \theta^n} + z_2^n \left[1 - \frac{1}{1 + \theta^n} \right] \tag{59}$$

$$z_2^{n-1} = z_2^n \tag{60}$$

$$n = 1, 2, \ldots, N$$

From equation (37)

$$\left.\begin{aligned} z_1^N &= 0 \\ z_2^N &= 1 \end{aligned}\right\} \tag{61}$$

From equations (59), (60) and (61)

$$z_2^{n-1} = z_2^n = 1 \qquad n = 1, 2, \ldots, N \tag{62}$$

$$z_1^{n-1} = \frac{z_1^n}{1 + \theta^n} + \frac{\theta^n}{1 + \theta^n} \tag{63}$$

From equations (28), (56) and (62)

$$H^n = z_1^n \left(\frac{x_1^{n-1} + \theta^n w}{1 + \theta^n}\right) + z_2^n\left(x_2^{n-1} + x_1^{n-1} - \frac{x_1^{n-1} + \theta^n w}{1 + \theta^n} - \lambda\theta^n\right)$$

$$H^n = (z_1^n - 1)\left(\frac{x_1^{n-1} + \theta^n w}{1 + \theta^n}\right) + x_2^{n-1} + x_1^{n-1} - \theta^n$$

$$\frac{\partial H^n}{\partial \theta^n} = \frac{(z_1^n - 1)(w - x_1^{n-1})}{(1 + \theta^n)^2} - \lambda \qquad (64)$$

Since λ imposes a restriction on θ^n automatically the nature of the physical situation assures the existence of the extremum points at which θ^n make equation (64) equal to zero (15). Thus

$$\theta^n = \left[\frac{(w - x_1^{n-1})(z_1^n - 1)}{\lambda}\right]^{\frac{1}{2}} - 1 \qquad (65)$$

Solving equation (65) for z_1^n

$$z_1^n = \frac{(1 + \theta^n)^2 \lambda}{w - x_1^{n-1}} + 1 \qquad (66)$$

For the (n+1)-th stage

$$z_1^{n+1} = \frac{(1 + \theta^{n+1})^2 \lambda}{w - x_1^n} + 1 \qquad (66a)$$

From equation (63)

$$z_1^n = \frac{z_1^{n+1}}{1 + \theta^{n+1}} + \frac{\theta^{n+1}}{1 + \theta^n}$$

Substituting equations (66) and (66a) into this equation.

$$\frac{(1 + \theta^n)^2}{w - x_1^{n-1}} = \frac{(1 + \theta^{n+1})}{w - x_1^n} \qquad (67)$$

From equation (56)

$$\theta^n + 1 = \frac{x_1^{n-1} - w}{x_1^n - w} \tag{68}$$

$$\theta^{n+1} + 1 = \frac{x_1^n - w}{x_1^{n+1} - w} \tag{68a}$$

Combining equations (67), (68) and (68a)

$$(x_1^{n-1} - w)(x_1^{n+1} - w) = (x_1^n - w)^2 \tag{69}$$

The general solution of this finite difference equation is (19):

$$x_1^n = c(d)^n + w, \quad n = 0, 1, \ldots, N+1 \tag{70}$$

From the boundary conditions, the constants, c and d, can be evaluated.

From equation (58)

$$c = a - w \tag{71}$$

Equation (69) then, becomes

$$x_1^n = (a - w)(d)^n + w \tag{72}$$

At the N-th stage, by equations (68) and (68a)

$$\theta^N + 1 = \frac{x_1^{N-1} - w}{x_1^N - w} \tag{68b}$$

Combining equations (61) and (66)

$$\theta^N + 1 = \left(\frac{x_1^{N-1} - w}{\lambda}\right)^{\frac{1}{2}} \tag{73}$$

Thus:

$$(x_1^N - w)^2 = (x_1^{N-1} - w)\,\lambda \tag{74}$$

Inserting equation (72) into equation (74) yields

$$d = \left(\frac{\lambda}{a-w}\right)^{\frac{1}{N+1}} \tag{75}$$

The general solution for x_1^n is then obtained by substituting equation (75) into equation (72)

$$x_1^n = (a - w)\left(\frac{\lambda}{a - w}\right)^{\frac{n}{N+1}} + w \tag{76}$$

Inserting x_1^n from equation (76) into equation (56)

$$\theta^n = \left(\frac{a - w}{\lambda}\right)^{\frac{1}{N+1}} - 1 \tag{77}$$

$$x_1^N = (\lambda)^{\frac{N}{N+1}}(a - w)^{\frac{1}{N+1}} + w \tag{78}$$

By a simple summation of equation (57)

$$x_2^N = x_2^0 + x_1^0 - x_1^N - N\lambda\theta^N \tag{79}$$

Combining equation (79) with equations (58), (76), (78) and (77)

$$x_2^N = a - w + N\lambda - (N+1)(\lambda)^{\frac{N}{N+1}}(a - w)^{\frac{1}{N+1}} \tag{80}$$

Since θ^n is independent of stage number, as can be seem from equation (77), the optimal policy is to distribute the wash water evenly among all the stages.

For this simple problem, both the dynamic programming and maximum principle approaches result in identical analytic solutions although the latter approach is more direct than the former.

These solutions are for the second type of the problems listed at the beginning of this section, but it also contains solutions for the first type, i.e., the case with fixed Q_N. Since the optimal policies for different values of λ, which is a paramter, give $Q_N = \sum_{n=1}^{N} Q^n = Q_N(\lambda)$, by searching through different λ, such λ can be found that $Q_N(\lambda)$ matches the given Q_N as described in the previous section describing the use of the Lagrange multiplier.

Step Rocket Problem

A step rocket is a multistage rocket system employing the technique of jettisoning structural weight during flight to increase performance. Figure 7 is a sketch of a three-stage step rocket system (20). The rocket weight is defined as the initial gross weight at the start of a stage which is the period of time starting just after the jettison of one structural package and lasting until just after the jettison of the next. The difference of weight between two consecutive stages is called the step weight, and weight remaining after the jettison of the last stage, the rocket payload.

The purpose of the rocket system is to attain a specified final velocity carrying a desired rocket payload (21). If the number of stages and the material and propellant for constructing the rocket system have been decided, the problem is to determine the optimum size of each stage so that the gross weight of the system is minimized. A two-stage optimization problem has been solved by Goldsmith (22) by using calculus and a three-stage problem by Schurmann (23). To make hand calculation possible, certain simplifying assumptions have been made in their solutions. For many problems the assumptions are quite reasonable and the results are found to compare favorably with actual practice (20). The simplifying assumptions, however, must be

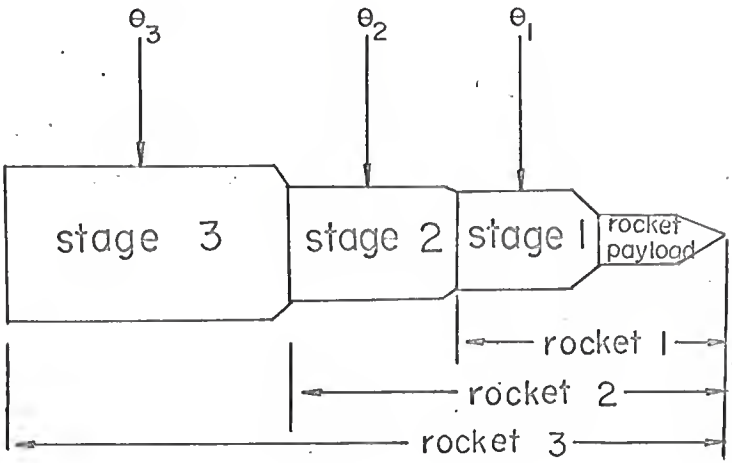

Fig.7. A sketch of three stage step rocket system.

verified by comparison with solutions obtained by removing all or some of the assumptions. Also, many practical problems may have to be solved by use of other approaches.

Dyke (20) employed the dynamic programming algorithm to solve the problem, which will be summarized below. A working scheme suitable for the machine computation of this algorithm and the maximum principle solution are also presented in this section. These approaches not only can be used to check the simplified solutions mentioned before but should also yield reliable results for rocket systems with more than three-stages.

Before introducing the general transformation equation, the following terms will be defined (20):

c_n = propellant exhaust velocity of stage n

$f_n(V)$ = minimum weight of rocket n achieving velocity V

M_n = ratio of initial thrust to weight of rocket n

n = number of stages

V_n = velocity ideally added by the rocket n; design velocity of rocket n

W_L = rocket system payload weight

w_n = initial gross weight of step n

w_{xn} = that portion of stage n jettison weight which is constant

$\alpha_n W_n$ = portion of stage n jettison weight dependent of step weight

$\beta_n M_n x_n$ = portion of stage n jettison weight dependent on thrust

$\sigma_n W_n$ = total jettison weight of stage n

x_n = initial gross weight of rocket n

θ_n = velocity ideally added during stage n

$$a_n = -\beta_n M_n - \alpha_n$$

$$b_n = 1 - \alpha_n$$

$$d_n = w_{xn}$$

It is assumed that M_n is constant for the stage n. The following relation holds for each stage (20).

$$\sigma_n w_n = \alpha_n w_n + \beta_n M_n x_n + w_{xn} \tag{81}$$

i.e. for stage n, total jettison weight is the sum of the jettison weight which depends on the step weight and step thrust plus a constant term.

This assumption is quite reasonable (20). α_n, β_n and w_{xn} are constraints for n-th stage. From the definition of x_n

$$x_{n+1} = x_n + w_n \tag{82}$$

The well-known rocket equation yields (20)

$$\frac{x_n}{x_{n-1} + \sigma_n w_n} = \exp\left(\frac{\theta_n}{c_n}\right) \tag{83}$$

Combining equations (81), and (82) and (83)

$$x_n = \frac{(1 - \alpha_n) x_{n-1} + w_{xn}}{\exp\left(-\frac{\theta_n}{c_n}\right) - \beta_n M_n - \alpha_n}$$

$$x_n = \frac{b_n x_{n-1} + d_n}{\exp\left(-\frac{n}{c_n}\right) + a_n} \tag{84}$$

Dynamic Programming Solution (20). Applying equation (4)

$$f_n(V_n) = \underset{0 \le \theta \le V_n}{\text{Min}} \left\{ \frac{d_n + a_n f_{n-1}(V_{n-1})}{\exp\left(-\frac{\theta_n}{c_n}\right) + a_n} \right\} \tag{85}$$

$$V_{n-1} = V_n - \theta_n \tag{86}$$

For the last stage

$$f_1(V_1) = \frac{d_1 + a_1 W_L}{\exp\left(-\frac{V_1}{c_1}\right) + a_1} \tag{87}$$

An outline of the computer flow chart for solving these recurrence relations is shown in Figure 8 and a Fortran 1620 program is given in Appendix A.

Maximum Principle Solution. Applying the maximum principle algorithm, the problem can be reformulated as shown in Figure 9 which corresponds to Figure 7. From equation (84)

$$x_1^n = \frac{b^n x_1^{n-1} + d^n}{\exp\left(-\frac{\theta^n}{c^n}\right) + a^n} \tag{84a}$$

Defining

$$x_2^n = x_2^{n-1} - \theta^n \tag{88}$$

the problem is then to minimize x_1^N with the following boundary conditions

$$\left.\begin{aligned} x_1^0 &= W_L \\ x_2^0 &= V \\ x_2^N &= 0 \end{aligned}\right\} \tag{89}$$

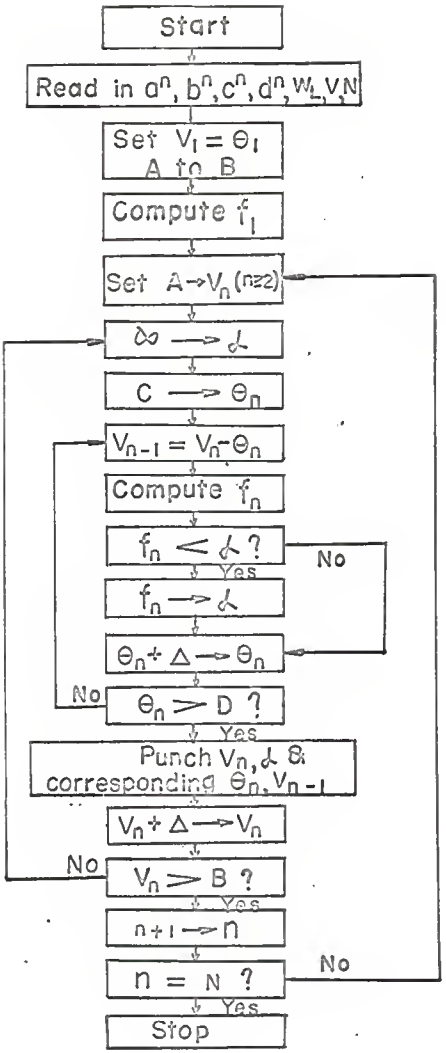

Fig. 8. Outline of computer program for rocket problem.

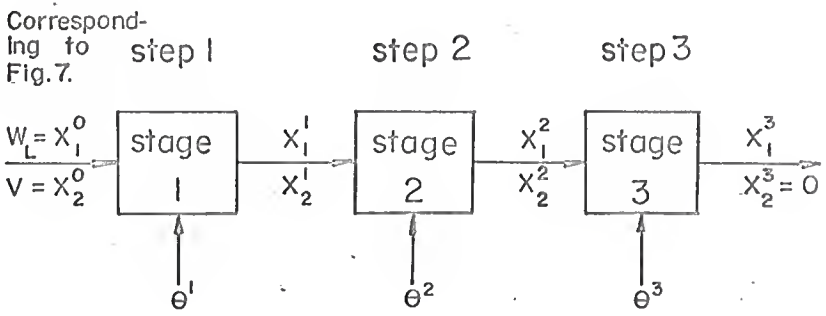

Fig.9. Stages sketch for the maximum principle algorithm.

It can be shown that

$$\sum_{n=1}^{N} (x_2^{n-1} - x_2^n) = \sum_{n=1}^{N} \theta^n$$

$$\sum_{n=1}^{N} \theta^n = V$$

Applying the end point specification case of the maximum principle algorithm equations (27), (28), (38) and (39) become

$$z_1^{n-1} = \frac{b^n z_1^n}{\exp(-\frac{\theta^n}{c^n}) + a^n} \tag{90}$$

$$z_2^{n-1} = z_2^n$$

$$z_1^N = 1 \tag{91}$$

Thus

$$z_2^n = z_2^N = z_2 \tag{92}$$

$$n = 1, 2, \ldots, N$$

$$H^n = z_1^n \left\{ \frac{b^n x_1^{n-1} + d^n}{\exp(-\frac{\theta^n}{c^n}) + a^n} \right\} + z_2(x_2^{n-1} - \theta^n)$$

Because of the introduction of the second state variable, x_2, the constraints, $0 \leq \theta^n \leq V$, are automatically removed as indicated by the relation $\sum_{n=1}^{N} \theta^n = V$ and the fact $\theta^n \geq 0$.

The optimal sequence of $\{\theta^n\}$ therefore must occur at the extreme points. That is:

$$\theta^n: \quad \frac{\partial H^n}{\partial \theta^n} = 0$$

$$\frac{\partial H^n}{\partial \theta^n} = -z_2^n + \frac{z_1^n}{c^n} \frac{(b^n x_1^{n-1} + d^n) \exp(-\frac{\theta^n}{c^n})}{\left[\exp(-\frac{\theta^n}{c^n}) + a^n\right]^2} = 0$$

Solving for z_1^n

$$z_1^n = \frac{z_2 c^n \left[\exp(-\frac{\theta^n}{c^n}) + a^n\right]^2}{(b^n x_1^{n-1} + d^n) \exp(-\frac{\theta^n}{c^n})} \tag{93}$$

likewise

$$z_1^{n-1} = \frac{z_2 c^{n-1} \left[\exp(-\frac{\theta^{n-1}}{c^{n-1}}) + a^{n-1}\right]^2}{(b^{n-1} x_1^{n-2} + d^{n-1}) \exp(-\frac{\theta^{n-1}}{c^{n-1}})} \tag{94}$$

Inserting equations (93) and (94) into equation (90)

$$\frac{c^{n-1} \left[\exp(-\frac{\theta^{n-1}}{c^{n-1}}) + a^{n-1}\right]^2}{(b^{n-1} x_1^{n-2} + d^{n-1}) \exp(-\frac{\theta^{n-1}}{c^{n-1}})} = \frac{b^n c^n \left[\exp(-\frac{\theta^n}{c^n}) + a^n\right]}{(b^n x_1^{n-1} + d^n) \exp(-\frac{\theta^n}{c^n})} \tag{95}$$

From equation (84a)

$$\exp(-\frac{\theta^n}{c^n}) = \frac{b^n x_1^{n-1} - a^n x_1^n + d^n}{x_1^n} \tag{96}$$

$$\exp(-\frac{\theta^n}{c^n}) + a^n = \frac{b^n x_1^{n-1} + d^n}{x_1^n} \tag{97}$$

likewise

$$\exp\left(-\frac{\theta^{n-1}}{c^{n-1}}\right) = \frac{b^{n-1}x_1^{n-2} - a^{n-1}x_1^{n-1} + d^{n-1}}{x_1^{n-1}} \tag{98}$$

$$\exp\left(-\frac{\theta^{n-1}}{c^{n-1}}\right) + a^{n-1} = \frac{b^{n-1}x_1^{n-2} + d^{n-1}}{x_1^{n-1}} \tag{99}$$

Combining equation (95), (96), (97), (98) and (99)

$$x_1^n = \frac{1}{a^n}\left\{ b^n x^{n-1} + d^n - \frac{b^n c^n x_1^{n-1}\left(b^{n-1}x_1^{n-2} - a^{n-1}x_1^{n-1} + d^{n-1}\right)}{c^{n-1}\left(b^{n-1}x_1^{n-2} + d^{n-1}\right)} \right\} \tag{100}$$

For the N-th stage, from equations (90) and (91)

$$z_1^{N-1} = \frac{b^N}{\exp\left(-\frac{\theta^N}{c^N}\right) + a^N} \tag{101}$$

Substituting equation (91) into equation (93)

$$\frac{\exp\left(-\frac{\theta^N}{c^N}\right)}{\left[\exp\left(-\frac{\theta^N}{c^N}\right) + a^N\right]^2} = \frac{z_2 c^N}{\left(b^N x_1^{N-1} + d^N\right)} \tag{102}$$

For the (N-1)-th stage, from equation (94)

$$\frac{\exp\left(-\frac{\theta^{N-1}}{c^{N-1}}\right)}{\left[\exp\left(-\frac{\theta^{N-1}}{c^{N-1}}\right) + a^{N-1}\right]^2} = \frac{z_2 c^{N-1}}{z_1^{N-1}\left(b^{N-1}x^{N-2} + d^{N-1}\right)} \tag{103}$$

Dividing equation (102) by equation (103) and inserting equation (101) for z_1^{N-1}

$$\frac{\exp(-\frac{\theta^N}{c^N})}{\exp(-\frac{\theta^N}{c^N}) + a^N} = \frac{b^N c^N (b^{N-1} x_1^{N-2} + d^{N-1}) \exp(-\frac{\theta^{N-1}}{c^{N-1}})}{\left[\exp(-\frac{\theta^{N-1}}{c^{N-1}}) + a^{N-1}\right]^2 (b^N x_1^{N-1} + d^N) c^{N-1}} \quad (104)$$

The working scheme for digital computation is as follows:

1. Since x_1^o and x_2^o are known, x_1^1 and x_2^1 can be evaluated from equations (84a) and (88), by assuming a value for θ^1.

2. Next, x^n can be calculated directly from equation (100) for stages 2 to (N-1). Then θ^n and x_2^n are calculated from equation (84a) and equation (88) respectively.

3. Finally, θ^N is obtained from equation (104) for the last stage (N-th stage). If x_2^N calculated by substituting the computed θ^N into equation (88) is not zero, the procedure is repeated from the first step by assuming a new value for θ^1.

4. If the x_2^N thus obtained is zero, this sequence of θ^n is the optimal control actions sought. x_1^N can be secured from equation (84a)

A similar working scheme may be followed by assuming a value of x_1^1 instead of θ^1.

Stagewise Biochemical Reactors

In this section a generalized optimization problem of first order consecutive biochemical reaction is considered. It is numerically solved in detail. A comparative study of the two approaches is also made.

In many of the biochemical reactions, the catalyst is an enzyme of protein origin (24). Although the catalytic behavior of enzymes is quite complicated, the two most influential factors on the activity of anzymes are temperature and pH value (24). An increase of temperature or pH value will enhance the activity of the enzyme; but, beyond a certain extent the enzyme would be deactivated by further increase of temperature or pH value. This feature of the biochemical reaction is different from that of nonbiochemical reactions whose specific reaction rates change according to the Arrhenius rate expression. The fact that there exists an optimum condition for the enzyme activity is well-known (24,25). Though the exact functions of the activity versus temperature and pH value should be acquired under the actual operating condition, the general form can be represented by the unimodal graph similar to that shown in Figure 10 which is a sketch of the following hypothetical functions (25):

$$k(\theta) = -3\theta^2 + 10\,\theta + 12 \tag{105}$$

$$h(\theta) = -\theta^2 + 9\theta - 6 \tag{106}$$

where θ is the variable representing either temperature or pH value, and $h(\theta)$ and $k(\theta)$ are the activities of enzymes.

The following consecutive biochemical reaction system will be considered

$$A \xrightarrow[\text{Enzyme 1}]{k(\theta)} B \xrightarrow[\text{Enzyme 2}]{h(\theta)} C$$

This reaction system is fairly common in the fermentation processes. Some of the examples are listed in Table 2 (25). The enzyme indicated is the catalyst for each reaction. Many of them are of commerical importance.

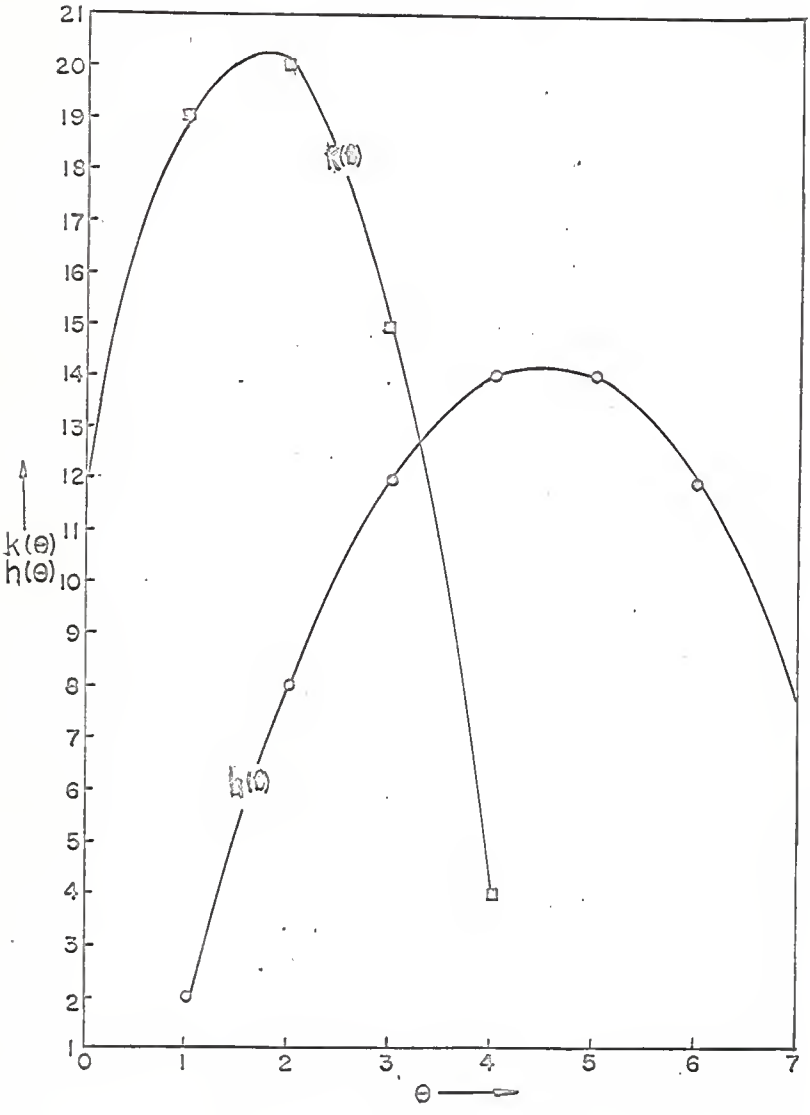

Fig. 10. Graphs of k(θ) and h(θ)

The raw material A is expensive, B is waste and C is the desired product. If a series of ideal equal volume backmix reactors shown in Figure 2 is available, the problem is to find an optimal decision sequence $\left\{ \theta^n \right\}$ in order to maximize the production of C for a fixed input A.

The feed rate is maintained constant and thus the residence time in each stage is equal, let it be t. A solution is presented in a general form first and then a three-stage optimization problem is numerically solved for the case with the residence time in each stage t = 0.01.

Dynamic Programming Solution. Let x, y and w represent the concentrations of A, B and C respectively. The material balance for the n-th stage is

$$x_{n+1} = x_n + k(\theta_n)\, x_n t \tag{107}$$

$$y_{n+1} = y_n + t\left[h(\theta_n)\, y_n - k(\theta_n)\, x_n\right] \tag{108}$$

$$w_{n+1} = w_n - h(\theta_n)\, y_n t \tag{109}$$

Defining

$$\left.\begin{aligned} u_{n+1} &= \frac{y_{n+1}}{x_{n+1}} & u_n &= \frac{y_n}{x_n} \\ v_{n+1} &= \frac{w_{n+1}}{x_{n+1}} & v_n &= \frac{w_n}{x_n} \end{aligned}\right\} \tag{110}$$

TABLE 2

Some examples of consecutive biochemical reactions (25)

1.	Celluloses	—Cellulase→	Cellobiose	—Cellobiase→	Glucose	
2.	Maltose Trehalose	—Maltase Trehalase→	Glucose	—Zymase→	Alcohol + CO_2	
3.	Insulin	—Inulase→	Fructose	—Zymase→	Alcohol + CO_2	
4.	Albumoses Peptones Peptides	—Erepsin→	Amino Acid	—Desaminases→	Hydroxy-acid + Ammonia	
5.	Xanthine	—Xanthoxides→	Uric Acid	—Uricase→	Allantoin	
6.	Sucrose	—Invertase→	Fructose + Glucose	—Zymase→	Alcohol + CO_2	

and dividing equations (108) and (109) by equation (107) and combining the resulting equations with equation (110)

$$u_{n+1} = \frac{u_n \left[1 + h(\Theta_n)t\right] - k(\Theta_n)\, t}{1 + k(\Theta_n)t} \tag{111}$$

$$v_{n+1} = \frac{v_n - t\, h(\Theta_n)\, u_n}{1 + k(\Theta_n)t} \tag{112}$$

From equation (109)

$$\frac{w_n - w_{n+1}}{x_{n+1}} = \frac{y_n}{x_{n+1}}\, h(\Theta_n)t = \frac{h(\Theta_n)t}{x_{n+1}} \left\{ \frac{y_{n+1} + k(\Theta)t \left(\frac{x_{n+1}}{1 + k(\Theta_n)t}\right)}{1 + h(\Theta_n)t} \right\}$$

$$\frac{w_n - w_{n+1}}{x_{n+1}} = \frac{h(\Theta_n)t}{1 + h(\Theta_n)t} \left\{ u_{n+1} + \frac{k(\Theta_n)t}{1 + k(\Theta_n)t} \right\} \tag{113}$$

The objective function of $f_N(x_{N+1}, y_{N+1})$ is

$$f_N(x_{N+1}, y_{N+1}) = \underset{\{\Theta_n\}}{\text{Max}} \sum_{1}^{N} (w_n - w_{n+1}) \tag{114}$$

It can be seen from equation (113) that equation (114) depends only on the ratio of x_{N+1} and y_{N+1} which is u_{N+1}. This reduction in the number of state variables from two to one is possible because of the homogenous nature of the first order consecutive reaction. Therefore:

$$f_N(x_{N+1}, y_{N+1}) = x_{N+1}\, g_N(u_{N+1}) \tag{115}$$

Now, applying the dynamic programming algorithm to the function f_N

$$f_N(x_{N+1}, y_{N+1}) = \underset{\theta_N}{\text{Max}} \left\{ (w_N - w_{N+1}) + f_{N-1}(x_N, y_N) \right\} \tag{116}$$

Combining equations (113), (115) and (116)

$$g_N(u_{N+1}) = \underset{\theta_N}{\text{Max}} \left\{ \frac{h(\theta_N)t}{1 + h(\theta_N)t} \left(u_{N+1} + \frac{k(\theta_N)t}{1 + k(\theta_N)t} \right) + \frac{1}{1 + k(\theta_N)t} g_{N-1}(u_N) \right\} \tag{117}$$

The general recursive equation for the n-th stage is,

$$g_n(u_{n+1}) = \underset{\theta_n}{\text{Max}} \left\{ \frac{h(\theta_n)t}{1 + h(\theta_n)t} \left(u_{n+1} + \frac{k(\theta_n)t}{1 + k(\theta_n)t} \right) + \frac{1}{1 + k(\theta_n)t} g_{n-1}(u_n) \right\} \tag{118}$$

Equation (118) contains only one state variable, u, and it can be solved recursively in conjunction with the following equation:

$$u_n = \frac{u_{n+1}\left[1 + k(\theta_n)t\right] + k(\theta_n)t}{1 + h(\theta_n)t} \tag{119}$$

A flow chart for solving the recurrence equations is shown in Figure 11, and a Fortran 1620 program is given in Appendix B. The results for u_{N+1} ranging from 0.03 to 2 are presented in Figure 12. About 12 hours were required to obtain such results using the increment of θ, 0.1, and that of u, 0.03. The range of θ considered was from 1 to 5. The computer used for this calculation is IBM 1620 with 60,000 memories.

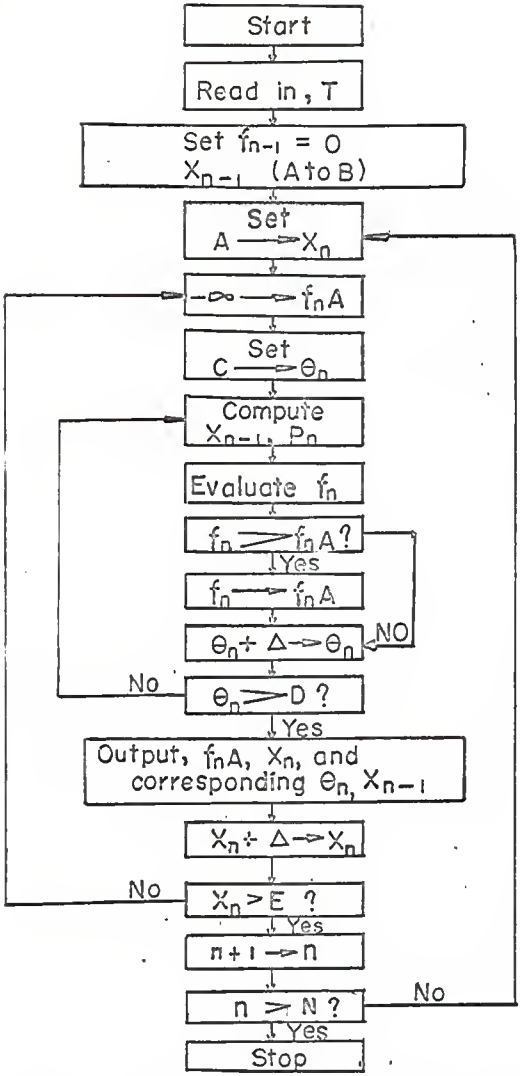

Fig.11. Flow chart for dynamic Programming solution of biochemical reaction problem.

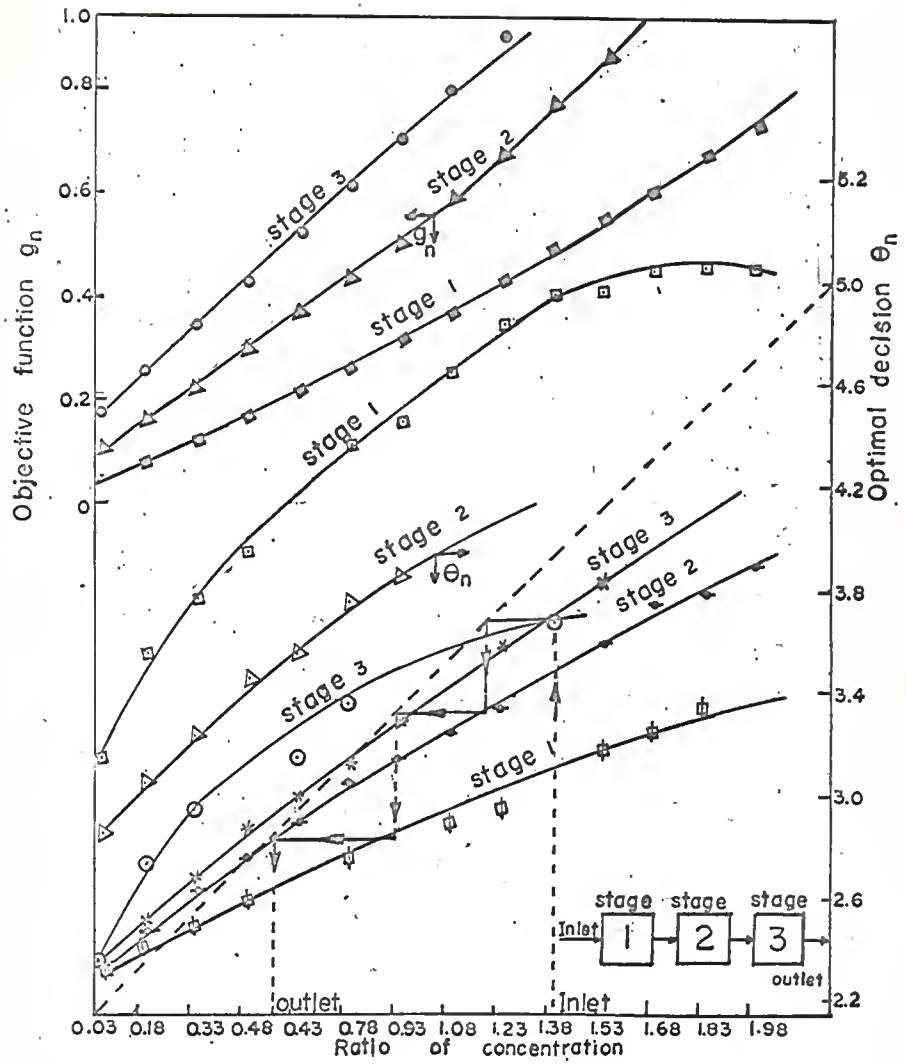

Fig.12. A graphical representation of the results of the dynamic programming solution for biochemical reaction.

Maximum Principle Solution. The notations for the maximum principle algorithm are adopted, x_1, x_2 and x_3 being used to represent the concentration of x, y and w respectively.

From equations (107), (108) and (109)

$$x_1^{n-1} = x_1^n \left[1 + k(\theta^n)t\right]$$

$$x_2^{n-1} = x_2^n \left[1 + h(\theta^n)t\right] - \frac{k(\theta^n)x_1^{n-1}\, t}{1 + k(\theta^n)t}$$

$$x_3^{n-1} = x_3^n - h(\theta^n)\, x_2^n t$$

Solving for x_k^n, k = 1, 2, 3

$$x_1^n = \frac{x_1^{n-1}}{1 + k(\theta^n)t} \tag{120}$$

$$x_2^n = \left\{ x_2^{n-1} + \frac{k(\theta^n)x_1^{n-1}\, t}{1 + k(\theta^n)t} \right\} / \left[1 + h(\theta^n)t\right] \tag{121}$$

$$x_3^n = x_3^{n-1} + h(\theta^n)t \left(x_2^{n-1} + \frac{k(\theta^n)\, x_1^{n-1}\, t}{1 + k(\theta^n)t}\right) / \left[1 + h(\theta^n)t\right] \tag{122}$$

The problem is to maximize x_3^N with the starting conditions,

$$\left.\begin{aligned} x_1^0 &= a \\ x_2^0 &= b \\ x_3^0 &= c \end{aligned}\right\} \tag{123}$$

Applying the maximum principle algorithm and referring to Figure 4, the following set of equations are obtained from equations (27), (28), (36) and (37):

$$z_1^{n-1} = \frac{z_1^n}{1 + k(\theta^n)t} + \frac{t\, k(\theta^n)\, (z_2^n + h(\theta^n)\, z_3^n t)}{(1 + k(\theta^n)t) \quad (1 + h(\theta^n)t)}$$

$$z_2^{n-1} = \frac{z_2^n + h(\theta^n)t\, z_3^n}{1 + h(\theta^n)t}$$

$$z_3^{n-1} = z_3^n$$

The boundary conditions for z_i are

$$z_i^N = \begin{cases} 1, & i=3 \\ 0, & i\neq 3 \end{cases}$$

Thus

$$z_3^n = 1 \tag{124}$$

$$z_2^{n-1} = \frac{z_2^n + h(\theta^n)t}{1 + h(\theta^n)t} \tag{125}$$

$$z_1^{n-1} = \frac{\left[1 + h(\theta^n)t\right] \quad z_1^n + k(\theta^n)t \quad \left[z_2^n + h(\theta^n)t\right]}{\left[1 + h(\theta^n)t\right] \quad \left[1 + k(\theta^n)t\right]} \tag{126}$$

and

$$H^n = \frac{x_1^{n-1}\, z_1^n}{1 + k(\theta^n)t} + \left[z_2^n + h(\theta^n)t\right] \left(x_2^{n-1} + \frac{k(\theta^n)t\, x_1^{n-1}}{1 + k(\theta^n)t}\right) / \left[1 + h(\theta^n)t\right] \tag{127}$$

The following substitutions are made for simplification:

$$F(\theta^n) = \frac{k(\theta^n)t}{1 + k(\theta^n)t} \qquad G(\theta^n) = \frac{h(\theta^n)t}{1 + h(\theta^n)t}$$

$$1 - F(\theta^n) = \frac{1}{1 + k(\theta^n)t} \qquad 1 - G(\theta^n) = \frac{1}{1 + h(\theta^n)t}$$

$$F_1(\theta^n) = \frac{d\,F(\theta^n)}{d\,\theta^n} \qquad G_1(\theta^n) = \frac{d\,G(\theta^n)}{d\,\theta^n}$$

$$H^n = x_1^{n-1} z_1^n \left[1 - F(\theta^n)\right] + \left[x_2^{n-1} + F(\theta^n)\, x_1^{n-1} t\right] \left\{ z_2^n \left[1 - G(\theta^n)\right] + tG(\theta^n) \right\} + x_3^{n-1} \tag{128}$$

The optimal decision θ^n is at

$$\theta^n : \frac{\partial H^n}{\partial \theta^n} = 0$$

$$\frac{\partial H^n}{\partial \theta^n} = - x_1^{n-1} z_1^n F_1(\theta^n) + F_1(\theta^n) x_1^{n-1} t \left\{ z_2^n \left[1 - G(\theta^n)\right] + tG(\theta^n) \right\} +$$

$$\left[x_2^{n-1} + F(\theta^n) x_1^{n-1} t\right] \left[tG_1(\theta^n) - z_2^n G_1(\theta^n) \right] = 0 \tag{129}$$

This problem does not yield the analytic solution as did the first two problems. The general working procedure stated before should be applied.

A flow chart for solving this problem is shown in Figure 13, and a Fortran 1620 computer program for the three-stage system is included in Appendix C. It was found to be more convenient to use the

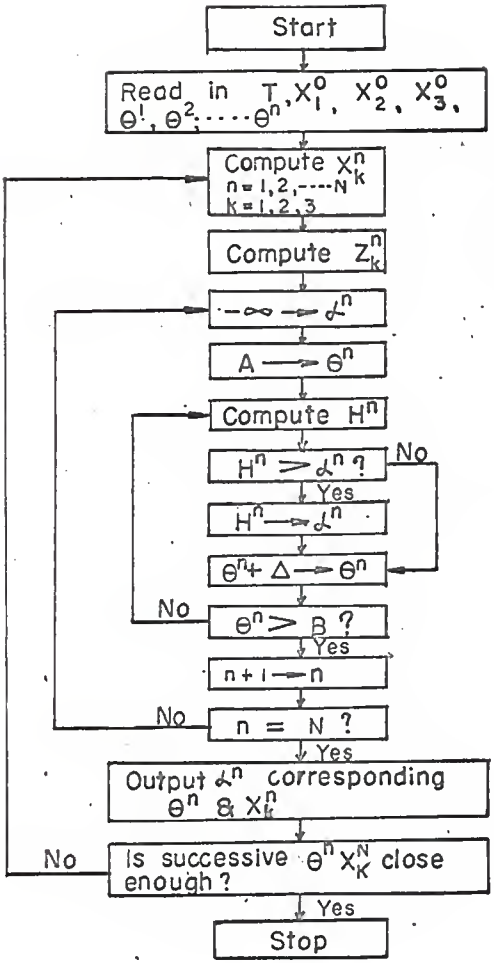

Fig. 13. Flow chart for maximum principle solution of biochemical reaction problem.

expression H^n = Max than $\frac{\partial H^n}{\partial \theta^n} = 0$ to solve the optimum θ^n in computation with the use of a digital computer.

With the given values of the initial concentrations, $x_1^0 = 1$, $x_2^0 = 0.3$, $x_3^0 = 0$, a set of θ^1, θ^2 and θ^3 is arbitrarily chosen. x is then calculated by equations (120), (121) and (122) and z by equations (125) and (126). Substituting these values into equation (127), a new set of θ^1, θ^2 and θ^3 is obtained. This new set of decisions is employed to calculate the x and z and again a set of θ^1, θ^2 and θ^3. The procedure is stopped when two consecutive sets of decisions are sufficiently close. Table 3 shows the detail of calculation.

The machine calculation from the first guess to the final answer required less than half an hour with θ^n ranging from 1.0 to 4.5 with an increment 0.1. Table 3 shows that z and x_3 converge very rapidly, and the results of two different first guesses are identical. The computer used was an IBM Fortran 1620 with 20,000 memories.

The dynamic programming solution for the same given initial concentrations were $\theta^1 = 2.8$, $\theta^2 = 3.2$, $\theta^3 = 4.0$, and $x_3^3 = 0.3231$ which agree very closely with the maximum principle solution. A detailed comparison of the two approaches will be presented later.

Denbigh's Reaction System

Denbigh(26) proposed the following reaction system

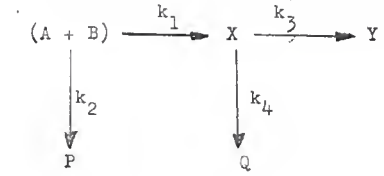

TABLE 3

Calculation according to the maximum principle algorithm

Initial concentrations $x_1^0 = 1$, $x_2^0 = 0.3$, $x_3^0 = 0$

1. First guessing $\theta^1 = 2.5$, $\theta^2 = 3.0$, $\theta^3 = 3.5$

	θ^1	θ^2	θ^3	x_3^3
First Calculation	2.7	3.3	4.0	0.3206
Second Calculation	2.6	3.1	3.9	0.3208
Third Calculation	2.7	3.2	4.0	0.3210
Fourth Calculation	2.6	3.1	4.0	0.3210
Fifth Calculation	2.6	3.2	4.0	0.3211
Sixth Calculation	2.6	3.2	4.0	0.3211

2. First Cuessing $\theta^1 = 4.0$, $\theta^2 = 4.0$, $\theta^3 = 4.0$

First Calculation	2.5	2.8	3.5	0.3126
Second Calculation	2.8	3.3	4.1	0.3196
Third Calculation	2.6	3.1	3.9	0.3208
Fourth Calculation	2.7	3.2	4.0	0.3210
Fifth Calculation	2.6	3.1	4.0	0.3210
Sixth Calculation	2.6	3.2	4.0	0.3211
Seventh Calculation	2.6	3.2	4.0	0.3211

and demonstrated that a great improvement in yield of the desired product Y can result when optimal policies are adopted for a stage-wise reactor system. X is an intermediate and P and Q are the products of side reactions. A and B are the raw material where B is cheap and excess amounts are available for reaction. This system comprises a general class of reactions which are encountered in the manufacturing chemical industry (26). The reactions are taken to be first order regarding the concentration of A and X. The reaction rate constants of the four reactions are of the usual Arrehnius form

$$k_i = a_i \exp\left\{-E_i/RT\right\} , \qquad i = 1, 2, 3, 4$$

Four cases, depending on the relative magnitude of the reaction rates, arise (6).

1. $E_1 > E_2$, $E_3 > E_4$: It can be seen that a conversion of nearly all A to Y is expected with a sufficiently high operating temperature.
2. $E_1 < E_2$, $E_3 < E_4$: For this case, the lowest possible temperature will give the best yield.
3. $E_1 > E_2$, $E_3 < E_4$: In the early stages of the reaction the temperature should be high to promote the production of X. As soon as the quantity of X becomes considerable, the temperature should be lowered to prevent the side reaction. Hence, a decreasing sequence of temperature is desirable.
4. $E_1 < E_2$, $E_3 > E_4$: In this case the same argument calls for an increasing temperature sequence.

The first two cases are not of much interest. The third and fourth cases are considered.

Let w, x and y denote the concentrations of A, X and Y respectively and ρ the residence time in each stage. The material balance around the n-th stage, referring to Figure 2, yields

$$w_{n+1} = w_n \left\{1 + \rho_n (k_{1n} + k_{2n})\right\}$$

$$x_{n+1} = - w_n \rho_n k_{1n} + x_n \left\{1 + \rho_n (k_{3n} + k_{4n})\right\}$$

$$y_{n+1} = - x_n \rho_n K_{3n} + y_n$$

Using the values of the specific reaction rates given by Denbigh (26)

$$\frac{k_2}{k_1} = 10^4 \exp\left(-\frac{3000}{T}\right)$$

$$\frac{k_3}{k_1} = 10^{-2}$$

$$\frac{k_4}{k_3} = 10^{-4} \exp\left(\frac{3000}{T}\right)$$

and making the following substitution

$$\theta = \frac{k_2}{k_1} = 10^4 \exp\left(-\frac{3000}{T}\right) \tag{130}$$

$$t = \rho k_1$$

the material balance around the n-th stage of a system consisting of N completely stirred tank reactors becomes

$$w_{n+1} = \left\{1 + t_n(1 + \theta_n)\right\} w_n \tag{131}$$

$$x_{n+1} = -t_n w_n + \left\{1 + 0.01\, t_n\, \theta_n^{-1} (1 + \theta_n)\right\} x_n \tag{132}$$

$$y_{n+1} = - 0.01\, t_n\, x_n + y_n \tag{133}$$

Since these equations are homogeneous in the concentrations, the ratio of the state variables may be employed (6) to reduce the dimensionality of the problem. Aris (6) solved this problem by means of the dynamic programming algorithm, proceeding in the same way as done in solving the biochemical reaction problem. When he encountered the difficulty of the individual state variables being too large, the reciprocals of the state variables were employed to improve the accuracy of interpolation (6). His results are listed in Table 4.

The remainder of this section describes the solution of the problem by use of the maximum principle algorithm. The results from both methods are also compared.

Adopting the notations of the maximum principle algorithm and referring to Figure 4, equations (131), (132) and (133) become

$$x_1^{n-1} = x^n\left\{1 + t^n(1 + \theta^n)\right\}$$

$$x_2^{n-1} = -t^n\, x_1^n + \left\{1 + 0.01\, t^n(1 + \theta^n)/\theta^n\right\} x_2^n$$

$$x_3^{n-1} = -\,0.01\, t^n\, x_2^n + x_3^n$$

Solving for x_k^n, k = 1, 2, 3,

$$x_1^n = \frac{x_1^{n-1}}{1 + t^n(1 + \theta^n)} \qquad (134)$$

$$x_2^n = \frac{x_2^{n-1} + t^n\left\{\dfrac{x_1^{n-1}}{1 + t^n(1 + \theta^n)}\right\}}{1 + 0.01\ t^n(1 + \theta^n)/\theta^n} \tag{135}$$

$$x_3^n = x_3^{n-1} + 0.01\ t^n \left\{ \frac{x_2^{n-1} + t^n \left[\dfrac{x_1^{n-1}}{1 + t^n(1 + \theta^n)} \right]}{1 + 0.01\ t^n(1 + \theta^n)\ /\theta^n} \right\} \tag{136}$$

The problem is to maximize x_3^N with the starting conditions

$$\left.\begin{aligned} x_1^0 &= 1 \\ x_2^0 &= 0 \\ x_3^0 &= 0 \end{aligned}\right\} \tag{137}$$

Applying the maximum principle algorithm the following set of equations corresponding to equations (27), (28), (36) and (37) is obtained

$$z_1^{n-1} = \frac{z_1^n}{1 + t^n(1 + \theta^n)} + \frac{t^n\ z_2^n}{\left[1 + t^n(1 + \theta^n)\right]\ \left[1 + 0.01\ t^n(1 + \frac{1}{\theta^n})\right]}$$

$$+ \frac{0.01\ z_2^n(t^n)^2}{\left[1 + t^n(1 + \theta^n)\right]\ \left[1 + 0.01\ t^n(1 + \frac{1}{\theta^n})\right]}$$

$$z_2^{n-1} = \frac{z_2^n}{1 + 0.01\ t^n(1 + \frac{1}{\theta^n})} + \frac{0.01\ t^n z_3^n}{1 + 0.01\ t^n(1 + \frac{1}{\theta^n})}$$

$$z_3^{n-1} = z_3^n$$

The boundary conditions for z_i are

$$z_i^N = \begin{cases} 1, & i = 3 \\ 0, & i = 1,2 \end{cases}$$

Thus

$$z_3^n = 1 \qquad n = 1, 2, \ldots, N \tag{138}$$

$$z_2^{n-1} = \frac{z_2^n}{1 + 0.01\ t^n(1 + \frac{1}{\theta^n})} + \frac{0.01\ t^n}{1 + 0.01\ t^n(1 + \frac{1}{\theta^n})} \tag{139}$$

$$z_1^{n-1} = \frac{z_1^n}{1 + t^n(1 + \theta^n)} + \frac{t^n\ z_2^n}{\left[1 + t^n(1 + \theta^n)\right]\ \left[1 + 0.01\ t^n(1 + \frac{1}{\theta^n})\right]}$$

$$+ \frac{0.01\ (t^n)^2}{\left[1 + t^n(1 + \theta^n)\right]\ \left[1 + 0.01\ t^n(1 + \frac{1}{\theta^n})\right]} \tag{140}$$

and

$$H^n = \frac{x_1^{n-1}\ z_1^n}{1 + t^n(1 + \theta^n)} + \left\{ \frac{x_2^{n-1} + \frac{x_1^{n-1}\ t^n}{1 + t^n(1 + \theta^n)}}{1 + 0.01\ t^n(1 + \frac{1}{\theta^n})} \right\} \left\{ z^n + 0.01\ t^n \right\} + x_3^{n-1}$$

$$\theta^n : \ H^n = \text{Max} \tag{141}$$

The general procedure for solving this problem is exactly the same as the one used in solving the biochemical problem in the last sections. In the results presented in Table 5, the notations are changed slightly to match those in Table 4 for the convenience of comparison. The Fortran 1620 programs for two and three-stage problems are included

TABLE 4

Aris's results for Denbigh's reactions (6)

Number of Reactors (N)	Temperature T_3	T_2	T_1	"Holding Times" t_3	t_2	t_1	Yield (%)
With no restrictions							
1	—	—	326	—	—	∞	25
2	—	280	∞	—	4.0	∞	57.4
3	270	320	∞	1.6	0.8	∞	66.3
With restrictions on temperature and holding time							
1	—	—	318	—	—	340	2221
2	—	276	394	—	2.82	2100	45.1
3	260	288	394	1.4	1.3	2100	49.5

in Appendices D and E. The programs can be easily converted into a general program for an N stage system. The computing time, dependent linearly on the number of stages, is almost independent of the number of state variables. While the time used for the two-stage system calculation was about two hours, the amount of time would largely depend on the searching range and searching grid of the two variables. The time required is approximately proportional to the products of the number of increments of both control variables, t^n and θ^n, used in search. Actually, a proper choice of the searching grid could make the calculation of a three-stage reaction system possible in half an hour without the sacrifice of too much accuracy.

A comparison of the results presented in Tables 4 and 5 shows that they are very close in the case without constraints on the control variables. But for the case with restrictions, the results do show sizable differences for the case of the three-stage system. The yields depend largely on the control variables of the last stage, and the difference between the control variables of the last stage and the rest of the stages is much greater in the first case than is that of the second case. This difference means that the control variables of the first two stages have more influence on the yields in the restricted case than in the case without restriction. Because of the inevitable interpolation in the dynamic programming solution, a greater error would result for the case with restriction than for the case without restriction.

Another fact which may uphold the statements above, is that, if the values of the control variables given by Aris for the three-stage process with restriction are used to calculate the yield directly from equations (131), (132) and (133), the answer is 53.83% instead of 49.5% as given by Aris. It appears that, for this problem, the maximum principle approach not only makes the calculation simpler but also gives more accurate results than dynamic programming.

Problem of Growth and Predation

Some interesting models of bacterial growth and competition have been proposed recently. In this section two problems of this kind, formulated by Aris (27) using the dynamic programming, are reformulated by using the maximum principle algorithm. These two problems, especially the second one, exhibit the superiority of the maximum principle approach over that of dynamic programming.

The dynamic programming solution is not repeated. A short statement of the problem, however, is provided, adopting notations in Figure 4.

First Problem. A population of creatures x_1 preys on a population x_2 and both classes are undesirable (27). A pesticide is effective against x_1, but not against x_2. If x_1^{n-1} is the population of x_1 at the beginning of the treatment in the n-th time interval and x_2^{n-1} the corresponding population of x_2 in the same time interval, the growth and predation can be represented by

$$x_1^n - x_1^{n-1} = h(\varphi^n)\, x_1^{n-1}$$

Table 5

Denbigh's system calculated by the maximum principle algorithm

Number of Reactors (N)	Temperature T_3	T_2	T_1	"Holding Times" t_3	t_2	t_1	Yield (%)
With no restrictions							
1	—	—	326	—	—	∞	25
2	—	279.9	∞	—	4.55	∞	57.99
3	267.4	295	∞	1.7	2.6	∞	66.55
With restrictions on temperature and holding time							
1	—	—	318	—	—	340	22
2	—	278.8	394	—	3.85	2100	49.03
3	265	292	394	1.35	2.35	2100	54.85

$$x_2^n - x_2^{n-1} = -k(\theta^n)\, x_1^{n-1} + m(\theta^n)\, x_2^{n-1}$$

$$x_1^n = \left[1 + h(\theta^n)\right] x_1^{n-1} \tag{142}$$

$$x_2^n = \left[1 + m(\theta^n)\right] x_2^{n-1} - k(\theta^n)\, x_1^{n-1} \tag{143}$$

where h, k and m are functions of the dosage of the pesticide, θ, which is added at a definite time interval-stage.

The problem is to minimize the linear combination function of x_1 and x_2 at the end of the pesticide process, with the initial conditions $x_1^0 = a$, $x_2^0 = b$. That is to minimize $(cx_1^N + dx_1^N)$, c and d being constants.

Defining a new state variable

$$x_3^n = cx_1^n + dx_2^n$$

$$x_3^n = x_1^{n-1}\left\{c\left[h\,(\theta^n) + 1\right] - dk(\theta^n)\right\} + d\left[1 + m(\theta^n)\right] x_2^{n-1} \tag{144}$$

The problem is then transformed to minimize x_3^n, with the starting conditions

$$\left.\begin{aligned} x_1^0 &= a \\ x_2^0 &= b \\ x_3^0 &= ac + bd \end{aligned}\right\} \tag{145}$$

Applying the maximum principle algorithm (5), the following set of equations corresponding to equations (27), (28), (36) and (37), is obtained:

$$z_1^{n-1} = \left[h(\theta^n) + 1\right] z_1^n - z_2^n k(\theta^n) + z_3^n \left\{c\left[h(\theta^n) + 1\right] - dk(\theta^n)\right\}$$

$$z_2^{n-1} = \left[1 + m(\theta^n)\right] z_2^n + z_3^n d \left[1 + m(\theta^n)\right]$$

$$z_3^{n-1} = 0$$

The boundary conditions for z_i are

$$z_i^N = \begin{cases} 1, i = 3 \\ 0, i = 1,2 \end{cases} \tag{146}$$

Thus

$$z^{n-1} = \left[h(\theta^n) + 1\right] z_1^n - z_2^n k(\theta^n) \tag{147}$$

$$z_2^{n-1} = \left[1 + m(\theta^n)\right] z_2^n \tag{148}$$

and

$$H^n = z_1^n \left[1 + h(\theta^n)\right] x_1^{n-1} + z_2^n \left[1 + m(\theta^n)\right] x_2^{n-1} - k(\theta^n)\, x_1^{n-1} \tag{149}$$

$$H^N = x_1^{N-1} \left\{c\left[h(\theta^N) + 1\right] - dk(\theta^N)\right\} + d\left[1 + m(\theta^N)\right] x_2^{N-1} \tag{150}$$

$$\theta^n \; : \; H^n = \text{Min} \tag{151}$$

$$n = 1,2,\ldots,N$$

The general procedure for solving this problem is to follow the procedure stated in the maximum principle section by using equations (142) through (151).

An alternate algorithm (28,29) to solve this problem without introducing the third state variable is to use the general working scheme in conjunction with equations (142), (143), (147) and (148) and the following boundary conditions instead of equations (145) and (146):

$$\left.\begin{aligned} x_1^0 &= a \\ x_2^0 &= b \\ z_1^N &= c \\ z_2^N &= d \end{aligned}\right\} \qquad (152)$$

Second Problem. A nutrient, x_5, is required by two forms of an organism, x_1 and x_2 if they are to remain viable. x_2 is not capable of reproduction but x_1 may be consumed to form a product x_3 and this product enhances the reproductive process. If the control action is taken at an equal time interval, say t, the following relationship among x can be deduced by a simple biological model:

1. the spontaneous change from x_2 to x_1

$$x_2^{n-1} - x_2^n = -ax_2^n t \qquad (153)$$

2. the formation of x_1 from x_2, the reproduction of a fraction, $(1 - \theta^n)$, of itself and the use of the remaining fraction θ^n to form x_3

$$x_1^{n-1} - x_1^n = ax_2^n t + (1 - \theta^n)x_4^n x_1^n t - \theta_b^n x_1^n t \qquad (154)$$

3. the formation of x_3 and its use to affect the reproduction constant x_4^n

$$x_3^{n-1} - x_3^n = b\theta^n x_1^n t - cx_3^n t \qquad (155)$$

4. the effect on the reproduction constant x_4^n

$$x_4^{n-1} - x_4^n = cx_3^n t \qquad (156)$$

5. the use of the nutrient x_5 to maintain the life of x_2 and x_1,

$$x_5^{n-1} - x_5^n = -(dx_2^n + kx_1^n)t \tag{157}$$

where a, b, c, d and k are constants.

From these relationships, the transformation functions can be derived from equations (153), (155), (156), and (157) as follows:

$$T_2(x_k^{n-1}, \theta^n) = x_2^n = \frac{x_2^{n-1}}{1 - at} \tag{158}$$

$$T_3(x_k^{n-1}, \theta^n) = x_3^n = \frac{x_3^{n-1} - b\theta^n x_1^n t}{1 - ct} \tag{159}$$

$$T_4(x_k^{n-1}, \theta^n) = x_4^n = x_4^{n-1} - \left(\frac{ct}{1 - ct}\right)(x_3^{n-1} - b\theta^n x_1^n t) \tag{160}$$

$$T_5(x_k^{n-1}, \theta^n) = x_5^n = x_5^{n-1} + \frac{tdx_2^{n-1}}{1 - at} + tkx_1^n \tag{161}$$

Combining equations (154), (158) and (160) .

$$\frac{ct}{1 - ct}(1 - \theta^n)t\theta^n bt(x_1^n)^2 + x_1^n\left\{1 - \theta^n bt + (1 - \theta^n)t\left(x_4^{n-1} - \frac{ctx_3^{n-1}}{1 - ct}\right)\right\} - x_1^{n-1} + \frac{atx_2^{n-1}}{1 - at} = 0 \tag{162}$$

If the constants are available, equation (162), which is a quadratic equation, can be solved. Let its solution be

$$x_1^n = T_1(x_k^{n-1}; \theta^n)$$

and also let

$$D \equiv \frac{1}{\left(\frac{ct}{1 - ct}\right)2T_1 bt\theta^n t(1 - \theta^n) + 1 - \theta^n bt + (1 - \theta^n)t\left(x_4^{n-1} - \frac{ctx_3^{n-1}}{1 - ct}\right)} \tag{163a}$$

Then differentiating equation (162) with respect to x_1^{n-1}, x_2^{n-1}, x_3^{n-1}, x_4^{n-1}, and x_5^{n-1}, and solving for $\frac{\partial T_1}{\partial x_1^{n-1}}$, $\frac{\partial T_1}{\partial x_2^{n-1}}$, $\frac{\partial T_1}{\partial x_3^{n-1}}$ $\frac{\partial T_1}{\partial x_4^{n-1}}$, and $\frac{\partial T_1}{\partial x_5^{n-1}}$

$$\frac{\partial T_1}{\partial x_1^{n-1}} = D \tag{163}$$

$$\frac{\partial T_1}{\partial x_2^{n-1}} = -\frac{atD}{1 - at} \tag{164}$$

$$\frac{\partial T_1}{\partial x_3^{n-1}} = \frac{ct(1 - \theta^n)tT_1D}{1 - ct} \tag{165}$$

$$\frac{\partial T_1}{\partial x_4^{n-1}} = -T_1(1 - \theta^n)tD \tag{166}$$

$$\frac{\partial T_1}{\partial x_5^{n-1}} = 0 \tag{167}$$

The objective is to choose optimal θ^n to maximize the final concentration of x_1^N with the specified initial concentrations of the x and with the exhaustion of the nutrient as the end point of the process. If θ^n is small, the reproduction constant x_4 remains small; but, if it is large, too much effort is put into enhancing the reproductive process and too little x_1 is left to reproduce. There is the optimal decision for θ^n to maximize the production of x_1.

The problem is then to maximize x_1^N with the boundary conditions

$$\left.\begin{aligned} x_1^0 &= a^1 \\ x_2^0 &= b^1 \\ x_3^0 &= c^1 \\ x_4^0 &= d^1 \\ x_5^0 &= k^1 \\ x_5^N &= 0 \end{aligned}\right\} \tag{168}$$

and the constraint:

$$\theta^n : 0 \leq \theta^n \leq 1 \tag{169}$$

This is a fixed end point problem. Employing equation (27) and combining it with equations (163) through (167)

$$z_5^{n-1} = z_5^n = z_5 \qquad n = 1,2,\ldots,N \tag{170}$$

$$z_1^{n-1} = D\left\{ z_1^n - \frac{b\theta^n t z_2^n}{1 - ct} + \frac{ctb\theta^n t z_4^n}{1 - ct} + ktz_5 \right\} \tag{171}$$

$$z_2^{n-1} = -\frac{atDz_1^n}{1 - at} + \frac{z_2^n}{1 - at} + \frac{atDbt\theta^n z_3^n}{(1 - ct)(1 - at)}$$

$$- \frac{atDct\theta^n btz_4^n}{(1 - at)(1 - ct)} + z_5\left[\frac{dt}{1 - at} - \frac{ktatD}{1 - at} \right] \tag{172}$$

$$z_3^{n-1} = \frac{ct(1 - \theta^n)tT_1Dz_1^n}{1 - ct} + \left[1 - \frac{\theta^n btct(1 - \theta^n)tT_1D}{1 - ct} \right]$$

$$\left(\frac{z_3^n}{1 - ct} - \frac{ctz_4^n}{1 - ct} + \frac{ktz_5}{1 - ct} \right) \tag{173}$$

$$z_4^{n-1} = -T_1(1 - \theta^n)tD \left[z_1^n - \frac{z_3^n \theta^n bt}{1 - ct} + z_5 kt + \frac{ct\theta^n btz_4^n}{1 - ct} \right] + z_4^n \tag{174}$$

with the boundary condition for z_i

$$z_i = \begin{cases} 1, i = 1 \\ 0, i \neq 1,5 \end{cases} \tag{175}$$

Thus the Hamiltonian expression is,

$$H^n = z_1^n T_1 + z_2^n T_2 + z_3^N T_3 + z_4^n T_4 + z_5^n T_5 \tag{176}$$

and

$$\theta^n : H^n = \text{Max} \tag{177}$$

z_5 is treated as a parameter, like the λ in the sludge problem. Assuming a value of z_5, the problem is solved by the general working procedure without recycle, using equations (158) through (161) and (168) through (177) and obtaining the optimal sequence of decisions $\{\theta^n\}$ for this particular z_5. The value of z_5 is varied until the calculated x_5^N is zero. The sequence of the optimal $\{\theta^n\}$ thus obtained is the answer for the problem.

The equation derived above appear to be lengthly and tedious, but the algorithm is straightforward and overcomes the difficulty associated with high dimensionality, encountered in the dynamic programming solution, and resulting from the large number of state variables involved in the problem.

Problems of Resource Allocation

In this short section, one and two-dimensional allocation processes formulated by Bellman according to the method of dynamic programming (4) are reformulated with the use of the maximum principle algorithm. A general approach to the problems of multi-dimensional allocation is also suggested. The problem of allocation of nutrients is included as a specific example.

First, the case of allocating one type of resource y is considered. The resource y is to be divided into N independent parts, each of which is called an activity. There is a utility function associated with each activity. This function measures the dependence of the return from this activity upon the quantity of the resource allocated. Let h be the utility function which is the net profit received from one particular activity. If the allocation of resource to the n-th activity is θ^n, the objective function to be maximized is the sum of the interval profits.

$$\theta^* = \underset{\{\theta^n\}}{\text{Max}} \left\{ \sum_{n=1}^{N} h^n(\theta^n) \right\}$$

with the constraints

$$\sum_{n=1}^{N} \theta^n = y$$

$$\theta^n \geq 0 \quad ; \quad n = 1, 2, \ldots, N.$$

The maximum principle formulation proceeds as follows:

Defining

$$x_1^n = h^n(\theta^n) + x_1^{n-1}$$

$$x_2^n = x_2^{n-1} + \theta^n$$

with the boundary conditions

$$\left.\begin{aligned} x_1^0 &= 0 \\ x_2^0 &= 0 \\ x_2^N &= y \end{aligned}\right\}$$

it is deduced that

$$x_1^N = \sum_{n=1}^{N} h^n(\theta^n)$$

$$x_2^N = \sum_{n=1}^{N} \theta^n = y$$

The problem is then transformed into that of maximizing x_1^N by employing the maximum principle algorithm for the case with the specified end point. Depending on the nature of the utility function, $h(\theta)$, the solution may be analytical or semi-analytical or numerical.

If there are two types of available resources, y and w, to be allocated, the objective function is

$$O^* = \max_{\{\theta_1^n, \theta_2^n\}} \left\{h^n(\theta_1^n, \theta_2^n)\right\}$$

under the constraints

$$\sum_{n=1}^{N} \theta_1^n = y$$

$$\sum_{n=1}^{N} \theta_2^n = w$$

$$\theta_1^n \geq 0 \qquad n = 1,2,\ldots,N$$

$$\theta_2^n \geq 0 \qquad n = 1,2,\ldots,N$$

Solution by the maximum principle proceeds in the manner similar to the case of the one-dimensional allocations. Defining

$$x_1^n = h(\theta_1^n, \theta_2^n) + x_1^{n-1}$$

$$x_2^n = x_2^{n-1} + \theta_1^n$$

$$x_3^n = x_3^{n-1} + \theta_2^n$$

with the boundary conditions

$$\left.\begin{aligned} x_1^0 &= 0 \\ x_2^0 &= 0 \\ x_3^0 &= 0 \\ x_2^N &= y \\ x_3^N &= w \end{aligned}\right\}$$

Now the problem is to maximize x_1^N employing the maximum principle algorithm for the case with the specified end point. It can be seen that an n-dimensional allocation problem with n+1 state variables and n control variables can be solved similarily.

If the function, $h(\theta)$ is quite complicated, the method of dynamic programming may be preferred for some cases of one and two-dimensional problems. However, for most of the cases, especially when analytical solutions are

permissible, the answers are much easier to obtain by using the maximum principle than by the dynamic programming algorithm. For multi-dimensional resource allocation problems, the dynamic programming approach will be at a impasse because of the exponential increase in computational labor; the maximum principle method may be the only choice.

Suppose that there are of N species of food; each unit of the individual food species contains α units of calories and β units of a particular nutrient content which is to be maximized in the diet. Let superscript n denote the n-th species of food and θ^n be the units of each food species to be allocated to make up a diet. If the diet requires the maximum content of the particular nutrient, as mentioned before, but at the same time, requires that the total units of calories be maintained at a definite value, y, the objective junction is,

$$O^* = \underset{\{\theta^n\}}{\text{Max}} \left\{ \sum_{n=1}^{N} \beta^n \theta^n \right\}$$

with the constraints

$$\sum_{n=1}^{N} \alpha^n \theta^n = y$$

$$\theta^n \geq 1 \quad , \quad n = 1, 2, \ldots, N.$$

The formulation according to the maximum principle proceeds as follows

Defining

$$x_1^n = \beta^n \theta^n + x_1^{n-1} \tag{178}$$

$$x_2^n = x_2^{n-1} + \alpha^n \theta^n \tag{179}$$

with the boundary conditions

$$\left.\begin{aligned} x_1^0 &= 0 \\ x_1^N &= 0 \\ x_2^N &= y \end{aligned}\right\} \tag{180}$$

The problem is transformed into that of maximizing x_1^N. From equation (28), the Hamiltonian expression is

$$H^n = z_1^n(x_1^{n-1} + \beta^n \theta^n) + z_2^n(x_2^{n-1} + \alpha^n \theta^n)$$

From equation (30)

$$z_1^n = z_1^{n-1} \tag{181}$$

$$z_2^n = z_2^{n-1} \tag{182}$$

Applying the boundary conditions of z_i

$$z_1^N = 1$$

thus

$$z_1^n = 1 \quad , \qquad n = 1, 2, \ldots, N \tag{183}$$

$$H^n = x_1^{n-1} + z_2 x_2^{n-1} + (z_2 \alpha^n + \beta^n)\theta^n \tag{184}$$

$$\theta^n \; ; \; H^n = \text{Max} \tag{185}$$

where

$$\theta^n \geq 1$$

Taking z_2 as the parameter, the solution can be carried out by the general working scheme by using equations (178), (179), (180), (184) and (185). z_2 is varied until the boundary conditions are met.

GENERAL DISCUSSION

The maximum principle and the dynamic programming are generally regarded as two alternative ways of solving optimization problems of nonlinear nature (30).

The principal prerequisite for optimizing a process by means of the dynamic programming and the maximum principle algorithms is a mathematical description of the process itself, which often lags far behind the theory of optimization. The principle of optimality, however, can be applied to any optimization problems of stagewise nature irrespective of the availability of the exact mathematical description of the process. The directed net work problem is one of the examples for which the transform functions are not in exact functional form.

The sludge treatment problem is an example for which the analytical solution can be obtained either by the maximum principle algorithm or by the dynamic programming algorithm. However, for many of the problems, these methods are not always practicable. A considerable number of problems fall into a class for which semi-analytical solutions can be derived by use of the maximum principle algorithm, as in the case of the step rocket problem, but it is not so when the dynamic programming algorithm is employed. This resulting in semi-analytical solution is one of the advantages in using the maximum principle algorithm. The semi-analytical solution reduces computational labor considerably. For some problems even a simple hand calculation enables one to obtain a fairly accurate answer.

The majority of problems belong to a class of problems for which no analytical solutions are obtainable, and for which the general computing

scheme must be employed. The last three problems in the application section belong to such a class of problems.

Despite outward difference the basic notions behind dynamic programming and the maximum principle are similar. Under most circumstances their numerical answers are very close and can be used jointly. Their essential difference is a tactical one (30). While dynamic programming starts the investigation by searching the entire grid of the s variables at one stage, stores this grid of values, and proceeds stage by stage, the maximum principle starts the investigation by computing one optimum path along the n stages and then proceeding to improve this optimum path based on the values obtained from the preceding computation. This scheme enables the maximum principle to deal with processes in which the optimum conditions at any stage can be disturbed by conditions at a following stage.

The dynamic programming algorithm leads immediately to a neat computer program for machine calculation. However, since it must store the results of the entire grid of s variables at each stage, the computing time may become considerable and require a large capacity of computer memories. Suppose that there are t decisions to be made at each stage for a process with s state variables at each stage and n total stages, and also suppose that a grid of ten for each variable--state and control--would yield the results with sufficient accuracy. In the dynamic programming formulation, 10^t calculations would be required at each stage for each state variable, and therefore, a total of 10^{s+t} calculations would be required at each stage, resulting in the n 10^{s+t} calculations altogether. The maximum principle formulation would need n 10^t calculations, almost independent of the increase in the number of state variables s. If the

s becomes large, dynamic programming may require much more calculation than the maximum principle, but the most serious difficulty associated with the use of dynamic programming lies in the exponential increase of storage space with respect to s. This difficulty is one of the reasons that, in dynamic programming formulation, an effort is always made to reduce the number of state variables, either by using the ratio of the state variables, either by using the ratio of the state variables as a new variable or by using other techniques. Since in employing dynamic programming the computer program must be written by double indexing the variables, memory space is easily exhausted by an increase in the number of state variables. In programming the maximum principle algorithm for computation only one index is used so that total memory space increases linearly with respect to s.

The error introduced due to the interpolation technique employed in the dynamic programming algorithm is difficult to estimate. If the grid is sufficiently small and the function has no flat portion, the interpolation may not create too much of a problem. It is to be noted, however, that an increase in grid number will result in tremendous increase in computing time and computer memories. Usually a balance should be sought in light of the nature of the transformation functions.

From the previous statements the following conclusion may be drawn: If an appreciable error results from the interpolation and the number of state variables is large, the maximum principle approach is preferable. If the results for a large number of initial values of the state variables are desired, it would be too tedious to run the maximum principle calculations repeatedly. In such instances dynamic programming may be the better choice. If the results for only a few initial values of the state variables

are wanted, the maximum principle approach would give more accurate answers in much shorter computing time than dynamic programming. Generally speaking, if the transformation functions are complicated, it may be advisable to work with dynamic programming, since the maximum principle algorithm requires handling of equations of very complex form which may easily lead to manipulation error.

ACKNOWLEDGEMENT

The author wishes to express his sincere appreciation to Dr. Liang-tseng Fan, under whose direction this study was carried out, for his assistance and advice; Dr. William H. Honstead, Head of the Department of Chemical Engineering, for his help and guidance; Dr. Herbert T. Bates and Dr. John M. Marr, for reading the manuscript; the K. S. U. Engineering Experiment Station for part of the financial support (Project 345 directed by Dr. Liang-tseng Fan) and the K. S. U. Computing Center for use of IBM 1620.

LITERATURE CITED

1. Boas, A. H., Chemical Engineering, 69, No. 25, 147 (1962).
2. Sherwood, G. E. F., and A. E. Taylor, "Calculus", Prentice-Hall, Inc., New York, 1954.
3. Elsgolc, L. E., "Calculus of Variations," Pergamon Press, London, 1962.
4. Bellman, R. E., and S. E. Dreyfus, "Applied Dynamic Programming", Princeton University Press, Princeton, New Jersey, 1962.
5. Katz, L., I.&E.C. Fundamentals, 1, No. 4, 226 (1962).
6. Aris, R., "The Optimal Design of Chemical Reactors — A Study in Dynamic Programming", Academic Press, New York, 1961.
7. Aris, R., D. F. Rudd, and N. R. Amundson, Chem. Eng. Sci., 12, 88, (1960).
8. Rudd, D. F., I.&E.C. Fundamentals, 1, No. 2, 138 (1962).
9. Rudd, D. F., Chem. Eng. Sci., 17, 609 (1962).
10. Roberts, S. M., "Dynamic Programming Formulation of the Catalyst Replacement Problem", A paper presented at the Symposium on Optimization in Chemical Engineering, New York University, New York, (May, 1960).
11. Roberts, S. M., and J. D. Mahoney, Chem. Eng. Prog. Symp. Series, 58, 37 (1962).
12. Dranoff, J. S., L. G. Mitten, W. F. Stevens, and L. A. Wanninger, Jr. Operations Research, 9, 388 (1961).
13. Mitten, L. G., and G. L. Nemhauser, "Multistage Optimization with Dynamic Programming", A paper presented at the Los Angeles Meeting of A.I.Ch.E. (February, 1962).

14. Ahn, Y. K., H. C. Chen, L. T. Fan, and C. G. Wan, "A Modified Moving Bed Grain Dryer", A paper submitted to I.E.C. for publication, (1963).

15. Wang, C. S., M.S. Thesis, Kansas State University (1963).

16. Gibson, J. E., Editor, "Proceedings of Dynamic Programming Workshop", Purdue University (1961).

17. Boas, A. H., Chemical Engineering, 70, No. 1, 95 (1963).

18. Fair, G. M., and J. C. Geyer, "Water Supply and Waste-Water Disposal", John Wiley & Sons, New York (1954) pp 783.

19. Boole, G., "Calculus of Finite Differences", Chelsea Publishing Company, New York, 4thEd. pp 230.

20. Dyke, R. P. T., Jet Propulsion, 28, 336 (1958).

21. Malina, F. J., and M. Summerfield, Journal of the Aeronautical Sciences, 14, 471 (August, 1947).

22. Goldsmith, M., Jet Propulsion, 27, 415 (April, 1957).

23. Schurmann, E. E. H., Jet Propulsion, 27, 863 (August 1957).

24. Laidler, K. J., "The Chemical Kinetics of Enzyme Action", Oxford University Press, London (1958).

25. Waksman, S. A. and W. C. Davison, "Enzymes", The Williams & Wilkins Company, Baltimore (1926).

26. Denbigh, K. G., Chem. Eng. Sci., 8, 125 (1958).

27. Aris, R., "Discrete Dynamic Programming", Blaisdell, New York, (1963).

28. Katz, S., Annals of the New York Academy of Sciences, 84, Art. 12, 441 (1960).

29. Chang, S. S. L., "Computer Optimization of Non-linear Control Systems by Means of Digitized Maximum Principle", A paper presented at I.R.E. International Convention, New York, (March, 1961).

30. Lee, E. S., "Optimization by Pontryagin's Maximum Principle on the Analog Computer", A paper presented at Fourth Joint Automatic Control Conference, University of Minnesota, Minneapolis, Minnesota (June, 1963).

31. Fan, L. T., Unpublished lecture notes from the course "Chemical Engineering Analysis II", Department of Chemical Engineering, K.S.U. (Fall Semester, 1962).

32. Pis'men, L. M., and I. I. Ioffe, International Chemical Engineering, 2, No. 4, 567 (1962).

NOMENCLATURE

a = a given constant

a^1 = a given constant

a_i = frequency factor in Arrehnius rate equation

a_n = $-B_n Mn - \alpha_n$

A = chemical specie

b = a given constant

b^1 = a given constant

b_n = $1 - \alpha_n$

B = chemical specie

c = a given constant

c_n = propellant exhaust velocity of stage n

C = chemical specie

d = a given constant

d^1 = a given constant

d_n = w_{xn}

D = defined by equation (163a)

e = x/y

E = sludge alkalinity

E_i = the activation energy of the chemical reaction

f = optimal net profit of the time interval from the present stage to the end of the process

$f_n(V)$ = minimum weight of rocket n achieveing velocity V

$F(\theta^n)$ = $\dfrac{K(\theta^n)t}{1 + K(\theta^n)t}$

$F_1(\theta^n) = \frac{dF(\theta^n)}{d\theta^n}$

$g = f/x$

$G(\theta^n) = \frac{h(\theta^n)t}{1 + h(\theta^n)t}$

$G_1(\theta^n) = \frac{dG(\theta^n)}{d\theta^n}$

h = optimal net profit taking λ as a parameter; function of control variable; chemical reaction rate constant

H = the Hamiltonian

i = node number

j = node number; an indication of the path the particle should follow

k = function of control variable; chemical reaction constant; a given constant

m = function of the control variable, total grid number

M_n = ratio of initial thrust to weight of rocket n

n = number of stages

N = total number of stages; total number of years for a resource to be available; last stage

O = objective function

P = interal profit; chemical specie

q = flow rate of the feed stream

Q = chemical specie

Q_N = total amount of wash water

r = flow rate of the recycle stream; rate of reaction

R = the ideal gas law constant; flow rate of wash water

s = total number of stete variables

t = residence time; total number of control variable; control variable

T = transformation function; absolute temperature

u = x/y

v = volumetric flow rate; w/x

V = reactor volume; design velocity of rocket

w = state variable; alkalinity of the wash water; initial gross weight of step rocket

w_x = that portion of jettison rocket weight which is constant

W_L = rocket system payload weight

x = state variable

X = chemical specie

y = state variable

Y = chemical specie

z = an independent variable; a variable introduced in the basic algorithm, equation (27)

Greek Letters

α = a given constant

β = a given constant

η = a given constant

λ = Lagrange multiplier; relative cost

δ = Kronecker delta, defined by $\delta_{im} = \begin{cases} 1, & i = m \\ 0, & i = m \end{cases}$

θ = residence time; control variable

ρ = residence time

APPENDIX

APPENDIX A

```
C       ROCKET A
        DIMENSION A(4),B(4),C(4),D(4),W(70,4),VA(70,4)
        DIMENSIONWA(70,4),WB(70,4),X(70,4),XA(70,4),XB(70,4),V(70,4)
      1 FORMAT(E10.4,E10.4,E10.4,E10.4,E10.4,E10.4)
        READ1,A(2),A(3),A(4),B(2),B(3),B(4)
        READ1,C(2),C(3),C(4),D(2),D(3),D(4)
        READ 1,P
        DO2M=1,70
        VM=M
        XA(M,2)=VM*50.0
        WA(M,2)=((1.0-A(2))*P+D(2))/(EXP(-XA(M,2)/C(2))-B(2)-A(2))
      2 PUNCH1,WA(M,2),XA(M,2)
        DO9I=3,4
        DO9N=1,70
        WA(N,I)=9999999999.9
        VN=N
        X(N,I)=300.0+VN*50.0
        V(N,I)=0.0
      4 X(N,I-1)=X(N,I)-V(N,I)
        DO13M=1,70
        XD=X(N,I-1)-XA(M,I-1)
        IF(XD-0.0)11,14,12
     11 XD=-XD
     12 IF(XD-25.0)14,14,13
     13 CONTINUE
     14 WA(N,I-1)=X(N,I-1)*WA(M,I-1)/XA(M,I-1)
        W(N,I)=((1.0-A(I))*W(N,I-1)+D(I))/(EXP(-V(N,I)/C(I))-B(I)-A(I),
        IF(WA(N,I)-W(N,I))6,6,5
      5 WA(N,I)=W(N,I)
        VA(N,I)=V(N,I)
        XA(N,I)=X(N,I)
        WB(N,I-1)=W(N,I-1)
        XB(N,I-1)=X(N,I-1)
      6 FA=X(N,I)-50.0
        IF(V(N,I)-FA)7,9,9
      7 V(N,I)=V(N,I)+50.0
        GO TO 4
      9 PUNCH1,WA(N,I),VA(N,I),XA(N,I),WB(N,I-1),XB(N,I-1)
        END
```

APPENDIX B

```
C     C.P. KINETICS PROBLEM
      DIMENSION DA(100,4),D(100,4),FA(100,4),F(100,4),DB(100,4)
    1 FORMAT(E10.4,E10.4,E10.4,E10.4,E10.4,E10.4,E10.4)
      READ1,T
      DO 3 M=1,100
      VM=M
      DA(M,1)=VM*0.03
    3 FA(M,1)=0
      DO6N=2,4
    4 DO 6 J=1,100
      VJ=J
      D(J,N)=VJ*0.03
      FA(J,N)=-0.99999E+20
      P=1.0
    5 Y=-3.0*P**2+10.0*P+12.0
      X=-P**2+9.0*P-6.0
      D(J,N-1)=(D(J,N)*(1.0+Y*T)+Y*T)/(1.0+X*T)
      F=X*T*(D(J,N)*(Y*T)/(1.0+Y*T))/(1.0+X*T)
      DO 12 M=1,100
      XD=D(J,N-1)-DA(M,N-1)
      IF(XD-0.0)10,10,11
   10 XD=-XD
   11 IF(XD-0.015)15,15,12
   12 CONTINUE
   15 F(J,N-1)=D(J,N-1)*FA(M,N-1)/DA(M,N-1)
      F(J,N)=B+F(J,N-1)/(1.0+Y*T)
      IF(F(J,N)-FA(J,N)) 9,9,8
    8 FA(J,N)=F(J,N)
      PP=P
      DB(J,N-1)=D(J,N-1)
      DA(J,N)=D(J,N)
    9 IF(P-5.0)20,20,6
   20 P=P+0.1
      GO TO 5
    6 PUNCH 1,DA(J,N),FA,FA(J,N),DB(J,N-1)
      STOP
      END
```

APPENDIX C

```
      PROGRAMMATICS
      DIMENSION GX(4),FX(4),G(4),F(4),GD(4),FD(4),X(4),Y(4),W(4)
      DIMENSION ZX(4),ZY(4),PA(4),RD(4),P(4),RC(4)
    1 FORMAT (F10.4,E10.4,E10.4,E10.4,E10.4,E10.4,E10.4)
      READ 1,T,P(2),P(3),P(4)
      READ 1,X(1),Y(1),W(1),PAP
    2 DO 4 N=2,4
      GX(N)=-P(N)**2+9.0*P(N)-6.0
      FX(N)=-3.0*P(N)**2+10.0*P(N)+12.0
      G(N)=T*GX(N)/(1.0+T*GX(N))
      F(N)=T*FX(N)/(1.0+T*FX(N))
      X(N)=(1.0-F(N))*X(N-1)
      Y(N)=(1.0-G(N))*(Y(N-1)+F(N)*X(N-1))
      W(N)=W(N-1)+G(N)*(Y(N-1)+F(N)*X(N-1))
    4 PRINT 1,X(N),Y(N),W(N),P(N),GX(N),FX(N)
      ZX(4)=0.0
      ZY(4)=0.0
      ZX(3)=F(4)*G(4)
      ZY(3)=G(4)
      ZX(2)=(1.0-F(3))*ZX(3)+F(3)*(G(3)+ZY(3)*(1.0-G(3)))
      ZY(2)=G(3)+(1.0-G(3))*ZY(3)
      DO 6 N=2,4
      PA(N)=-999999999.99
      RD(N)=1.0
    5 GX(N)=-PA(N)**2+9.0*PA(N)-6.0
      FX(N)=-3.0*PA(N)**2+10.0*PA(N)+12.0
      F(N)=T*FX(N)/(1.0+T*FX(N))
      G(N)=T*GX(N)/(1.0+T*GX(N))
      FD(N)=T*(10.0-6.0*PA(N))/(1.0+T*FX(N))**2
      GD(N)=T*(9.0-2.0*PA(N))/(1.0+T*GX(N))**2
      RD(N)=X(N-1)*ZX(N)*(1.0-F(N))+W(N-1)
      RC(N)=RD(N)+(Y(N-1)+F(N)*X(N-1))*(G(N)+ZY(N)*(1.0-G(N)))
      IF(RC(N)-RC(N))10,10,9
    9 RD(N)=RD(N)
      P(N)=PA(N)
   10 PA(N)=PA(N)+PAP
      IF(PA(N)-4.5)5,6,6
    6 PRINT 1,RC(N),P(N)
      GO TO 2
      END
```

APPENDIX D

```
C     DENOTEM D
      DIMENSION A(4),X(4),Y(4),ZA(4),RC(4),DA(5),PA(5)
      DIMENSION RA(4),RB(4),ZX(4),D(4),P(4),G(4),E(4)
    1 FORMAT (E10.4,E10.4,E10.4,E10.4,E10.4,E10.4,E10.4)
      READ 1,D(2),D(3),D(4),P(2),P(3),P(4)
      READ 1,A(1),X(1),Y(1),DAP,PAP,U,V
      READ 1,PTD,DTD
    2 DO 4 N=2,4
      E(N)=1.0/(1.0+D(N)*(1.0+P(N)))
      G(N)=1.0/(1.0+0.01*D(N)*(1.0+(1.0/P(N))))
      A(N)=A(N-1)*E(N)
      X(N)=G(N)*(X(N-1)+A(N-1)*D(N)*E(N))
      Y(N)=Y(N-1)+0.01*D(N)*X(N)
    4 PRINT 1,A(N),X(N),Y(N),E(N),G(N)
      ZA(4)=0.0
      ZX(4)=E.?
      ZA(3)=0.01*E(4)*G(4)*D(4)**2
      ZX(3)=0.01*D(4)*G(4)
      ZA(2)=ZA(3)*E(3)+ZX(3)*D(3)*E(3)*G(3)+0.01*E(3)*G(3)**2*D(3)**2
      ZX(2)=ZA(3)*G(3)+0.01*D(3)*G(3)
      DO 6 N=2,5
      RC(N)=-999999999999.99
      DA(N)=1.0
    7 PA(N)=0.00
    8 E(N)=1.0/(1.0+DA(N)*(1.0+PA(N)))
      G(N)=1.0/(1.0+0.01*DA(N)*(1.0+(1.0/PA(N))))
      RB(N)=A(N-1)*ZA(N)*E(N)+Y(N-1)
      RA(N)=RB(N)+G(N)*(X(N-1)+A(N-1)*DA(N)*E(N))*(ZX(N)+0.01*DA(N))
      IF(RA(N)-RC(N))10,10,9
    9 RC(N)=RB(N)
      P(N)=PA(N)
      D(N)=DA(N)
   10 PA(N)=PA(N)+PAP
      IF(PA(N)-PTD)8,8,11
   11 DA(N)=DA(N)+DAP
      IF(DA(N)-DTD)7,6,6
    6 PUNCH 1,RC(N),D(N),P(N)
      D(4)=U
      P(4)=V
      GO TO 2
      END
```

APPENDIX E

```
C     DENSIGN S
      DIMENSION A(4),X(4),Y(4),ZA(4),RC(4),DA(3),PA(3)
      DIMENSION RA(4),RB(4),ZX(4),D(4),P(4),G(4),E(4)
    1 FORMAT (E10.4,E10.4,E10.4,E10.4,E10.4,E10.4,E10.4)
      READ 1,D(2),D(3),P(2),P(3)
      READ 1,RC(1),X(1),Y(1),DAD,PAP,U,V
      READ 1,PTD,DTD
    2 DO 4 N=2,3
      E(N)=1.0/(1.0+D(N)*(1.0+P(N)))
      G(N)=1.0/(1.0+0.01*D(N)*(1.0+(1.0/P(N))))
      A(N)=A(N-1)*E(N)
      X(N)=G(N)*(X(N-1)+A(N-1)*D(N)*E(N))
      Y(N)=Y(N-1)+0.01*D(N)*X(N)
    4 PRINT 1,A(N),X(N),Y(N),E(N),G(N)
      ZA(3)=0.7
      ZA(3)=1.0
      ZA(2)=1.1*E(3)*G(3)*D(3)**2
      ZX(2)=0.01*D(3)*G(3)
      N=2
      RC(N)=99999999999.99
      DA(N)=1.0
    7 PA(N)=0.05
    8 E(N)=1.0/(1.0+DA(N)*(1.0+PA(N)))
      G(N)=1./(1.0+0.01*DA(N)*(1.0+(1.0/PA(N))))
      X(N)=A(N-1)*ZA(N)*E(N)+Y(N-1)
      RB(N)=X(N)+G(N)*(X(N-1)+A(N-1)*DA(N)*E(N))*(ZX(N)+0.01*DA(N))
      IF(RA(N)-RC(N))10,10,9
    9 RC(N)=RA(N)
      P(N)=PA(N)
      D(N)=DA(N)
   10 PA(N)=PA(N)+PAP
      IF(PA(N)-PTD)8,8,11
   11 DA(N)=DA(N)+DAD
      IF(DA(N)-DTD)7,6,6
    6 PUNCH 1,RC(N),D(N),P(N)
      D(3)=U
      P(3)=V
      GO TO 2
      END
```

THE OPTIMIZATION OF CHEMICAL REACTORS

by

CHEE-GEN WAN

B.S., National Taiwan University, 1959

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Chemical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1963

A brief introduction to the principle of optimality is given at the beginning by an illustration of it using the directed net work problem. Brief summaries of dynamic programming and the maximum principle algorithms follow.

In the application section, six problems are solved and presented in order of complexity. The sludge treatment problem is a fairly simple case for which identical analytical solutions could be obtained by both dynamic programming and the maximum principle algorithms. The step rocket problem resulted in a semi-analytical solution by the use of the end specification algorithm of the maximum principle. Regarding computational, this method is often much simpler than dynamic programming solution. The third problem is the optimization of a general consecutive first order biochemical reaction system in a series of ideal backmix reactors. The general working schemes were employed. The numerical solutions obtained by both methods were compared in detail. Denbigh's reaction system was solved by means of the maximum principle and the answers were compared with those obtained by Aris with the use of the dynamic programming approach.

The section on application also includes two problems of growth and predation. The solutions of these two problems, especially the second one, clearly show the advantage of employing the maximum principle approach over employing the dynamic programming method when the number of state variables is large.

The last section gives a general discussion on the relative merits of the two approaches.