At present, there are many industrial processes of a nonlinear character for which it is difficult to develop an effective industrial process control system because no efficient mathematical method is known to carry out the optimization procedure.

□ *This paper presents a flow chart description of a computer program incorporating a new optimization technique which will resolve many such problems. Although the mathematical basis for the technique is suggested, details and proofs are omitted—these will appear in a subsequent paper.*

□ *The technique has been successfully tested on a number of problems. Testing was conducted using a control system* (IBM 1710) *as well as both small and large computers* (IBM 1620, 7090).

A program for optimal control of nonlinear processes

by R. A. Mugele

general procedure

The development of a computer-directed control system for a complex industrial unit, such as a chemical plant, a petroleum refinery, or an electrical power distribution system, can be segmented into three major parts.

A *mathematical model* of the system must be developed. The basic model must describe at least the internal balances and transfer of material and energy, and reactions that occur within the system. This model is also useful in determining requirements for constructing, operating, or modifying the unit. The basic model may be refined by allowing for variability in operating conditions so that the response of the system to external changes will be shown and other useful information, e.g., accounting data, can be obtained.

Next, an *optimizing procedure* must be found which is applicable to the model and which will permit calculation of better, or best, operating conditions, according to preset criteria. This is the type of problem to be considered in detail in the present paper.

Finally, a *control procedure* must be formulated for conveying to the process (by means of controller set-point and reset rates, switches, clocks, and other devices) the improved mode of operation which has been computed by the optimizing program.

mathematical considerations

In general, some sort of approximation to an industrial system may be obtained with a linear model. In the linear model, things change at a constant rate relative to the basic variables. For example, if each ton per day increase in throughput at a plant produces $100 per day increase in profit, we have a linear situation. But there are cases in which the linear model is inadequate (not precise enough, or misleading as to trends). Perhaps, in the above example, each additional ton per day increase in throughput produces a smaller improvement than the preceding increase. In such cases, solutions cannot be obtained by using the relatively direct and simple methods of linear programming. We require nonlinear programming techniques which will optimize nonlinear objective functions and restraints.

Several workers have devised methods of nonlinear programming and these have been applied to such diverse problems as gasoline blending,[1] chemical plant problems,[2] and algebraic test problems.[3,4]

However, these methods do not apply to cases in which several of the restraints have complicated nonlinear forms. Such cases become important when we consider something as complex as a chemical plant or a petroleum refinery. Here the interaction of physical, chemical, and economic factors leads to an involved objective function. Also, practical considerations such as product specification and component availability lead to a number of restraint inequalities, some of them highly nonlinear. Thus, we face the problem of nonlinear programming with complicated nonlinear restraints. To handle such problems, extensions of existing nonlinear methods have been proposed.[5,6] In the following sections we shall describe such proposals and also some effective methods which have already been developed and applied to full-scale problems.

definitions and notation

Before describing optimization programs and their relation to process and control, we offer some definitions. Variations from these will be found in other publications, as is usually the case in a relatively new field. But in the present paper, the following definitions and accompanying notation will be used consistently. Since we are simultaneously concerned with computer programs, some use of Fortran notation has been made.

Control variables. Quantities $X1$, $X2$, etc., that are to be used in implementing the result of optimization, i.e., producing optimal control of a process. The X's may be subject to bounds and restraints, but they are expected to vary during solution of the problem. The function F and the R's (see definitions below of objective function and restraints, respectively) are to be expressed, either explicitly or implicitly, in terms of the X's. The X's form a K-dimensional set in which a point determines the state of the system. In the dynamic mode, at least one of the X's will be a variable expressing or implying time.

Uncontrollable variables. Quantities $V1$, $V2$, etc., that are not used in implementing the result of optimization, but do enter as parameters in the model, varying primarily as a result of disturbances in the inputs.

Bounds. Quantities which define the range of the X's. Upper bounds are $XU1$, $XU2$, etc. Lower bounds are $XL1$, $XL2$, etc. The range of $X1$ is defined by $XU1 - XL1$. Where the problem conditions do not specify such bounds, arbitrary values may be provided (e.g., large negative $XL1$ and large positive $XU1$). The relations $XL \leq X \leq XU$ define a set of states including the feasible points as a subset.

Increments (steps). Quantities $D1$, $D2$, etc., to be used in computing the effects of changing control variables $X1$, $X2$, etc., during the optimization calculation. These increments are used to change the X's positively or negatively, individually or in combinations, as required by the program. The D's have assigned starting values, but are increased or decreased during the computation, as required by the program.

Tolerance. A quantity T, positive and usually small, which defines the minimum allowable ratio of D's to the starting values of the D's. When D's have been reduced often enough to match this tolerance, the optimization calculation stops.

Restraints. Quantities $R1$, $R2$, etc., which have been chosen, formulated, or programmed, so as to impose limitations on some action, product, process, or method. In simple cases, R's may limit some directly verifiable or measurable quantity (e.g., a minimum octane number in gasoline blending, or the total mass charged to a blast furnace). In more complex cases, R's may involve technical and economic quantities that are not directly verifiable or measurable (e.g., the yield of an undesirable intermediate in a chemical process, or the expected additional cost of purchasing some raw material if it is required in small lots). The R's are required to be non-negative at the end of each major cycle. Some caution must be exercised in dealing with restraints, to avoid thinking of properties of a particular R as properties of the optimization problem. For example, the functions $R1 \equiv X1 - X2$ and $R2 \equiv X1 ** 3 - X1 ** 2 * X2 + X1 * X2 ** 2 - X2 ** 3$ are interchangeable as restraints, since they are non-negative in exactly the same region; yet their gradients are very different.

Feasible. Having all control variables within (or at) bounds, and no restraints negative. A feasible region is one in which all points are feasible ($XL \leq X \leq XU$ and $R \geq 0$).

Objective function. A quantity F that has been chosen, formulated, or programmed, so as to measure the desirability of some action, product, process, or method. In simple cases, F may be a directly verifiable or measurable quantity (e.g., the daily production of a particular grade of paper at a mill, or the number of typewriters produced on an assembly line). In more complex cases, F may involve technical and economic quantities that are not directly verifiable or measurable (e.g., safety penalties, tax estimates, amortization).

Optimization. Determining a maximum for an objective function.

Local optimum. A feasible point such that any feasible small change will cause a decrease in the objective function.

Restrained optimum. A local optimum (condition of system, or control array) that has some control variable at its bound, or some restraint equal to zero, i.e., just feasible.

Global optimum. That local optimum which has the highest value of the objective function.

Point. An element of the K-dimensional space determined by the K control variables (X's).

Base point. A point from which probes originate.

Probe. A point established by increasing or decreasing a particular control variable ($X \pm D$).

Solution. A point calculated by means of an optimization program to be a local optimum.

Major cycle. A cycle of computation which leads to a feasible improvement in the objective function.

Region. A set of points.

Convex (set or region). Having the property that $A * X + (1 - A) * Y$ is in the set whenever X, Y are both in the set and $0 \leq A \leq 1$.

Vector. An ordered set of quantities (components, or major variables) which determines the state of a system (operating vector). It also determines a change in the state (gradient or step vector). As an example, consider the condition of a reactor to be specified by feed temperature T, catalyst concentration C, and pressure P. Then we may write $\tilde{X} \equiv (T, C, P)$ to represent the operating vector. A function of this vector is evaluated according to the separate values of the components; for example, if F is the daily product value for the reactor, we may write $F \equiv f(\bar{X})$ to indicate that F is some known function of T, C, and P.

Gradient. A vector which expresses the local variation of a function. In the case of $F(X1, X2, \cdots, XK)$, the gradient vector is the set $(\partial F/\partial X1, \partial F/\partial X2, \cdots, \partial F/\partial XK)$. This may be expressed by grad F or ∇F. It determines the direction of most rapid change in F. This direction is sometimes described as normal to $F =$ constant.

Linear. Having a constant gradient. To emphasize the far-reaching consequences of this simple restriction, we note the success already achieved by linear programming.[1,7,8,9] This success is in large measure due to the advantage of having a constant gradient and therefore one which does not have to be recalculated at each step toward the optimum.

Projection. A vector which expresses the local variation of a function in a special direction. When the "projection of Z on RJ" is specified, this vector has the form $P = Z + A * \nabla RJ$. It is also normal to ∇RJ. Hence, we have $\sum ZN * (\partial RJ/\partial ZN) + A * \sum (\partial RJ/\partial ZN) ** 2 = 0$, from which A may be calculated. When the "projection of Z on RJ and RM" is specified, P will have the form $Z + A1 * \nabla RJ + A2 * \nabla RK$ and will be normal to both ∇RJ and ∇RM, with the coefficients $A1$ and $A2$ to be determined from two equations. This procedure may be extended to projections of higher order, but is necessarily limited to projection on $K - 1$ confluent restraints, where K is the number of control variables. In particular, if $Z = \nabla F$, then P is the "gradient projection of F."

the basic problem

The principal problem considered in this paper may now be stated in mathematical form.

Given (1) F, a real single valued function of set of K real bounded control variables $(X1, X2, \cdots XK)$, (2) a set of restraint functions ($R1, R2 \cdots$), each of which is a real single valued function of the control variables, and (3) that there exists a feasible subset of the K-dimensional space determined by the control variables; *we are required to find* (1) a global optimum.

nonlinear optimization

We will now describe some nonlinear programming techniques that have been applied (with varied success) to the above problem. These techniques are fairly well known and serve as a useful point of departure in the development of more general nonlinear methods.

Analytical method. Use elementary calculus to determine all proper maxima possessed by the objective function. The highest of these maxima may be the solution, if feasible. Next determine restrained local maxima subject to each of the restraints and bounds (Lagrange's method of indeterminate multipliers is convenient for this). The highest of these may be the solution, if feasible and not surpassed in the earlier calculations. Next, do the same for local maxima subject to *pairs* of restraints or bounds. Continue in this way until all local maxima on groups of restraints or bounds have been isolated. Then select the global maximum. Some of the mathematics required in this method may be found in the literature.[7,10,11,12]

Grid method. Space XJ values at reasonable intervals for each variable (e.g., divide the entire range of each variable into ten parts). Calculate F at each feasible member of the $(i_1 + 1)(i_2 + 1) \cdots (i_k + 1)$ resulting grid points (where i_u is the number of intervals for variable XJ). Designate the point with the highest F value as the "first solution." If more precision is required, explore the vicinity of the first solution on a finer mesh than before (or else apply some interpolation device, such as by using a second-degree hypersurface). Continue this until satisfactory precision is obtained. Check the solution by making gradient or gradient-projection excursions. To reduce computing time when using this method, eliminate portions of the grid whenever possible by considering analytical properties of the restraints. This is sometimes called the "case study" method in design work.

Monte Carlo methods. Use a random number table (or generator) to select a point in the bounded region. Retain this point if feasible, and calculate its F. Repeat for another random point. Retain the new point if its F is higher than that of the old point. Continue in this way for a predetermined number of points (calculated from statistical considerations to give a preset confidence level). The last retained point is taken as an approximate solution. Check this solution by making gradient or gradient-projection excursions.

A variation of this method uses the retained point as a start for the next move. Then the direction of the move is determined at random, but the magnitude is fixed. If a certain number of moves does not produce an improvement, the magnitude of the move is decreased. Brooks[3] discusses some of the possibilities of these methods.

Cross-section or univariate method. With fixed starting values of

$X2, X3 \cdots, XK$, vary $X1$ by pre-selected steps, spanning its range. At each point, calculate F if feasible. Select the best F so far and set the corresponding value of $X1$. Then vary $X2$ in the same way, and set $X2$ according to the best feasible point so far. Continue in this way until a complete cycle produces no further change in F. As with the grid method, a finer mesh may be introduced after the final approximation is achieved. As usual, the solution must be checked by making gradient or gradient-projection excursions.[13]

Blocking methods. Consider the possibility of reducing the region to be examined in a systematic rather than a random manner (cf. Monte Carlo methods). The general idea is to eliminate about half of the bounded region at each step. These methods may be satisfactory for a small number of variables, but they become unmanageable for a large number of variables because the procedure requires a knowledge of the objective function and of the restraint functions for at least the "corners" of the region to be eliminated. In a region with k major variables, there are 2^k such corners. For $k = 20$, this number is more than one million.

Gradient projection method. From a feasible starting point, follow the gradient until stopped by a restraint. Follow this restraint, via gradient projections on tangent hyperplanes, until stopped by another restraint. Then follow the two restraints, via gradient projections on intersections of tangent hyperplanes, until stopped by a restraint. Follow this restraint, via gradient projections on tangent hyperplanes, until stopped by another restraint. Continue in this way until "cornered" (at a restrained maximum) or stopped by a zero gradient or gradient projection (at a proper maximum or partially restrained maximum). Check the final calculated point to see whether it is a solution. Due to the use of tangents to approximate restraints, it will sometimes be necessary to return to the feasible region when a calculated step has led to a non-feasible point. This is done by an interpolation procedure.[6]

Among the above, probably the gradient projection method is most adequately developed for wide use in systems that have non-linear objective functions.

Probe Program description

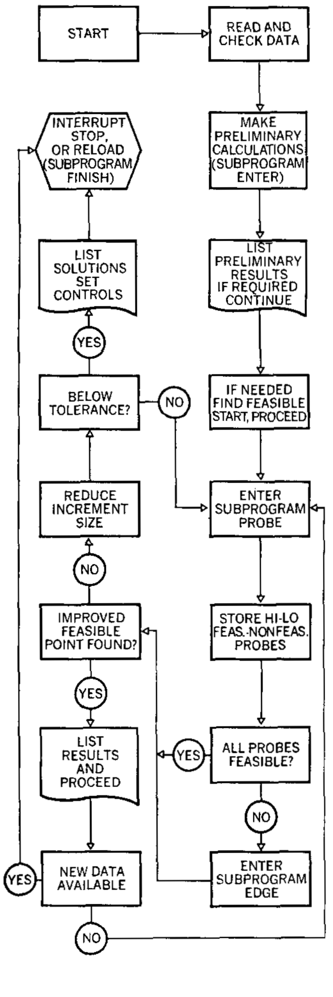
We will present our approach to the basic problem stated above by describing, at the flow chart level, a computer program incorporating the mathematical technique. This computer program will be referred to as the Probe Program. The program employs a new optimizing procedure, which applies some principles not found in any of the methods previously described. It does not require continuity or the existence of derivatives for either the control variables or the objective or restraint functions. It does not require the feasible region to be convex. It is convergent, with a self-adjusting feature for the increment size. It has been programmed with special consideration for the needs of control systems and has been successfully tested on the IBM® 1620, 1710, and 7090 computers. In the development of the Probe Program, certain theorems for existence and convergence have been derived. However, the object of the present paper is to describe the program; the precise statements and proofs of these theorems are reserved for a later paper. We know of no "pathological" cases corresponding to an industrial system for which the Probe Program would fail (assuming, of course, that the computer involved has appropriate capability).

The Probe Program attacks the basic problem by finding (1) a feasible point $X(N)$ and (2) a sequence of points $X(J)$ with the properties (a) the first $X(J)$ is $X(N)$, (b) for $I > J$, $F(X(I)) > F(X(J))$, and (c) $F(X(L))$ is a local optimum, where $X(L)$ is the last $X(J)$.

The program starts by making probes (excursions) from a feasible point, in directions of increasing and decreasing X's. The first successful probe (feasible point with improved F) determines the next point of the course, and this point becomes a base for further probing. The course is continued until no improved feasible point is achieved, possibly due to "approaching" a restraint. In a later section, we shall describe a procedure for "following" a restraint.

When a course terminates "away from" all bounds and restraints, approach to a solution is indicated. When the appropriate procedures have failed to produce any advance, it is time to decrease the increment sizes and repeat the entire procedure. After repeated reductions, the tolerance is matched, and the calculation stops. The point resulting from the last major cycle is reported as a solution.

Figure 1 Flow chart for Probe Program


START
READ AND CHECK DATA
MAKE PRELIMINARY CALCULATIONS (SUBPROGRAM ENTER)
LIST PRELIMINARY RESULTS IF REQUIRED CONTINUE
IF NEEDED FIND FEASIBLE START, PROCEED
ENTER SUBPROGRAM PROBE
STORE HI-LO FEAS.-NONFEAS. PROBES
ALL PROBES FEASIBLE?
YES
NO
ENTER SUBPROGRAM EDGE
IMPROVED FEASIBLE POINT FOUND?
YES
LIST RESULTS AND PROCEED
NEW DATA AVAILABLE
YES
NO
INTERRUPT STOP, OR RELOAD (SUBPROGRAM FINISH)
NO
REDUCE INCREMENT SIZE
BELOW TOLERANCE?
NO
YES
LIST SOLUTIONS SET CONTROLS

A number of practical additions to the basic logic outlined above are included in the program. One of these involves assigning priorities to probes, and to restraint calculations, in order to improve the efficiency of calculation. Another involves special calculations which improve efficiency in a "ridge" situation (that is, a situation where ∇F changes greatly within one step). Another consists in designating certain variables as "discrete," in which case their increments will be integer multiples of specified values (usually unity), and the variables themselves will take on only specified values (usually integers).

These variations are incorporated in the Probe Program but are omitted from the "Compact Probe Program," which is the subject of another report.[14]

The Probe Program consists of an executive routine and several major subprograms—called, EDGE, ENTER, FENCE, FINISH, FOUL, OBJECT, ORDER, PROBE, and RIDGE (which for notational convenience will be designated in small capitals). A schematic flow chart for the Probe Program is presented in Figure 1.

In Figure 1, we see that the basic "optimizing" logic starts at the fifth block (entering subprogram PROBE), which is described later. The assumption here is that a feasible starting point is available. If not, one would normally be found by means of subprogram FOUL, described later.

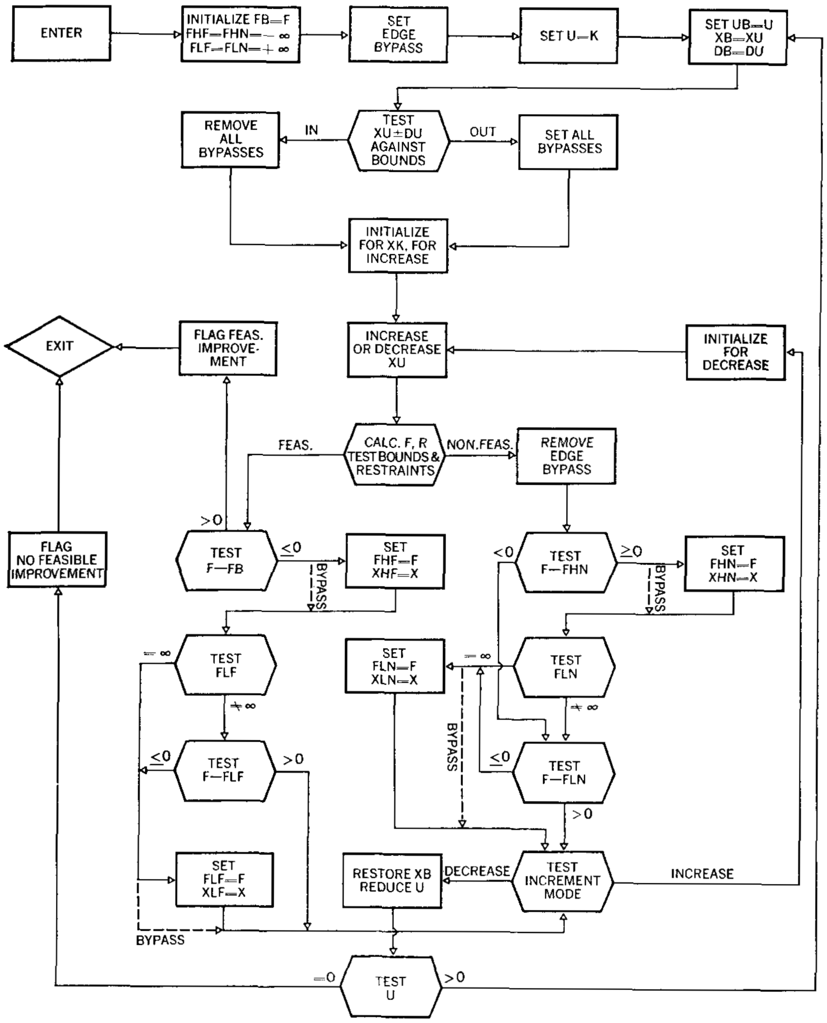
In the course of subprogram PROBE, the critical probes (highest and lowest feasible and non-feasible) are stored. If any probe is non-feasible, the gate to subprogram EDGE is opened, and that subprogram will be used to follow a restraint, if needed. In any case, when an improved feasible point is found, the results are listed as required and the control situation is reviewed (Is there an "interrupt" situation? Is an alarm set? Are new data available?). If no control action is required, the program returns to subprogram PROBE for another cycle. When a point is reached where the subprogram fails to produce feasible improvement, the increments are reduced (halved). When the ratio of increments to their initial values falls below a preassigned tolerance, the solution is listed if required, and the controls are reset.

Thus we see that the Probe Program is simple in concept. However, it is suitable for use with large and complicated problems. In the sections that follow, the entire program is discussed in enough detail to guide coding for any stored-program computer.

optimizing subprograms

We will now describe the optimizing subprograms. Subprogram PROBE is shown in flow form in Figure 2. This program implements the probing procedure which was mentioned in the Probe Program description. The general procedure is as follows. With

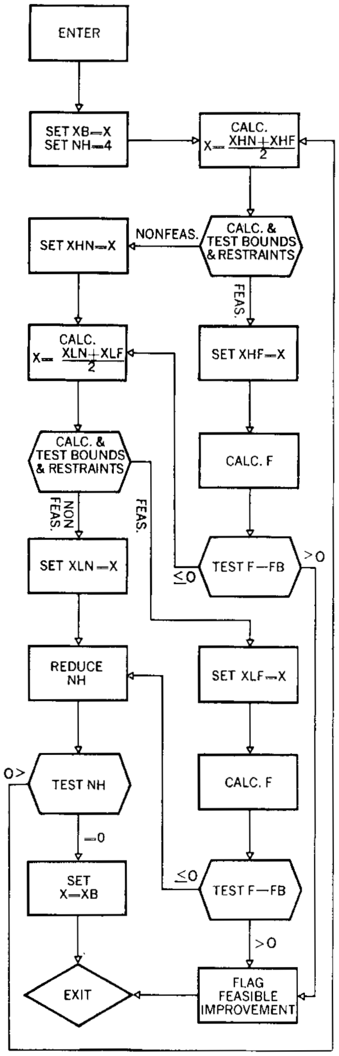
Figure 2 Flow chart for subprogram PROBE



Notation—Figure 2

B	*base point*
D	*increment*
F	*objective function*
FB, FHF, . . .	*value of* F *at* B, HF, . . .
HF, HN	*highest feasible, non-feasible probes*
K	*no. of control variables*
LF, LN	*lowest feasible, non-feasible probes*
R	*restraint*
U	*control variable index*
X	*control variable*
XL	*lower bound of* X
XU	*upper bound of* X
XB, XHF, . . .	*value of* X *at* B, HF, . . .

Figure 3 Flow chart for subprogram EDGE



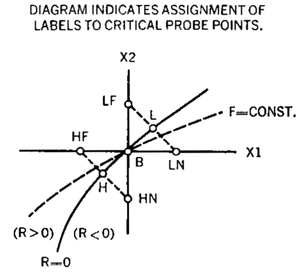


DIAGRAM INDICATES ASSIGNMENT OF
LABELS TO CRITICAL PROBE POINTS.
X2
LF
L
F=CONST.
HF
X1
B
LN
H
HN
(R>0)
(R<0)
R=0

Notation

B	*base point*
F	*objective function*
FB, FHF, ...	*value of* F *at* B, HF, ...
HF, HN	*highest feasible, non-feasible probes*
LF, LN	*lowest feasible, non-feasible probes*
NH	*index for half-steps*
X	*control variable*
XB, XHF, ...	*value of* X *at* B, HF, ...

preselected step sizes for all control variables, probe from a feasible point (base point) in the (positive and negative) directions of the control variables, in a preset order. Desist as soon as a feasible point with a higher value of F is found. Use this point as a new base point and set a flag to indicate this condition. If no improved feasible point is found, set a flag to indicate this condition, before leaving the subprogram. While probing, upgrade the high and low feasible and non-feasible probes (see description of subprogram EDGE) when needed. Also remove the "bypass EDGE" flag whenever a non-feasible probe occurs.

Subprogram EDGE is shown in flow form in Figure 3. This program directs the procedure after a restraint (or set of restraints) has been sensed during the operation of subprogram PROBE. It directs the course of computation so as to "follow" the "nearest" restraint, unless improvement is available in a direction away from the restraint. An outline of the procedure follows.

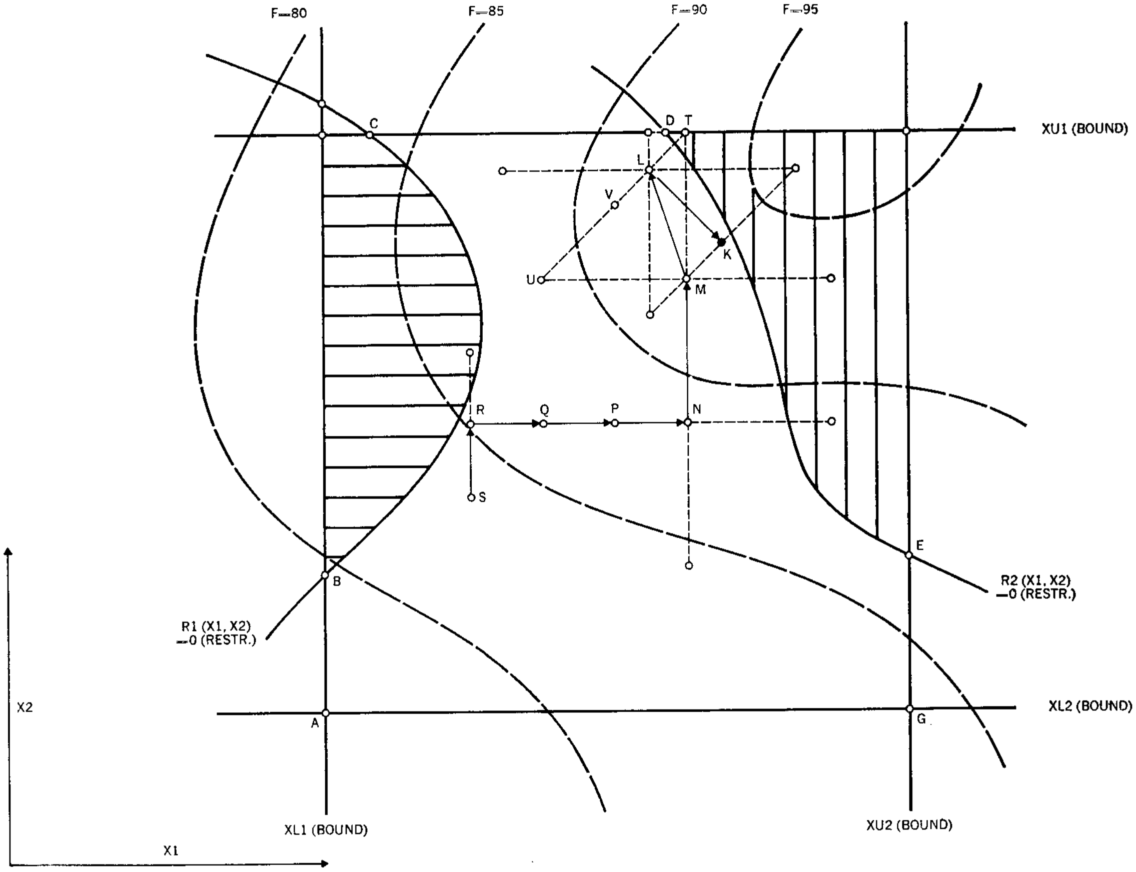
Since at least one of the probe points is non-feasible and all probes within one step of the base point have been evaluated (otherwise we would not have entered subprogram EDGE), we know that highest and lowest feasible (denoted HF, LF respectively) and non-feasible (denoted HN, LN respectively) points have been evaluated. We now test the midpoint between HF and HN. If feasible, evaluate. If not better than the base point, replace HF by the new point and proceed similarly with LF and LN. Repeat the procedure using HF and HN (as well as the procedure using LF and LN) four times before leaving this subprogram.

an example of PROBE and EDGE operation

In Figure 4, some of the concepts involved in the nonlinear problem and its solution are illustrated. For a certain chemical reaction, the yield of a desirable product (F) is known as a function of pressure ($X1$) and temperature ($X2$). Natural lower bounds exist for temperature (since this reactor will always be heated, never cooled) and for pressure (since the reactants enter under positive pressure). Practical upper bounds also exist, since either a temperature greater than $XU2$ or a pressure greater than $XU1$ would be harmful to the reactor. Hence the feasible region for this problem must be within the indicated rectangle. But there are further conditions. (1) For certain combinations of temperature and pressure, polymerization occurs. This is undesirable, and is therefore ruled out by the relation $R1\ (X1, X2) \geq 0$. In Figure 4, this removes the shaded area at the left from the feasible region. (2) For other conditions of temperature and pressure, the product will contain more than some specified amount of an undesirable component, even though the yield of the desirable product is high. This condition is ruled out by the relation $R2(X1, X2) \geq 0$. In Figure 4 this removes the shaded area at the right from the feasible region. The remaining unshaded area ABCDEGA represents the feasible region.

In Figure 4, the combined operation of PROBE and EDGE is illustrated by course SRQPNMLK. At S, the initial probe increases $X2$, and an improved feasible point R is achieved. At R, the restraint $R1 = 0$ is encountered when increasing $X2$, but an increase in $X1$ produces improvement. At N, four feasible advances have been made with the basic step size. So the step size is increased, and M is the next improved feasible point. The probes from M do not produce an improved feasible point, but interpolation between the low-feasible probe (U) and the low-non-feasible probe (T) leads to L (second half-step), an improved feasible point. From L, a similar procedure takes us to K. The point K is very close to the restrained optimum which represents the solution of this problem. In fact, any of the points M, L, or K might represent a good operating condition for the reactor. However, if a close tolerance has been chosen, the continued applica-

Figure 4 Operation of subprograms PROBE and EDGE



tion of PROBE and EDGE with reduced step sizes will produce still further feasible improvements.

Probe executive program

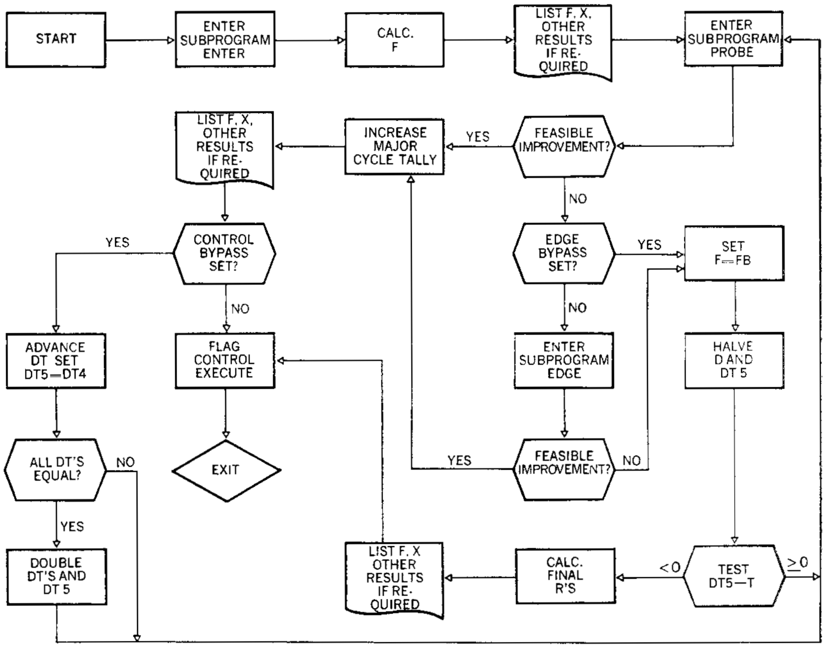
The main, or executive program is shown in flow form in Figure 5. In addition to connecting the subprograms and organizing the necessary initializing, finalizing and branching, the executive program also calls in the subprograms which define the particular problem to be solved. Such subprograms are briefly described in the following paragraphs.

The executive program will branch to ENTER only at the start of the calculation. Accordingly, this subprogram should include all one-time operations needed to set up later calculations. It will set the values needed to satisfy existing control requirements. It may set initial values for D's, or use values stored during a previous cycle.

The executive program and the subroutines will branch to OBJECT whenever the objective function F is to be calculated. This subprogram computes F from the currently stored values of $X1$, $X2$, etc. It returns the result to a fixed location and branches back. If desired, a tally can be programmed to advance each time OBJECT is entered; this will provide a record of how many times F has been recalculated.

The executive program and the subprograms will branch to FENCE whenever the restraint functions $R1$, $R2$, etc. are to be calculated. This subprogram computes $R1$, $R2$, etc. from the currently stored values of $X1$, $X2$, etc. If any restraint comes out negative, the subprogram sets a non-feasible flag and exits. If no restraint is negative, the exit is reached with the feasible flag set. The non-feasible flag can also be set by overstepping a bound; but this is handled by the executive program. Figure 6 shows the simple logic of this subprogram.

Figure 5 Flow chart for executive program



Notation—Figure 5

B	*base point*
D	*increment*
DT	*historical increment array*
DT4, DT5	*successive increment indices*
F	*objective function*
FB	*value of* F *at* B
R	*restraint*
T	*tolerance*
X	*control variable*

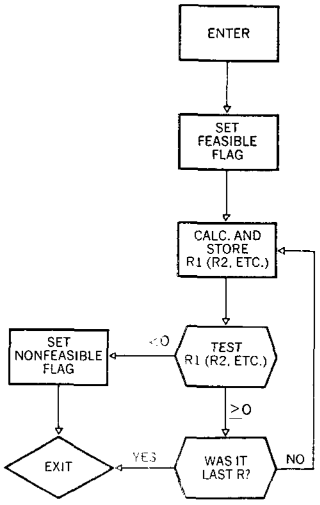
The executive program will branch to FINISH under one of these conditions: (1) at the conclusion of the optimization, (2) if control is to be initiated, (3) if an error is indicated, (4) if a feasible starting point is not available. Subprogram FINISH will provide instructions for implementing control, correcting errors, or setting alarms, as required.

description of other subprograms

As previously indicated, a number of additional subprograms are included for purposes of flexibility and efficiency, although they might be omitted for the sake of compactness. These subprograms are briefly described in the following paragraphs.

The starting point will ordinarily be feasible, since the Probe Program will be running on a process which runs within the assigned restraints and bounds. However, if a feasible starting point is not known, one is calculated via subprogram FOUL as follows. Set up the pseudo-objective function FR whose value is that of the algebraically smallest R. Run the optimizer on this function (normally without restraints), *not* to a maximum of FR, but only until FR becomes positive. We then have a feasible point, since all R's must be positive. Then we return to the main program.

Figure 6 Flow chart for subprogram FENCE



A fixed order of probing, and a fixed order of calculating the restraints, may not be the most efficient. Subprogram ORDER is used to make this ordering more efficient. The procedure used in probing is to bring the successful probe to the head of the list, moving its opposite to the end. For example, if the probe order in a 3-variable problem has been -2, $+3$, $+1$, -1, -3, $+2$, and a successful probe is made on the fourth trial (-1), the new probe order becomes -1, -2, $+3$, -3, $+2$, $+1$. In the ordering of restraint calculations, priorities are assigned according to the number of times a restraint turns up negative during a major cycle; in this way, the number of restraint calculations needed to discover non-feasible points is usually reduced.

In some problems, progress may be slowed somewhat in regions where gradients change sharply within one step (although this is not common with well designed processes, and indicates some danger of instability in the system). For such cases, RIDGE is used to increase efficiency. Suppose that probes from the base point (at full step size) have failed to indicate an impending restraint, yet have also failed to yield an improved point. Then before reducing step size, proceed as follows. Between the two highest-valued probes, evaluate the midpoint. Exit from the subprogram if the midpoint is non-feasible (failure) or if it is feasible and higher than the base point (success). Otherwise, use the evaluation together with a quadratic approximation to estimate the maximum F between the critical probe points.

control strategies

The Probe Program is compatible with various modes of control. Some of these will be described briefly in the following paragraphs, with an indication of how the Probe Program applies.

Steady-state mode. The steady-state condition of a process is one in which changes in control, once made, will remain fixed for a long time (for example, in a continuous chemical plant with uniform inputs and outputs). In this case, the Probe Program is to be used whenever a significant change in inputs or process conditions occurs. The results will provide an operator guide, indicating the setpoints for control instruments to attain optimal operation. Occasionally, the Probe Program can also be used to determine whether some operating conditions far different from normal might provide a still better operation.

Dynamic mode. The dynamic condition of a process is one in which changes in control are made frequently (for example, in a catalytic process with several reactors, where the catalyst activity continually varies). In this case, the Probe Program is to be used in the closed loop, being called in after each new data-gathering cycle to start reoptimizing the operation. In a case where the data cycles are far apart, the control variables should be not only set points, but also derivatives, so that control conditions are advanced smoothly. In a case where the data cycles are close together, the Probe Program, operating after several stages of data processing, may not have time to complete the optimization calculation to the specified tolerance. In this case, (see Figure 5) an exit to the control cycle is permitted at the end of any major cycle of optimization. When this exit is used, an improved, although not strictly optimal, control strategy is directed to the process through the closed loop.

Dynamic scheduling mode. The dynamic scheduling condition of a process is one in which changes in control are made frequently, while changes in the objective or the restraints also occur (for example, in a refinery, where restraints vary with shipping schedules and seasons, and the objective may be a profit function integrated over several weeks). In this case, the Probe Program is to be used both in off-line calculation (as described for the steady-state mode) and in the closed loop (as described for the dynamic mode). Included in the off-line calculations there will occasionally be some suboptimizations, which will be applied to simplify the subsequent closed-loop calculation.

application to a profit function

As an example, suppose we have a process with five control variables, and wish to optimize a profit function G, integrated over six future time periods. This presumes that we are prepared to solve a 30-variable optimization problem as indicated in Table 1.

The problem here is to optimize $F = W1 * G1 + \cdots + W6 * G6$ where the G's are values of the objective function for the successive time periods, and the W's are weighting factors, related to the reliabilities of forecasts for the respective time periods. In

Table 1 Optimization—30-variable problem

time period	1	2	3	4	5	6
control variable	variable in dynamic optimization					
$X1$	$X11$	$X12$		. . .		$X16$
$X2$	$X21$	$X22$		. . .		$X26$
$X3$	$X31$	$X32$		. . .		$X36$
$X4$	$X41$	$X42$		. . .		$X46$
$X5$	$X51$	$X52$		. . .		$X56$

addition to the "integrated" objective function, there will also be certain restraint functions which apply over the span of six time periods. In this setting, the 30-variable problem is solved off-line. The result is a schedule on which the process control is started. After the first time period, the optimizing calculation is carried out on-line, and only as a 5-variable problem. In the 5-variable calculation, only the schedule for the sixth time period is calculated, the other X's being treated as constants. The former values of $X16$ to $X56$ now become scheduled values of $X15$ to $X55$, etc. At any time when a check calculation of $X11$ to $X51$ (the control variables next to be implemented) is desired, this can also be run as a 5-variable optimization, using the previously calculated values of $X11$ to $X51$ as a starting point, again treating the remaining X's as constants. This check calculation will ordinarily be used when significant changes in uncontrollable variables are known to occur, thus producing changes in the objective or restraint functions.

example of Probe Program output

Problems of different degrees of complexity have been used to test the Probe Program. To conclude, the output of a 4-variable test problem is exhibited to give the reader some indication of the relations between problem data and program parameters.

In this problem (see Table 2) F represents the profit per unit of a certain chemical product. This has been formulated as a quadratic function of four principal process variables. The total production rate R is required to be not greater than $K1$ and not less than $K2$. This production rate has also been formulated as a quadratic function of the four principal controllable variables.

Looking at Cycle 1, we see that the objective function F has increased to 29.075 from an initial value of 29.005. This was done by increasing $X2$ to .0500 from its initial value of .0000. The restraints remain positive.

Looking at Cycle 4, we see that the objective function has increased still more to 29.0785. Although probes have been made with other variables, $X2$ is the only one that has given a useful change, increasing still more, to .0524. The first restraint has

Table 2 Example of output Probe Program—4-variable problem

$F = F(X1, X2, X3, X4)$, and F is a quadratic function
$R1 = K1 - R(X1, X2, X3, X4)$, and $R1$ is a quadratic function
$R2 = -K2 + R(X1, X2, X3, X4)$
Upper bound for each X is 2.0
Lower bound for each X is −2.0
Tolerance is .0000001
Initial step size for each X is 0.1

CYCLE	$X1$	$X2$	$X3$	$X4$	$R1$	$R2$	F
START	−2.0000	0.0000	2.0000	−2.0000	0.1933	1.8067	29.005
1	−2.0000	0.0500	2.0000	−2.0000	0.0089	1.9911	29.075
2	−2.0000	0.0516	2.0000	−2.0000	0.0031	1.9969	29.077
3	−2.0000	0.0523	2.0000	−2.0000	0.0002	1.9998	29.0784
4	−2.0000	0.0524	2.0000	−2.0000	0.0000+	1.9999+	29.0785+
5	−2.0000	0.0524	2.0000	−2.0000	0.0000+	1.9999+	29.0785+
6	−2.0000	0.0524+	2.0000	−2.0000	0.0000+	1.9999+	29.0785+
7	−2.0000	0.0524+	2.0000	−2.0000	0.0000+	1.9999+	29.0785+
END	−2.0000	0.0524+	2.0000	−2.0000	0.0000	2.0000	29.0785+

become zero (indicating the maximum allowable production rate). The step-size has become very small, indicating approach to a maximum.

Looking at the solution, we see that no further improvement was made after Cycle 4. The solution is fully restrained with $X1$ and $X4$ at their lower bounds, $X3$ at its upper bound, and $X2$ at a value which makes $R1$ exactly zero.

response optimization

In the present paper, the main problem considered has been that of optimizing a process with respect to certain control variables. On this basis, we have not specifically considered the means of optimizing the response of a control system. However, this problem has been attacked by a number of workers, and some results appear in the literature of control systems.[1,15,16] A special application of the Compact Probe Program to a study of this type (response optimization) has also been reported.[17]

ACKNOWLEDGMENT

The material presented in this report was developed in the Control Systems group of the General Products Division of IBM. Appreciation is due to many members and former members of this group, and to members of the Advanced Systems Development Division and Research Divisions of IBM engaged in related work, for suggestions, interest, and encouragement in this project.

Some of this material was presented May 1, 1962 at the Spring Joint Computer Conference, San Francisco, and published in the Proceedings of that conference. We appreciate the permission of the American Federation of Information Processing Societies to publish this paper in its present form.

REFERENCES

1. Manne, A. S., *Scheduling of Petroleum Refinery Operations,* Harvard University Press, Cambridge (1956).

2. Kalman, R. E., Koepcke, R. W., "The Role of Digital Computers in the Dynamic Optimization of Chemical Reactions," IBM Research Report RC-77, IBM Research Center, Yorktown Heights, N. Y. (1958).

3. Brooks, S. H., "A Discussion of Random Methods for Seeking Maxima," *Operations Research,* v. 6, p. 244 (1958).

4. Wolfe, Philip, "Computational Techniques for Nonlinear Programs," *Princeton Conference on Linear Programming* (March 13-15, 1957).

5. Roberts, S. M., Lyvers, H. I., "The Gradient Method in Process Control," *Ind. & Eng. Chem.,* v. 53, #11, p. 877 (Nov. 1961).

6. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming,—Part I, Linear Constraints," *J. Soc. Industrial and Applied Math.,* v. 8, p. 181 (March 1960); "Part II, Nonlinear Constraints," to be published.

7. Dorfman, R., Samuelson, P. A., Solow, R. M., "*Linear Programming and Economic Analysis,*" McGraw-Hill, New York (1958).

8. Ferguson, R. O., Sargent, L. F., *Linear Programming Fundamentals and Applications,* McGraw-Hill, New York (1958).

9. Gass, S. I., *Linear Programming Methods and Applications,* McGraw-Hill, New York (1958).

10. Bellman, R., "Programming and Lagrange Multipliers," *National Academy of Sciences,* v. 42, p. 767 (1956).

11. Saaty, T. L., *Mathematical Methods of Operations Research,* McGraw-Hill, New York (1959).

12. Sokolinkoff, I. S., and E. S., *Higher Mathematics for Engineers and Physicists,* (Second Ed.), McGraw-Hill, New York (1941).

13. Hildreth, Clifford, "A Quadratic Programming Procedure," *Naval Research Logistics Quarterly*, v. 4, p. 79 (1957).

14. Mugele, R. A., "A Program for Optimal Control of Nonlinear Processes," IBM Control Systems Tech. Rept. 02.191, San Jose, Calif. (1962).

15. Bertram, J. E., Sararchik, P. E., "On Optimal Computer Control", *Proc. First International Congress on Automatic Control*, Moscow, 1960.

16. Kalman, R. E., Bertram, J. E., "General Synthesis Procedure for Computer Control of Single and Multi-loop Linear Systems," *Trans A. I. E. E.*, v. 77, pp. 602–609 (1958).

17. Kurzweil, F., "The Dynamic Control of Multi-input Multi-output Continuous Processes," IBM Control Systems Tech. Rept. 02-178 (Oct. 31, 1961).

BIBLIOGRAPHY

Amundson, N. R., American Institute of Chemical Engineers, Tenth Annual Institute Lecture, Cincinnati (December 1958).

Carroll, C. W., "The Created Response Surface Technique for Optimizing Nonlinear, Restrained Systems," *Operations Research*, March 1961.

Carter, I, P. V., "Optimization Techniques," *IBM Nordiska Laboratories*, TN 18.048 (1961).

Dorn, W. S., "Nonlinear Programming," Research report RC-266, IBM Corp., Yorktown Heights, N. Y., May 24, 1960.

Dorn, W. S., "On Lagrange Multipliers and Inequalities," *Operations Research*, v. 9, p. 95 (Jan. 1961).

Hsieh, George J. (Ed.) *Optimization Techniques in Chemical Engineering*, Symposium, New York University (May 18, 1960).

Lapidus, L., Shapiro, E., Shapiro, S., Stillman, R. E., "Optimization of Process Performance," *A. I. Ch. E. Journal*, v. 7, #2, p. 288 (June 1961).

Sandelien, J. F., "An Approach to Dynamic Optimizing," *Communications and Electronics* (I. A. E. E.) p. 1 (July 1960).