Kirby, C. E., and M. Van Winkle, "Vapor-Liquid Equilibria: 2,3-Dimethylbutane-Methanol and 2,3-Dimethylbutane-Methanol-Chloroform Systems," *J. Chem. Eng. Data,* **15**, 177 (1970).

Klaus, R. L., and H. C. Van Ness, "An Extension of the Spline Fit Technique and Applications to Thermodynamic Data," *AIChE J.*, **13**, 1132 (1967).

Koizumi, E., and S. Ouchi, "Vapor Pressure of Tetrahydrofuran and Some Thermodynamic Values," *Nippon Kagaku Zasshi,* **91**, 501 (1970).

Kunz, R. G., and R. S. Kapner, "Correlation of Second Virial Coefficients Through Potential Function Parameters," *AIChE J.*, **17**, 562 (1971).

McGlashan, M. L., and D. J. B. Potter, "An Apparatus for the Measurement of the Second Virial Coefficients of Vapours; The Second Virial Coefficients of Some *n*-Alkanes and Some Mixtures of *n*-Alkanes," *Proc. Roy. Soc.*, **267A**, 478 (1962).

McGlashan, M. L., and R. P. Rastogi, "The Thermodynamics of Associated Mixtures," *Trans. Faraday Soc.*, **54**, 496 (1958).

Mixon, F. O., B. Gumowski, and B. H. Carpenter, "Computation of Vapor-Liquid Equilibrium Data from Solution Vapor Pressure Measurements," *Ind. Eng. Chem. Fundamentals,* **4**, 455 (1965).

Moelwyn-Hughes, E. A., and R. W. Missen, "Thermodynamic Properties of Methyl Iodide and Chloromethane Solutions," *Trans. Faraday Soc.*, **53**, 607 (1957).

Osborn, A. G., and D. R. Douslin, "Vapor Pressure Relations of 13 Nitrogen Compounds Related to Petroleum," *J. Chem. Eng. Data*, **13**, 534 (1968).

Prausnitz, J. M., "Fugacities in High-Pressure Equilibria and in Rate Processes," *AIChE J.*, **5**, 3 (1959).

Scott, D. W., "Tetrahydrofuran: Vibrational Assignment, Chemical Thermodynamic Properties, and Vapor Pressure," *J. Chem. Thermodynamics,* **2**, 833 (1970).

Steinbrecher, M., and H.-J. Bittrich, "Das isotherme Flussigkeits-Dampf-Gleichgewicht der binären Systeme Zwischen den Komponenten Tetrachlormethan, "Äthylacetat und Dioxan," *Z. phys. Chem.* (*Leipzig*), **232**, 312 (1966).

Van Ness, H. C., *Classical Thermodynamics of Non-Electrolyte Solutions*, pp. 120, 139-143, Pergamon, Oxford (1964).

———., "Thermodynamic Excess Properties of Binary Liquid Mixtures," *Ind. Eng. Chem.*, **59**, 33 (Sept., 1967).

Manuscript received August 11, 1972; revision received September 22, 1972; paper accepted September 22, 1972.

A Feasible-Point Algorithm for Structured Design Systems in Chemical Engineering

Structured design systems are systems in which the equations representing the equality and inequality constraints are sparse and highly precedence orderable. An algorithm has been developed for such systems which is guaranteed, under certain assumptions, to arrive in a finite number of steps at a feasible point (that is, one which satisfies all the constraints) or to identify a subset of the constraints for which no feasible point can be found. The algorithm can be applied to a system with only inequality constraints or to a system with both equality and inequality constraints.

The algorithm uses an indirect approach. It hypothesizes that a subset of constraints has no feasible region and then attempts to verify this conjecture. If successful, the subset is identified as infeasible and obviously no feasible point exists. If unsuccessful, either a new hypothesis can be generated or the algorithm has indirectly found a feasible point.

Limited computational experience with the algorithm indicates that the number of steps required to find a feasible point for a system of constraints has been of the same order of magnitude as the total number of constraints in the system. For linear constraints, the efficiency of the algorithm has been comparable to phase one of the Simplex algorithm of Linear Programming.

CHARLES J. DEBROSSE
and
ARTHUR W. WESTERBERG

Department of Chemical Engineering
University of Florida
Gainesville, Florida 32601

SCOPE

A rather successful and common approach to locating a feasible point for small design problems by hand is to guess at such a point and determine which constraints (if any) are violated. If the violated set contains only a few structurally simple constraints (such as $T \leqq 300°C$), these are used with the design equation set to guess a new candidate feasible point. The process is repeated until hopefully a feasible point is reached. For a design system with a large number of constraints, the problem of finding a first feasible point may be too formidable to permit a trial-and-error approach. The algorithm presented here provides a practical approach to the feasible-point problem.

The equations representing the constraints for most practical problems in chemical engineering design are highly structured, that is, each equation involves few of the variables and therefore the set of equations is highly precedence orderable. For structured problems, the solution of a set of equations is considered a reasonable task provided the structure is used in developing the solution procedure. Currently systems are under development

Correspondence concerning this paper should be addressed to A. W. Westerberg.

which can automatically manipulate the equations and variables to derive the efficient solution procedure needed. One such system is GENDER, General ENgineering DEsign Routines, being developed at the University of Florida by one of the authors in the area of computer-aided design. Assuming the existence of such a system, a natural extension to its more obvious use in the generation of permanent (FORTRAN) programs for fixed sets of equations is to use such a system to modify the solution procedure as the problem is being solved. This strategy can, in principle, permit some numerical problems to be overcome as encountered, but perhaps equally important it can also permit the equations defining the problem to change as the solution proceeds with hopefully little if any added computational burden.

The existence of a system such as GENDER and the general success of the common approach to locating feasible points used by engineers for simple design problems motivated the development of the feasible-point algorithm of this paper.

SIGNIFICANCE AND CONCLUSIONS

The feasible-point algorithm presented in this paper was applied to the seven small examples indicated in Table 1 to ascertain its effectiveness. The smallest was a linear problem involving only two variables and four constraints. The largest was a nonlinear alkylation process optimization problem presented and solved by Bracken and McCormick (1968) in a book of sample nonlinear optimization problems. As indicated it involved ten variables, three equality constraints, and 31 inequality constraints. For this latter problem, the first feasible point supplied by Bracken and McCormick before optimizing was found by guessing it, a common approach.

In the examples tried, five of which are nonlinear, a first feasible point was found with remarkable efficiency when one existed. In all cases, the number of steps to such a point took fewer than the number of constraints in the problem. For the linear problems, the algorithm was as effective as Phase 1 of the Simplex algorithm of Linear Programming, the standard of the industry. In all cases no first guess was provided; it was obtained by the algorithm.

Extrapolating on these limited, but remarkably successful, experiments with the algorithm is obviously dangerous but tempting. It appears the ability to add and delete equations while proceeding with the solution of a highly structured optimization problem is an extremely useful concept. For linear problems this capability is readily implemented and is of course used in the Simplex algorithm, where the pivoting step is simply the trading of one equation for another.

When each function evaluation is the solution of the hundreds to thousands of equations defining a process flowsheet, any inefficiency in finding a feasible point is obviously costly. Also if a feasible point does not exist, any information as to which constraints are causing the problem is valuable to the design engineer. Thus the algorithm appears to have real use for this type of problem.

We shall consider the following general mathematical programming problem:

$$\underset{\mathbf{X}\,\epsilon\, E^{m'}}{\text{minimize}} \quad F(\mathbf{X}) \tag{A}$$

subject to

$$f_i(\mathbf{X}) = 0 \quad i = 1, \ldots, n' < m' \tag{1}$$

$$g_j(\mathbf{X}) \leqq 0 \quad j = 1, \ldots, p, \tag{2}$$

where any of the functions F, $\{f_i\}$, and $\{g_j\}$ may be nonlinear.

Most techniques applicable to the solution of Problem A require a feasible starting point, that is, the starting point must satisfy Equations (1) and (2) simultaneously. The usefulness of a systematic procedure for finding a feasible starting point when such a point is not immediately available is obvious. Since it may happen that a particular set of constraints is not feasible, a practical feasible-point algorithm must be able to detect an infeasible system.

If the objective function and Equations (1) and (2) in Problem A are all linear, Phase 1 of the Simplex algorithm (Gue and Thomas, 1968) proceeds automatically to a feasible point, if one exists. Otherwise, the algorithm yields information showing that the system is infeasible. More generally, the objective function or some of Equations (1) and (2) are nonlinear and the Simplex algorithm is not applicable. Fiacco and McCormick (1968) and Hadley (1964) describe methods for finding a feasible point in the nonlinear case.

The method of Fiacco and McCormick uses a penalty-function approach to locate a point in the interior of the region defined by Equations (2). At an infeasible point $\mathbf{X}$, the sets $S = \{s | g_s(\mathbf{X}) \geqq 0\}$ and $T = \{t | g_t(\mathbf{X}) < 0\}$ are defined. Then, by using appropriate penalty functions associated with sets S and T, a sequence of points is selected that decreases $\sum_{s \epsilon S} g_s(\mathbf{X})$ without violating any of the $g_t(\mathbf{X})$ which are already satisfied. Each time a new point strictly satisfies one of the constraints in the set S, the corresponding index is removed from S and placed in T. An interior point is attained when S is empty. The equality constraints in Equations (1) are then driven to zero by means of a separate penalty function.

Hadley suggests that a feasible point be located by the introduction of artificial variables which are then driven to zero by minimization of an appropriate objective function. This technique is simply an extension of Phase 1 of the Simplex algorithm to the nonlinear case.

A principal motivation for the development of the algorithm presented here is that the equations representing the constraints for most practical problems are highly structured, that is, each equation involves few of the variables in the problem and as such the set is highly precedence orderable. The algorithm involves at each step the simultaneous solution of a particular subset of the constraints in the system; this subset always includes the equality constraints $\{f_i\}$. For structured problems, the solution of a set of equations is considered a reasonable task provided the structure is used in developing the solution procedure. Practical application is dependent upon the development of a system which can automatically manipulate the equations and variables to derive the efficient solution procedures needed. One such system is GENDER, General ENgineering DEsign Routines, being developed at the University of Florida by one of the authors in the area of computer-aided design for engineering systems.

The algorithm uses an indirect approach to find a feasible point for Equations (1) and (2). At each step in the algorithm, we examine a particular subset of the constraints W for infeasibility. This is accomplished by evaluating a point on the boundary of every region determined by the constraints in W.

If every point evaluated according to this procedure violates one or more of the constraints in W, W is infeasible. If a point is found which is feasible with respect to W and all other constraints, we exit. If a point is found which satisfies W but violates one or more of the other constraints, a new set of constraints to be investigated for infeasibility is formed.

The algorithm is constructed so that a particular set of constraints is never investigated more than once. Under certain restrictions which will be discussed later, this means that the algorithm proceeds in a finite number of steps either to a feasible point or an indication that a particular subset of constraints is infeasible.

THEORETICAL BASIS OF A NEW FEASIBLE-POINT ALGORITHM

Notation

For the sake of brevity we may refer to a constraint by its index. Thus constraint j or $\{j\}$ should be understood as constraint g_j.

Preliminary Definitions

In order to establish the theoretical foundation of the algorithm, we shall need the definitions which follow.

Definition. Consider a system of p constraints $p \geqq 2$ in an n-dimensional space $n \geqq 2$. Each subset of $m \leqq p$ constraints, $g_i(X) \leqq 0$, $i = 1, \ldots, m$, whose limiting surfaces $g_i(X) = 0$ intersect simultaneously determines by this intersection at least one surface. Any surface determined by such a simultaneous intersection of m constraints whose dimensionality is greater than $n - m$ will be said to be a *redundant intersection.*

Definition. A surface determined by the simultaneous intersection of m constraints in an n-dimensional space whose dimensionality is less than $n - m$ will be said to be a *degenerate intersection.*

Definition. A set of m constraints which intersect simultaneously to form more than one surface will be said to have a *multiple intersection.*

A constraint itself may have more than one limiting surface, and this special case of a multiple intersection is called a *multiple constraint.*

Necessary and Sufficient Conditions for Infeasibility

As we mentioned in the introduction, the algorithm uses an indirect approach to find a feasible point. The following theorem furnishes the basis for this approach.

Theorem 1. Consider a system of p inequality constraints $g_i(X) \leqq 0$, $i = 1, \ldots, p$, $p \geqq 2$ in an n-dimensional space $n \geqq 2$. Each subset of $m \leqq p$ constraints $g_i(X) \leqq 0$, $i = 1, \ldots, m$ whose limiting surfaces $g_i(X) = 0$ intersect simultaneously determines by this intersection at least one surface. (In the case of a multiple intersection, more than one surface is determined. If the dimensionality of one of the surfaces is greater or less than $n - m$, a redundant or degenerate intersection exists, respectively.)

Suppose a point is evaluated on each surface determined by the simultaneous intersection of a subset of the p constraints such that the surface in question intersects no other of the p constraints. We shall define such a surface as a surface of maximal intersection. Then at least one point has been placed on the boundary of every region defined by the p constraints. Further, the system is infeasible if, and only if, none of the points evaluated is feasible with respect to all p constraints.

Proof. Every region determined by the p constraints is bounded by at least one constraint; the limiting case occurs when one of the p constraints does not intersect any of the remaining $p - 1$ constraints. In general, every region is bounded by more than one constraint, some of which intersect simultaneously. In particular, every region has on its boundary at least one surface determined by the intersection of j constraints $1 \leqq j \leqq p$ such that this surface intersects none of the remaining constraints.

To see this, we consider an arbitrary region R and imagine that we are located somewhere in its interior. The region must be bounded by the limiting surface of at least one constraint, so we proceed from our starting point to this surface and arrive at a point on the boundary of R. Now this surface may intersect the limiting surface of another constraint. If so, we move along the first surface until it intersects the second and arrive at a surface defined by the intersection of two constraints which is still on the boundary of R. (As long as we do not actually cross a limiting surface we cannot leave the boundary of R.) In turn, this last surface may intersect the limiting surface of a third constraint, and so on, until finally we arrive either at a surface which intersects none of the limiting surfaces of the remaining constraints or at a surface determined by the intersection of the limiting surfaces of all p constraints.

In either case, under the conditions of the hypothesis we have placed at least one point on the boundary of the arbitrary region R. That is, under the conditions of the hypothesis we have placed at least one point on the boundary of every region, including the feasible region, if one exists. Thus, the system is infeasible if and only if all the points evaluated are infeasible.

Upper and Lower Bounds on the Number of Constraints in a Minimal Infeasible Set

Theorem 1 furnishes a necessary and sufficient condition for the infeasibility of a set of constraints. We now consider the magnitude of the task of proving that a set of constraints is infeasible. We seek to put an upper bound on the number of constraints in a given system which must be investigated to prove that the system is infeasible. Since any number of constraints may be added to an infeasible set of constraints without changing the infeasibility of the system, we need to define the concept of a minimal infeasible set.

Definition. A set of constraints H is a minimal infeasible constraint set if H is infeasible but every proper subset of H is feasible.

We remark at this point that the algorithm works in such a way that if a system of constraints is infeasible, the algorithm identifies a minimal infeasible constraint set.

It is obvious that for a space of arbitrary dimension as few as two constraints may be sufficient to determine an infeasible system. If two inequality constraints do not intersect and point away from each other, that is, the two constraints have nonoverlapping feasible regions, any set of constraints containing these two is infeasible. Since any constraint is feasible by itself, two is the smallest possible number of constraints in a minimal infeasible set.

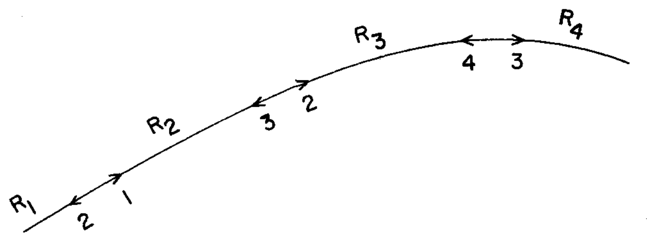

Fig. 1. A minimal infeasible set of four constraints on a line. (Note that constraints 2 and 3 have multiple intersections.)

If a finite system of constraints in an n-dimensional space satisfies either of the following two conditions, we can also place an upper limit of $n + 1$ on the number of constraints in a minimal infeasible set for that system:

a) All constraints in the system are convex;

b) There are no multiple intersections or multiple constraints in the system.

Rockafellar (1970) proves that condition (a), a version of Helley's Theorem, is sufficient to establish an upper limit of $n + 1$ on the number of constraints in a minimal infeasible set in an n-dimensional space. In Theorem 2 we shall prove that condition (b) is likewise sufficient. Theorem 2 will be preceded by three lemmas required for its proof.

Lemma 1. Consider a set M of m constraints and a feasible proper subset P of p constraints. Suppose that the feasible region of P is bounded by all p constraints in P (requires all p constraints for its definition), is connected, and is intersected by none of the $m - p$ constraints in $M - P$. Then either M is feasible or there exists at least one constraint $i \in M - P$ such that $P \cup \{i\}$ is infeasible.

Proof. By hypothesis none of the constraints in $M - P$ intersects the feasible region of P. Therefore either all $m - p$ constraints in $M - P$ have at least one surface whose feasible region wholly includes the feasible region of P, or there exists a constraint $i \in M - P$ such that all (one or more) surfaces defined by constraint i have feasible regions which wholly exclude the feasible region of P. If the former is true, M is feasible. If the latter holds, $P \cup \{i\}$ is infeasible.

Lemma 2. If a set of p $(p > 2)$ constraints on a line is a minimal infeasible set, there are at least $p - 2$ multiple constraints on the line.

Proof. Since the set of p constraints are minimally infeasible, each subset of $p - 1$ constraints must have a feasible region along the line. For that region, only the remaining constraint g_k is violated and we can label that region R_k. We can also renumber the constraints so the regions R_k are encountered in order, starting with R_1 proceeding to R_p as we move along the line in a fixed direction. If R_k appears more than once, only its initial encounter need be used in the renumbering.

While proceeding from region R_{k-1} to R_k to R_{k+1}, $2 \leq k \leq p - 1$, it is clear that constraint g_k must be crossed twice as it is satisfied in both regions R_{k-1} and R_{k+1} but not satisfied in R_k. Thus all $p - 2$ constraints g_k, $k = 2, \ldots, p - 1$, must be crossed at least twice.

Figure 1 illustrates this lemma for a simple case when $p = 4$ constraints.

Lemma 3. Consider a minimal infeasible set P of p constraints with no multiple intersections or multiple constraints. Then: (a) there is at least one set $P_j \equiv P - \{j\}$ such that the feasible region of P_j is bounded by all $p - 1$ constraints in P_j; (b) the feasible region of P_j is connected; and (c) the feasible region of P_j is not intersected by constraint j.

Proof. By hypothesis there are no multiple intersections or multiple constraints, and thus the feasible region for any subset of P is connected. Thus (b) is proved.

To prove (a) and (c), consider the set Q, $Q \subseteq P$, of q constraints with the following properties: (i) the feasible region of Q is bounded by all q constraints in Q, and (ii) the feasible region of Q is intersected by none of the constraints in $P - Q$.

It is obvious that at least one such set exists for any set P of p constraints with no multiple intersections or multiple constraints. Further, under the conditions of the hypothesis, the number of constraints $q < p$, since $q = p$ would imply that P is feasible, which contradicts the statement of the theorem.

Suppose $q < p - 1$. Then by Lemma 1 either P is feasible or there exists a constraint $i \in P - Q$ such that the set of $q + 1$ (where $q + 1 < p$) constraints $Q \cup \{i\}$ is infeasible. This also contradicts the statement of the theorem, and we therefore know that $q \geq p - 1$.

Since we have proved that $q \geq p - 1$ and $q < p$, we have proved $q = p - 1$. Then properties (i) and (ii) of Q prove (a) and (c).

Theorem 2. If a set W of $n + p$ $(p > 1)$ constraints in an n-dimensional space is a minimal infeasible set, there is at least one multiple intersection or multiple constraint among the constraints in W.

Proof. Consider an infeasible system W of $n + p$ $(p > 1)$ constraints in an n-dimensional space $(n > 1)$ such that W is a minimal infeasible constraint set. According to the theorem, there must be at least one multiple intersection or multiple constraint among the constraints in W. We shall prove this by contradiction. Figure 2 should aid in following this proof.

Assume that there are no multiple intersections or multiple constraints in W. Then by lemma 3 there must be a set of $n + p - 1$ constraints $W_j \equiv W - \{j\}$ such that the feasible region of W_j is bounded by all $n + p - 1$ constraints in W_j and is not intersected by constraint j.

Consider the $(n - 1)$-dimensional surface S_1 defined by constraint j. Since W is infeasible, we know that on S_1 the system of $n + p - 1$ constraints W_j is infeasible. We will now show that any set of $n + p - 2$ of these constraints is feasible, and thus W_j is a minimal infeasible set on S_1.

Consider a set of $n + p - 2$ constraints $W_{ij} \equiv W -$

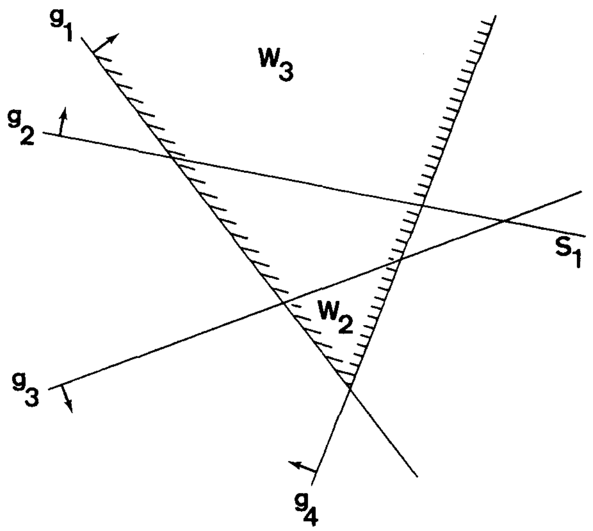

Fig. 2. Regions W_i, W_j, and W_{ij} and Surface S_1 in Theorem 2 for a 2-dimensional space. Here $i = 3$ and $j = 2$. Note W_2 is bounded by all three constraints g_1, g_3, and g_4 and is not intersected by g_2. Region W_{23} is the indicated V-shape feasible region of constraints g_1 and g_4.

$\{i, j\}$, for any i, $i \neq j$. Clearly the feasible region for W_j lies entirely in the infeasible region of constraint j. Also clearly the feasible region of W_i lies entirely in the feasible region of constraint j (and is therefore disjoint from the feasible region of W_j), since W_i includes constraint j. Since W_{ij} is a proper subset of both W_i and W_j, the feasible region of W_{ij} includes the feasible regions of both W_i and W_j. By Lemma 3 the feasible region of W_{ij} is connected; it must therefore intersect S_1, which separates the feasible and infeasible regions of constraint j. Thus, we have shown that there is a feasible region of every W_{ij}, $i \neq j$, on surface S_1 defined by W_j.

Therefore by Lemma 3 there must exist a set of $n + p - 2$ constraints $W_{jk} \equiv W - \{j, k\}$ such that the feasible region of W_{jk} on S_1 is bounded by all $n + p - 2$ constraints in W_{jk} and is not intersected by constraint k.

Applying Lemma 3 $n - 3$ more times, we conclude that there must exist a one-dimensional surface S_{n-1} defined by the intersection of $n - 1$ constraints in W along which the remaining $p + 1$ constraints in W form a minimal infeasible set. But then by Lemma 2 there must be at least $p - 1$ multiple constraints along S_{n-1}, which in turn implies that there are multiple intersections or multiple constraints in W. We have thus arrived at a contradiction of what we assumed at the beginning of the proof.

Thus, our assumption that there are no multiple intersections or multiple constraints in W leads to a contradiction, and we conclude that under the conditions of the hypothesis there must be at least one multiple intersection or multiple constraint in W.

The fact that condition (b) is sufficient to establish an upper bound of $n + 1$ on the number of constraints in a minimal infeasible set in an n-dimensional space follows directly from Theorem 2.

It should be noted that neither condition (a) nor condition (b) implies the other. A convex set of constraints may have any number of multiple intersections or multiple constraints. Conversely, if a system has no multiple intersections or multiple constraints there is no guarantee that every constraint in the system is convex or can be made convex by a deformation of the space E^n. From the point of view of our algorithm, condition (b) is more useful than condition (a) since multiple intersections necessitate additional point evaluations in order to establish infeasibility according to the criterion of Theorem 1.

Multiple Intersections and Cycles

As we stated before, at each step in the algorithm we hypothesize that a subset H_i of the system of constraints of interest is infeasible and attempt to prove the hypothesis according to the criterion of Theorem 1. Under ordinary circumstances, information obtained at a single point is sufficient either to tell us to retain the current hypothesis or to enable us to make a new hypothesis which has not been disproved previously. It may happen, however, that a point evaluation by itself would result in the repetition of a previously disproved hypothesis.

Definition. A sequence of steps in the algorithm in which a hypothesis H_i, disproved at point X_l, is regenerated at X_{l+j} ($j > 1$) is called a cycle. The cycle is said to extend from X_l to X_{l+j}. The term cycle still applies even if alternate hypotheses exist at one or more points within the cycle.

From what we have seen so far, it is obvious that multiple intersections can complicate the implementation of the algorithm considerably. If multiple intersections are permitted, we could have an arbitrarily large minimal infeasible set in any dimensional space. Furthermore, a finite set of constraints could have an infinite number of intersections, requiring an infinite number of point evaluations to check every region defined by the constraints. Although we have been unable to prove that every cycle is a result of multiple intersections, it is certainly true that multiple intersections can cause cycles. Perhaps most important of all, the existence of multiple intersections which we have failed to detect could cause us to conclude erroneously that a set of constraints is infeasible.

Minimum Number of Point Evaluations to Establish Infeasibility

The following theorem, which follows directly from Theorem 1, enables us to construct an algorithm which will establish the infeasibility of a set of constraints with a minimum number of point evaluations.

Theorem 3. Consider a set of p constraints P with no multiple intersections. Suppose we wish to determine whether P is infeasible. Then a point must be evaluated on the surface S determined by the simultaneous intersection of all constraints in M, an m-subset of P, if and only if all $p - m$ of the $(m + 1)$-subsets of P which contain M have no simultaneous intersections.

Proof. If all $p - m$ of the $(m + 1)$-subsets of P which contain M have no simultaneous intersections, then S intersects none of the other constraints in P. Therefore by Theorem 1 a point on S must be evaluated to establish the infeasibility of P.

On the other hand, if one of the $(m + 1)$-subsets of P which contain M has a simultaneous intersection, S intersects another constraint in P. Theorem 1 then states that a point on S need not be evaluated.

A FEASIBLE-POINT ALGORITHM FOR STRUCTURED DESIGN SYSTEMS

The n-Dimensional Feasible-Point Algorithm

We are now ready to present the feasible-point algorithm and give some illustrative examples. The algorithm given will be for a system of inequality constraints only. In general, equality constraints can be viewed as merely reducing the number of degrees of freedom in a system. With only slight modifications, therefore, the algorithm can be applied to a system which includes both equality and inequality constraints. These modifications will be presented later.

At a typical step in the algorithm we shall say that $m \leqq n$ equations are to be incorporated simultaneously in an n-dimensional space. This means that we must specify $n - m$ components of X and calculate the remaining m

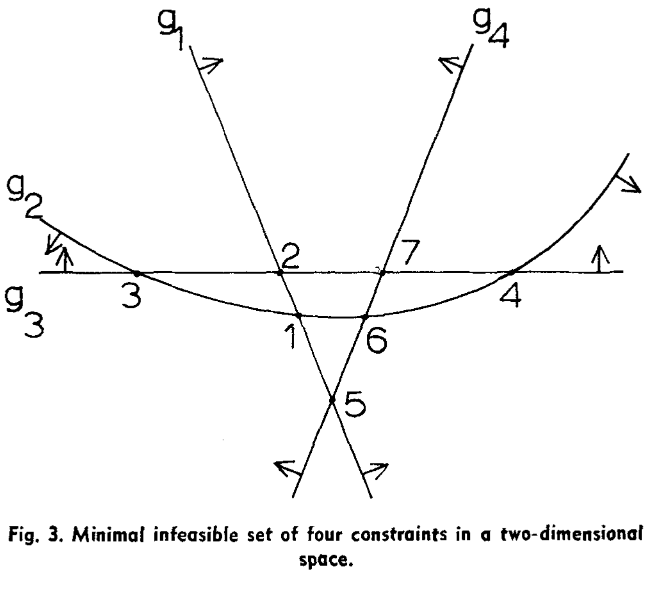

Fig. 3. Minimal infeasible set of four constraints in a two-dimensional space.

components as outputs of a set of equations

$$g_i(\mathbf{X}) = 0 \quad i = 1, \ldots, m \tag{3}$$

We shall assume that the system of constraints for which a feasible point is sought is highly structured, that is, most of the equations contain relatively few variables. This assumption is very important because the efficiency of the algorithm depends upon it.

The term *structurally infeasible set* will appear in the algorithm. We shall define a structurally infeasible set of equations as a set of equations which cannot be incorporated simultaneously because the total number of variables in a subset of the equations is smaller than the number of equations in the same subset. The following three equations in two unknowns furnish an example of a structurally infeasible set:

$$x_1 + x_2 - 5 = 0 \tag{4}$$

$$2x_1 - x_2 - 3 = 0 \tag{5}$$

$$x_2 - 1 = 0 \tag{6}$$

We shall say that an attempt to incorporate Equations (3) simultaneously results in no solution when, no matter which $n - m$ components of X we choose as decision variables and what values we assign to these decision variables, Equations (3) cannot be satisfied simultaneously for any values of the other m components of X. A no solution which results from a structurally infeasible subset of Equations (3) is fairly easy to detect (Lee et al., 1966); for example, $n + 1$ equations in n variables cannot be satisfied simultaneously unless one of the equations is redundant. However, the determination in general that a set of equations has no solution by numerical methods alone is admittedly difficult.

The algorithm which follows is guaranteed to find a feasible point, if one exists, for a system of inequality constraints provided all multiple intersections, if any, are detected. If the system of constraints is infeasible, a minimal infeasible subset of the constraints is identified.

The algorithm is written basically for a system with no multiple intersections because without this assumption a general algorithm would degenerate into an exhaustive search of every constraint surface. However, the algorithm can be used even if multiple intersections are present, provided the multiple intersections are detected by the user. Frequently, the nature of the constraints themselves can indicate the existence of such intersections.

We present the algorithm with a subroutine format as this represents a probable form for its implementation by anyone programming it on a computer. Four routines and a main algorithm, which coordinates the use of these routines, are specified. We present the main algorithm first, followed by the needed routines.

Main Algorithm

A. We first require an initial hypothesis. Find a set of constraints which can be incorporated, resulting in the point X. Also determine which constraints are violated at X. Then call Routine I to find an initial hypothesis H_1. Let $j = 1$.

B. Investigate H_j, which is now a set of p constraints which cannot be incorporated simultaneously. If a subset of H_j containing $m < p$ constraints is structurally infeasible, make a new hypothesis H_{j+1} containing only these constraints. Let j be set to $j + 1$, p be set to m and repeat this step.

C. Test hypothesis H_j using Routine II.
   1. If H_j is disproved, go to step E.
   2. Otherwise go to step D.

D. H is proved. Call Routine III to find a minimal infeasible constraint set and *exit feasible point algorithm*.

E. H_j is disproved. Test the point which disproved H_j.
   1. The point is feasible. Exit algorithm with a feasible point.
   2. The point is not feasible. Go to step F.

F. Call Routine I to generate a new hypothesis H_{j+1}.
   1. If H_{j+1} has been generated and disproved before, a cycle has been found. Go to step G.
   2. Otherwise let j be set to $j + 1$ and repeat from step B.

G. Call Routine IV to generate a new hypothesis H_{j+1}, let j be set to $j + 1$, and repeat from step B.

Routine I—Generating a New Standard Hypothesis

Entry is with a point X at which a known set of constraints is incorporated and another known set is violated.

Exit is either from the entire feasible point algorithm with a feasible point X or back to the caller with a hypothesis H containing $p \leq n + 1$ constraints which cannot all be incorporated simultaneously.

A. Create a hypothesis H which contains all constraints incorporated at X plus one or more of those from the set violated at X. (Constructing H in this manner precludes having point X reject the hypothesis.)

B. If H contains $n + 1$ constraints return to the caller with H as the new hypothesis.

C. Attempt to incorporate all the constraints in H, yielding a new point X.
   1. If the attempt fails, return to the caller with H as the new hypothesis.
   2. Find the set of constraints violated at X. If none are violated, a feasible point has been found. Exit entire algorithm. Otherwise repeat from step A.

Routine II—Testing the Current Hypothesis

Entry is with the current hypothesis H, the number of constraints p in H, and the dimension of the search space n.

On exit, H will have been proved or disproved. In the former case, all constraint subsets needed to isolate a minimum infeasible constraint set will have been created on lists L_1 to L_p. In the latter case, the point X which disproves H and the set of constraints incorporated at X will be available.

A. If $p > n$, let $r = n$. Otherwise let $r = p - 1$,

B. Let $l = 1$ and establish an empty list L_l.

C. Put onto list L_l all constraint subsets possible of size r among the constraints in H. Label them $E_r(k)$, $k = 1, 2, \ldots, p!/r!(p - r)!$.

D. For each entry $E_r(k)$ on list L_l, proceed as follows.
   1. Attempt to incorporate the constraints in $E_r(k)$.
      a. If unsuccessful go to step 4.
      b. Otherwise continue.
   2. Delete $E_r(k)$ from list L_l.
   3. If one of the points, X, found (there will be only one if no multiple intersections exist) does not violate at least one of the constraints in H not incorporated at the point, H is disproved. Return to caller with the set $E_r(k)$, the point X, and indicate H disproved. Otherwise continue.
   4. Repeat from step 1 with the next constraint subset.

E. When all constraint subsets of size r have been tested and H has not been disproved, we must then check for any surfaces of maximal intersection we might have missed because of the subsets of constraints $E_r(k)$ leading to no solution in the previous step. Each is listed on L_l. Theorem 3 will help us find the minimum number of additional points we have to evaluate to guarantee we do not miss any surfaces of maximal intersection. Remember Theorem 3 assumes no multiple intersections exist.

Generate entries for list L_{l+1} as follows:

1. If fewer than $p - r + 1$ entries remain on list L_l, H has been proved. Return to caller.
2. Form all the set intersections possible for the entries on list L_l taken $p - r + 1$ at a time. In this way, the common constraints are found among each group of $p - r + 1$ entries on list L_l. Place each resulting set intersection containing exactly $r - 1$ constraints onto list L_{l+1}.
3. Increment l by one and decrement r by one.
4. Repeat from step D.

Routine III—Detecting a Minimal Infeasible Set of Constraints

Entry is with the constraint H proved infeasible, the lists L_1 to L_q of constraint subsets of H found to have no solution, and all points **X** evaluated in proving H.

Exit is with a minimal infeasible constraint set.

A. Form a master list of subsets of H_j having no solution, starting with the entries on the last list L_q formed, then L_{q-1}, up to L_1, and finally H itself.

B. Proceed in order through the above list, checking each against all the points evaluated in testing H. The first constraint set which cannot be shown feasible by one of these points is minimally infeasible.

C. Return with a minimal infeasible constraint set.

Routine IV—Generating a New Hypothesis after a Cycle is Detected

Entry is with all constraint sets incorporated and resulting points **X** found during the cycle as well as all hypotheses disproved.

Exit is with either a new standard hypothesis H containing $p \leq n + 1$ constraints or with a nonstandard hypothesis H containing $p > n + 1$ constraints because of the cycle to the caller. If a feasible point is found, exit is made from the entire feasible point algorithm (via Routine I) with the feasible point.

A. Examine in turn each point **X** found in the cycle. Apply Routine I to find a new standard hypothesis not already disproved by adding different constraints than those added previously to enlarge H from among the violated set at **X**. If a new standard hypothesis can be formed, return to the caller.

B. Start with the first hypothesis in the cycle. Take the union of successive hypotheses until a hypothesis H is formed which has not been disproved before.

C. If H has $n + 1$ or more constraints in it, return to caller with H.

D. Otherwise H has n or fewer constraints in it. Attempt to incorporate all the constraints in H.

1. If unsuccessful, return to caller with H.
2. If successful, call Routine I with the point **X** to obtain the new hypothesis. Return to caller with the resulting hypothesis.

Modifications of the Feasible-Point Algorithm to Include Equality Constraints

Suppose we wish to find a feasible point in an n-dimensional space for a system which includes m equality constraints in addition to inequality constraints. The number of degrees of freedom in the system is then $n - m$. The algorithm for this situation is then nearly identical to the algorithm for a system of only inequality constraints in an $(n - m)$-dimensional space. The m equality constraints are incorporated at every step in addition to those inequality constraints which are incorporated according to the rules of the algorithm for inequality constraints.

The following modifications are necessary, however, in order for the algorithm to apply generally to mixed systems, that is, systems which include both inequality and equality constraints.

1. Mixed systems which include only one inequality constraint may be infeasible, although this is obviously not possible for a system of inequality constraints alone. Suppose for a mixed system that a subset of the constraints H which includes only one inequality constraint is hypothesized to be infeasible. Then the hypothesis is investigated by evaluating a point on the surface determined by the simultaneous intersection of the equality constraints. If the point violates the inequality constraint in question, H is infeasible.
2. For inequality-constrained systems, Routine II can go no further than the evaluation of points on surfaces determined by a subset of the inequality constraints, with the constraints taken one at a time. In the case of mixed systems, this routine may require that a point **X** be evaluated on the surface determined by the equality constraints only. This will occur if an attempt to incorporate a single inequality constraint results in no solution.

Illustrative Examples

We now consider three examples to illustrate the mechanics of the feasible-point algorithm. The first is a hypothetical example designed to show how the algorithm identifies a minimal infeasible set. The next one is a real example which illustrates how multiple intersections are handled. The third example shows how a cycle is handled by the algorithm. Algorithm steps are indicated with M referring to steps in the Main Algorithm. Also in column H, new hypotheses are indicated as generated. In column E, the constraints one is attempting to incorporate are listed, and in column X the resulting point is given or the symbol N.S. which stands for no solution. Column V gives the constraints violated at **X**.

TABLE 1. FEASIBLE-POINT ALGORITHM PERFORMANCE DATA

Problem number	Number variables	Number eq. constraints	Number ineq. constraints	Problem description	Number of steps to feasible point
1	3	0	6	Nonlinear constraints	5
2*	3	0	6	Nonlinear constraints	7
3	4	0	12	Linear constraints	13
4‡	2	0	4	Nonlinear constraints with cycle	7***
5**	10	3	28	Alkylation process model (nonlinear)	8
6	2	0	4	Linear constraints	2
7‡‡	4	0	7	Nonlinear constraints	3

* Example 2 of this paper.
‡ Example 3 of this paper.
** Problem statement in Bracken and McCormick (1968).
‡‡ Problem statement on p. 100, Bracken and McCormick.
*** Number of steps to prove infeasibility.

Example 1 ($n = 3$-dimensional space)

This example refers to no real problem so the responses at each step are chosen *only* to illustrate various steps in the algorithm.

Step	H	E	X	V
M.A.		1,2,3	X_1	4,5
I.A.	H_1:1,2,3,4			
M.C.				
II.D.1		1,2,3	X_1 (again)	4,5
II.D.1		1,2,4	X_2	5,6
	Point X_2 disproves H_1.			
M.F.				
I.A.	H_2:1,2,4,5			
M.C.				
II.D.1		1,2,4	X_2 (again)	5,6
II.D.1		1,2,5	X_3	4,7
II.D.1		1,4,5	N.S.	
II.D.1		2,4,5	N.S.	
II.E	List L_1 contains 1,4,5 and 2,4,5. After step II.E list L_2 contains $\{1,4,5\} \cap \{2,4,5\} = \{4,5\}$.			
II.D.1		4,5	X_4	7,8
M.F.				
I.A.	H:4,5,7			
I.C.		4,5,7	X_5	8
I.A.	H_3:4,5,7,8			
M.C.				
II.D.1		4,5,7	X_5 (again)	8
II.D.1		4,5,8	X_6	1,7
II.D.1		4,7,8	X_7	5
II.D.1		5,7,8	X_8	2,4
	H_3 is proved infeasible.			
M.D.				
III	The master list contains only H_3, which is therefore minimally infeasible. Exit algorithm.			

Example 2 ($n = 3$ dimensional space)

Find a feasible point for the following system of inequality constraints.

1.	$-x_1 \leqq 0$	4.	$x_1 - x_3^2 \leqq -16$
2.	$-x_2 \leqq 0$	5.	$-x_1 + x_2 - 2x_3 \leqq -20$
3.	$-x_3 \leqq 0$	6.	$-x_1^2 + x_2^2 - x_3^2 \leqq -36$

Step	H	E	X	V
M.A.		1,2,3	X_1:0,0,0	4,5,6
I.A.	H_1:1,2,3,4			
M.B.	Constraints 1,3,4 of H_1 are structurally infeasible. H_2:1,3,4			
M.C.				
II.D.1		1,3	X_1:0,0,0 (again)	4,5,6
		1,4	X_2:0,0,−4	3,5,6
		1,4	X_3:0,0,4	5,6
	The intersection of constraints 1 and 4 determines two surfaces. Point X_3 on one of these surfaces disproves H_2.			
M.F.				
I.A.	H_3:1,4,5,6 (We choose the option of enlarging the hypothesis directly to 4 constraints in step I.A.)			
M.C.				
II.D.1		1,4,5	X_4:0,−12,4	2,6
II.D.1		1,4,5	X_5:0,−28,−4	2,3,6
II.D.1		1,4,6	N.S.	
II.D.1		1,5,6	X_6:0,18.04,19.02	ϕ
		(1,5,6	X_7:0,−5 72.7.64	2)
	We note the set 1,5,6 has two solutions, but we only consider the first.			
M.E.	X_6 is feasible. We exit entire algorithm with a feasible point.			

Example 3 ($n = 2$ dimensional space)

Find a feasible point for the system of inequality constraints depicted in Figure 3.

Step	H	E	X	V
M.A.		1,2	X_1	3
I.A.	H_1:1,2,3			
M.C.				
II.D.1		1,2	X_1 (again)	3
II.D.1		1,3	X_2	2
II.D.1		2,3	X_3	1
II.D.1		2,3	X_4	4
	Point X_4 disproves H_1.			
M.F.				
I.A.	H_2:2,3,4			
M.C.				
II.D.1		2,3	X_3 (again)	1
	Point X_3, a point already evaluated earlier, disproves H_2.			
M.F.				
I.A.	H_3:1,2,3 We find H_3, but H_3 is the same as H_1 and we have a cycle. We cancel H_3 as a hypothesis. The points X_1 to X_4 are in our cycle.			
M.G.				
IV.A	No alternate standard hypothesis can be generated.			
IV.B	H_3:1,2,3,4			
M.C.				
II.D.1	We will list only those points not already found before (X_1 to X_4 would appear here too otherwise).			
		1,4	X_5	3
II.D.1		2,4	X_6	3
II.D.1		3,4	X_7	2
II.E.1	H_3 is proved. Return to caller.			
M.D.				
III.A	The master list contains only the hypothesis H_3.			
III.C	Return with H_3 being the minimum infeasible constraint set. Exit feasible point algorithm.			

NOTATION

E_i	=	set of equations incorporated at $\mathbf{X}_i$
E^n	=	Euclidean n-dimensional space
f	=	equation or equality constraint, may be subscripted
F	=	objective function
g	=	inequality constraint, may be subscripted
H	=	subset of constraints hypothesized to be infeasible, may be subscripted
ϕ	=	Null set
V_i	=	set of constraints violated at $\mathbf{X}_i$
$\mathbf{X}$	=	point in E^n, may be subscripted

Mathematical Symbols

ϵ	=	(an) element(s) of
$\subset$	=	proper subset of
$\subseteq$	=	subset of
$\cup$	=	union with
$\sim$	=	negation of
$\mid$	=	such that
$\cap$	=	set intersection with

LITERATURE CITED

Bracken, J., and G. P. McCormick, *Selected Applications of Nonlinear Programming*, Wiley, New York (1968).

Fiacco, A. V., and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* Wiley, New York (1968).

Gue, R. L., and M. E. Thomas, *Mathematical Methods in Operations Research.* Macmillan, New York (1968).

Hadley, G., *Nonlinear and Dynamic Programming*, Addison-Wesley, Reading, Mass. (1964).

Lee, W., J. H. Christensen, and D. F. Rudd, "Design Variable Selection to Simplify Process Calculations," *AIChE J.*, **12**, p. 1104 (1966).

Rockafellar, R. T., *Convex Analysis*, Princeton Univ. Press, N. J. (1970).

Manuscript received March 9, 1972; revision received September 14, 1972; paper accepted September 15, 1972.