B	= penetration parameter in Equation (17), dimensionless
f	= Fanning friction factor defined by Equation (8), dimensionless
g_c	= conversion factor = 32.2 (lb. mass) (ft.)/(lb. force) (sec.^{2})
k	= universal constant in Equation (17), dimensionless
L	= Prandtl mixing length, ft.
N_{Re}	= Reynolds number defined by Equation (7), dimensionless
r	= radius to an arbitrary point in the fluid, ft.
r_m	= radius of maximum velocity, ft.
r_0	= radius of tube, ft.
R_H	= hydraulic radius for the portion of the stream between r_m and r, ft.
R_{H_s}	= hydraulic radius for the portion of the stream between r_m and the boundary s, ft.
u	= local fluid velocity at radius r, ft./sec.
u_m	= maximum local velocity of fluid, ft./sec.
V	= bulk average linear velocity of fluid, ft./sec.
y	= normal distance from wall of conduit, ft.

Greek Letters

ϵ	= local eddy viscosity at radius r, lb. mass/(sec.) (ft.)
ϵ^0	= velocity-mean eddy viscosity between the boundary s and radius r, lb. mass/(sec.) (ft.)
$\epsilon_m{}^0$	= velocity-mean eddy viscosity between the boundary s and the radius of maximum velocity, lb. mass/(sec.) (ft.)
η_s	= criterion parameter defined by Equation (10), dimensionless
μ	= fluid viscosity, lb. mass/(sec.) (ft.)
ρ	= fluid density, cu. ft./lb. mass
τ	= local shearing stress, lb. force/sq. ft.
τ_s	= skin friction at boundary s, lb. force/sq. ft.
ϕ	= damping factor in Equation (17), dimensionless
Φ	= distance parameter defined in Equation (5), dimensionless
Φ_m	= distance parameter defined in Equation (6), dimensionless

Subscripts

1	= inner boundary of annulus or portion of stream between r_m and r_1
2	= outer boundary of annulus or portion of stream between r_m and r_2
i	= equivalent tube t or p, whichever is pertinent
L	= laminar flow
m	= maximum velocity or its position
p	= tube equivalent to inner portion of annulus between r_m and r_1
s	= boundary s or portion of stream between r_m and r_s
t	= tube equivalent to outer portion of annulus between r_m and r_2
c	= critical value at onset of disturbance eddies

LITERATURE CITED

1. Croop, E. J., Ph.D. thesis, Carnegie Inst. Technol., Pittsburgh, Pa. (1958).
2. ———, and R. R. Rothfus, *AIChE J.*, **8**, 26 (1962).
3. Gill, W. N., and Marvin Scher, *ibid.*, **7**, 61 (1961).
4. Hanks, R. W., *ibid.*, **9**, 45 (1963).
5. ———, paper presented at Am. Inst. Chem. Engrs., 60 Annual Meeting, New York (Nov., 1967).
6. Lamb, Horace, "Hydrodynamics," 6 ed., p. 586, Dover, New York (1945).
7. Rothfus, R. R., and C. C. Monrad, *Ind. Eng. Chem.*, **47**, 1144 (1955).
8. Rothfus, R. R., W. K. Sartory, and R. I. Kermode, *AIChE J.*, **12**, 1086 (1966).
9. Rothfus, R. R., J. E. Walker, and G. A. Whan, *ibid.*, **4**, 240 (1958).
10. Sartory, W. K., Ph.D. thesis, Carnegie Inst. Technol., Pittsburgh, Pa. (1962).
11. Senecal, V. E., Ph.D. thesis, Carnegie Inst. Technol., Pittsburgh, Pa. (1951).
12. ———, and R. R. Rothfus, *Chem. Eng. Progr.*, **49**, 533 (1953).
13. Walker, J. E., Ph.D. thesis, Carnegie Inst. Technol., Pittsburgh, Pa. (1957).
14. Whan, G. A., Ph.D. thesis, Carnegie Inst. Technol., Pittsburgh, Pa. (1956).
15. ———, and R. R. Rothfus, *AIChE J.*, **5**, 204 (1959).
16. Van Driest, E. R., *J. Aeronaut. Sci.*, **23**, 1007 (1956).

Manuscript received April 18, 1968; revision received July 28, 1968; paper accepted July 31, 1968.

The Structuring of Process Optimization

JAMES H. CHRISTENSEN

University of Wisconsin, Madison, Wisconsin

The efficiency of process optimization by mathematical programming can be increased by tearing, that is, rearranging the design equations so as to reduce the number of equality constraints. The structure of a system of equations may be depicted as an undirected bipartite graph; algorithm I-T utilizes this graph to determine an order of solution for the equations which requires no tears. If this is impossible, then algorithm II-T uses indexing in conjunction with algorithm I-T to produce an order which minimizes the number of torn equations. This procedure is extended to the problem of minimum recycle parameters, and the two-way interaction between tearing and algebraic simplification is illustrated.

THE STRUCTURING OF PROCESS OPTIMIZATION

The problem of optimizing a process design may be stated in general algebraic form as:
Maximize the objective function

$$z(x_1, x_2, \dots, x_n) = z(\mathbf{x}) \qquad (1)$$

subject to the equality constraints

$$f_i(x_1, x_2, \dots, x_n) = 0, \quad i = 1, 2, \dots, m$$

or

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \qquad (2)$$

James H. Christensen is at the University of Oklahoma, Norman, Oklahoma.

in vector form. These equations may represent material or energy balances, kinetic equations, etc., and are often referred to as the *design equations*. In addition, the vector **x** of process variables may have to satisfy inequality constraints of the form

$$h_i(\mathbf{x}) \geqq 0, \quad i = 1, 2, \ldots, m_2; \text{ equivalently, } \mathbf{h}(\mathbf{x}) \geqq \mathbf{0} \tag{3}$$

These may be, for instance, limitations on the maximum temperatures attainable, availability of raw materials, physical impossibility of negative concentrations, flow rates, etc.

DiBella and Stevens (*1*) present a good example of this way of formulating the problem in the optimization of the steady state material balances of a small chemical process originally presented by Williams and Otto (*2*) as an application of computer control. DiBella and Stevens describe the system in terms of nine design equations in twelve variables; however, the introduction of five additional equations (f_{10} through f_{14}) and variables (ϕ, t, R_1, R_2, and R_3) greatly simplifies the form of the equality constraints, as follows:

(4)

Equality constraints:

Overall material balance:

$$f_1 = F_A + F_B - F_G - F_D - F_P = 0$$

Azeotropic separation in column:

$$f_2 = F_{RP} - 0.1F_{RE} - F_P = 0$$

Overall balance on component E:

$$f_3 = F_{RE}\phi - (M_E/M_B)R_2 = 0$$

Overall balance on component P:

$$f_4 = R_2 - (M_P/M_C)R_3 - (F_{RP} - F_p)\phi - F_p = 0$$

Overall balance on component A:

$$f_5 = F_A - R_1 - F_{RA}\phi = 0$$

Overall balance on component B:

$$f_6 = F_B - R_1 - R_2 - F_{RB}\phi = 0$$

Overall balance on component C:

$$f_7 = (M_C/M_B)(R_1 - R_2) - R_3 - F_{RC}\phi = 0$$

Overall balance on component G:

$$f_8 = (M_G/M_C)R_3 - F_G = 0$$

Definition of F_R:

$$f_9 = F_R - F_{RA} - F_{RB} - F_{RC} - F_{RE} - F_{RP} - F_G = 0$$

Definition of t:

$$f_{10} = t - V\rho/F_R^2 = 0$$

Kinetic equation for reaction 1:

$$f_{11} = R_1 - A_1 e^{-B_1/T} F_{RA}F_{RB}t = 0$$

Kinetic equation for reaction 2:

$$f_{12} = R_2 - A_2 e^{-B_2/T} F_{RB}F_{RC}t = 0$$

Kinetic equation for reaction 3:

$$f_{13} = R_3 - A_3 e^{-B_3/T} F_{RP}F_{RC}t = 0$$

Definition of bottoms split ratio ϕ:

$$f_{14} = \phi - F_D/(F_R - F_G - F_P) = 0$$

Inequality constraints: (5)

$$h_1 = T - 580 \geqq 0$$

$$h_2 = 680 - T \geqq 0$$

$$h_3 = 1 - \phi \geqq 0$$

h_4 through h_{19}. All variables (except T, which is already constrained by h_1 and h_2) $\geqq 0$

Objective function: (6)

Percent return:

$$z = (368\ F_P + 8.4\ F_D - 28\ F_A - 42\ F_B - 14\ F_G - 0.37\ F_R)/V\rho - 10$$

The flow rates F are in pounds per hour; reactor capacity $V\rho$ is in pounds; the M_1 represent the molecular weights of the respective components; and the kinetic constants are based on mass fraction and temperature T in degrees Rankine. The values of the constant parameters of the system are given in reference *1*.

The development of the objective function is given by Williams and Otto (*2*); its general form is explained by noting that P is a salable product and D a fuel which may be used for process power generation. Costs for cooling water and other utilities are roughly proportional to the total throughput F_R, while costs for waste disposal depend on the amounts of components A, B, and G present. The total investment is approximately scaled to the capacity $V\rho$ of the reactor, and fixed costs are approximately 10% of the total investment.

This type of problem may be solved by the techniques of mathematical programming. If the constraints and objective function contain solely linear functions of the vector **x** of process variables, then the well-known and efficient simplex method of linear programming or one of its variants may be used (*3*). If the problem contains nonlinear functions, then many nonlinear programming methods are available (*4* to *6*). It is characteristic of mathematical programming methods that the computational effort required to solve the problem increases very rapidly with the number of constraints. A generalization of a previous paper (*7*) will now be presented which will use the structure of the system of equations to reduce the number of equality constraints which actually enter into the mathematical programming problem.

State and Design Variables

The basic procedure will be to rearrange the vector **x** so that it may be partitioned into two parts: the state variables $\mathbf{x}_s = (x_1, x_2, \ldots, x_{m-k})$ and the design variables $\mathbf{x}_d = (x_{m-k+1}, \ldots, x_n)$. This partition is chosen such that the state variables may be computed noniteratively as functions of the design variables; that is, the equality constraints may be rearranged to give

$$\left.\begin{array}{l} x_i = g_i(x_1, \ldots, x_{i-1}, \mathbf{x}_d), \quad i = 1, \ldots, m-k \\ [\text{equivalently, } \mathbf{x}_s = \mathbf{g}(\mathbf{x}_d)]; \\ f_i(\mathbf{x}_s, \mathbf{x}_d) = 0, \quad i = m-k+1, \ldots, m \end{array}\right\} \tag{7}$$

The programming problem may then be rewritten as

$$\left.\begin{array}{ll} \text{maximize} & z[\mathbf{g}(\mathbf{x}_d), \mathbf{x}_d] \\ \text{subject to} & \mathbf{f}_d[\mathbf{g}(\mathbf{x}_d), \mathbf{x}_d] = \mathbf{0} \\ \text{and} & \mathbf{h}[\mathbf{g}(\mathbf{x}_d), \mathbf{x}_d] \geqq \mathbf{0} \end{array}\right\} \tag{8}$$

Each state variable has been assigned as the output variable of a design equation. The total number of inequality constraints has been reduced to the k equations in the tear set $\mathbf{f}_d$, so called because the problem is torn apart at these equations, and it is up to the mathematical programming method to put it back together again by adjusting the design variables to maximize the objective while satisfying the torn equations. Some means must now be found to reduce as far as possible the number of tears k.

Graphical Representation

The directed graphs representing recycle calculations (8) are inadequate to represent the present problem because the outputs of each equation are not fixed at the outset. Furthermore, the equations and variables may not be neatly associated into units and streams such that each stream links only two units.

A more adequate representation is provided by the bipartite graph (9) often associated with similar assignment problems (3, p. 367). This graph will consist of two sets of nodes. Each equality constraint will be represented by a separate node in the set of f nodes, and each component of the vector $\mathbf{x}$ of process variables by a separate node in the set of x nodes. An edge connecting f node f_i and x node x_j will correspond to an appearance of variable x_j in the equality constraint $f_i = 0$. The edge (f_i, x_j) will be directed from f_i to x_j if x_j is chosen as the output of the equation $f_i = 0$; otherwise, the edge is given the reverse direction; that is, x_j is an input to f_i.

Consider, for example, the countercurrent stirred-tank reactor problem with N stages, discussed in references 8 and 10. A computational scheme which iterates upon only two variables, independent of the number of stages, might be the following:

1. Assume values for the set $\mathbf{x}_d = (A_1, B_1)$.
2. Compute A_{i+1} from the equation $f_{Ai} = 0$ and B_{i+1} from $f_{Bi} = 0$, in order for $i = 1, 2, \ldots, N - 1$.
3. Compute $\mathbf{f}_d = (f_{AN}, f_{BN})$, and iterate upon $\mathbf{x}_d$ to force $\mathbf{f}_d$ to $\mathbf{0}$.

For $N = 3$, this corresponds to the acyclic graph of Figure 1. A system of six equations and variables has thus in effect been reduced to one of two equations and variables. This iteration scheme is essentially the same as one of those proposed by Frank (10) and is in fact k minimal, as will be shown later.

ALGORITHM I-T FOR ACYCLIC OUTPUT SET ASSIGNMENT

If the f nodes corresponding to the tear set $\mathbf{f}_d$ are removed from the graph, then the remaining subgraph must be capable of assignment of an output set (12) yielding an acyclic directed graph in which each f node has exactly one output and each x node has at most one input. Algorithm I of (7) is based on the fact that such a graph must have at least one x node with exactly one edge. This is so because every directed path in the graph must terminate by definition, and since every f node has one outgoing edge, the path must terminate at an x node with one incoming edge and no outgoing edge. If this x node and the f node of which it is the output are deleted, the remaining subgraph must also be acyclic and satisfy the same assignment criteria. The deletion process can thus be repeated until only the x nodes corresponding to design variables are left, unattached to any f nodes. The order in which the equations may be solved is then given by the reverse of the order of deletion.

For example, if the tear set (f_{A3}, f_{B3}) is deleted from the graph of Figure 1, deletion may proceed in the order (f_{B2}, B_3), (f_{A2}, A_3) (f_{B1}, B_2), (f_{A1}, A_2). X nodes A_1 and B_1 are then left unattached and constitute the design variable set. Algorithm I-T, with the flow chart shown in Figure 2, represents the systematic application of this deletion process to determine whether a proposed tear set can produce an acyclic output set assignment.

A LOWER BOUND ON k^*

One means of reducing the combinatorial effort of finding a feasible tear set yielding a minimum number k^* of tears would be to obtain a lower bound $k_{\min}$ for the number of tears in any feasible set. Assume that algorithm I-T has been applied to the original undirected graph of the system. If the remaining graph G_0 is not empty, then acyclic assignment with $k = 0$ is impossible; furthermore, any x node x_j in G_0 has at least two edges, that is, its local degree $\rho(x_j) \geqq 2$. One now deletes a trial set of f nodes from G_0 and uses algorithm I-T to determine if the set is feasible. But algorithm I-T cannot produce any assignments unless this deletion yields at least one x node with at most one edge; hence, one needs at least

$$k_{\min} = \min_{G_0} \rho(x_j) - 1 \tag{9}$$

INDEXING

Since, with any feasible choice of a trial tear set, algorithm I-T will immediately assign and delete an x node with just one edge, one might as well get a head start on it by deleting the x node and all its attached f nodes. It

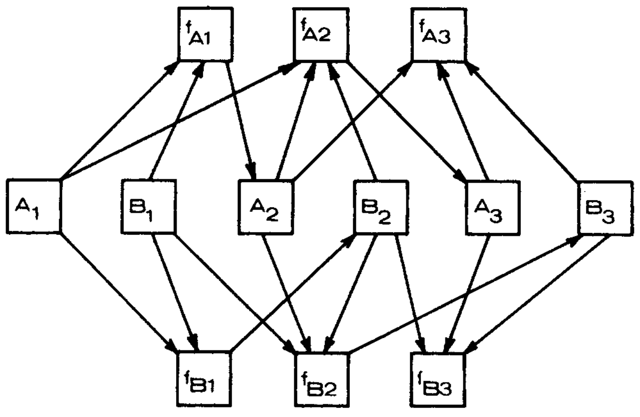

Fig. 1. Acyclic output set assignment.

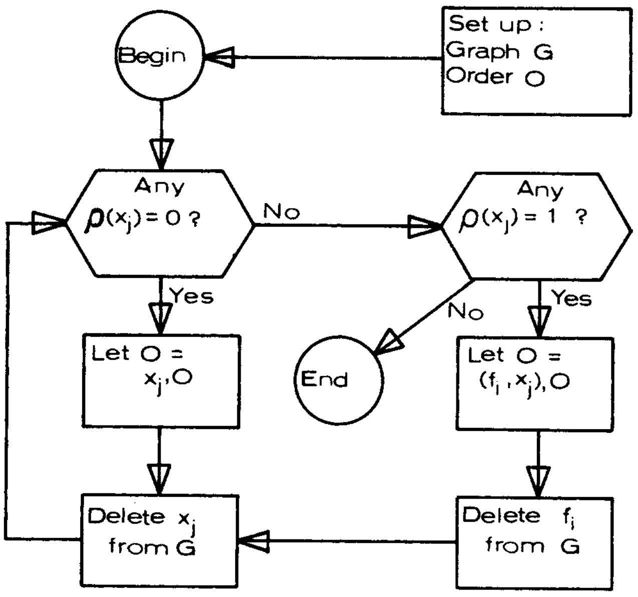

Fig. 2. Flow chart of algorithm I-T.

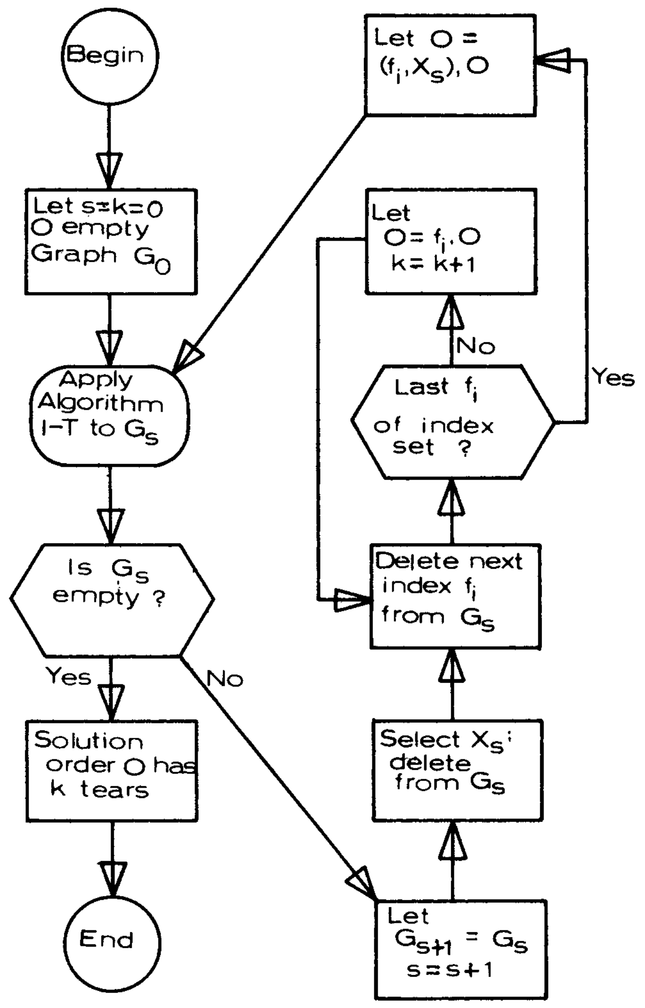
Begin
Let O = (f_i, X_s), O
Let s=k=0
O empty
Graph G_0
Let
O= f_i, O
k= k+1
No
Yes
Apply Algorithm I-T to G_s
Last f_i of index set ?
Is G_s empty ?
Delete next index f_i from G_s
Yes
No
Solution order O has k tears
Select X_s; delete from G_s
End
Let G_{s+1} = G_s
s=s+1

Fig. 3. Indexing on a sequence of nodes.

then makes no difference to which f node the x node is assigned as an output; the remaining f nodes will constitute the trial tear set. Since the x node will be an input of, or point to, the tear set, it will be called the *index node*, and the set of nodes consisting of the index node and all its attached f nodes will be designated as the *index set*. The process of deleting an index set and applying algorithm I-T to the resulting subgraph will be referred to as *indexing* on the corresponding index node.

When $k^* = k_{\min}$, indexing on a single node will produce a minimal tear assignment. For instance, in the example of Figure 1, $k_{\min} = 2$ at x nodes A_1, B_1, A_3, and B_3. If, for example, the tear set $(B_3, f_{A3}, f_{B3}, f_{B2})$ is deleted, algorithm I-T will produce the same deletion sequence as before, with the exception of (f_{B2}, B_3). B_3 may now be assigned as the output of either f_{A3}, f_{B3}, or f_{B2}, with the remaining two f nodes forming the tear set $\mathbf{f}_d$, and an assignment has been found with $k^* = k_{\min} = 2$ tears.

Another case in which $k^* = k_{\min}$ is the process optimization example given at the beginning of this paper, where $k_{\min} = 1$ at x nodes F_A, F_B, and F_D. Indexing on F_A yields F_{RP}, (f_2, F_{RE}), R_2, (f_3, ϕ), (f_4, R_3), R_1, (f_7, F_{RC}), T, (f_{13}, t), (f_{12}, F_{RB}), (f_6, F_B), (f_{11}, F_{RA}), (f_8, F_G), (f_9, F_R), (f_{14}, F_D), (f_5, F_A), f_1, (f_{10}, V) as an order in which the equations may be solved with $k^* = 1$ tear. Output assignments are here indicated as parenthesized (f, x) pairs, and design variables and torn equations are unparenthesized.

An extreme case in which $k^* = k_{\min}$ occurs when each variable appears in all the equations. Then $k_{\min} = m - 1$, and deletion of any index set deletes all the f nodes. All the other variables then are the set $\mathbf{x}_d$, and $k^* = k_{\min} = m - 1$; hence, for any system, $k_{\min} \leqq k^* \leqq m - 1$.

SEQUENCES OF INDEX NODES

When $k^* > k_{\min}$, it may be necessary to index on a sequence of nodes $(X_1, X_2, \ldots, X_s)$ in order to produce an assignment. This is done by applying algorithm I-T to the original graph, producing a first subgraph G_0. One then indexes on X_1, producing G_1, then on X_2, producing G_2, etc., until indexing on X_s produces an empty subgraph G_s. The order of solution of the equations is then reconstructed from the reverse of the partial sequences produced by algorithm I-T. This indexing procedure is represented schematically in Figure 3.

As an example, consider the original system of equations given by diBella and Stevens (*1*) without benefit of the additional definitions f_{10}-f_{14}, which has the structure shown in Table 1. It will be shown later that $k^* = 3$ for this example; an index node sequence producing this number of tears is (F_A, F_D). Thus, deletion of the index set (F_A, f_1, f_5) from G_0 yields the single assignment (f_6, F_B); now, deletion of (F_D, f_3, f_4, f_7) from G_1 yields the assignments (f_9, F_{RB}), F_{RA}, (f_8, F_R), F_G, V, F_{RC}, T, (f_2, F_{RP}), F_{RE}, giving an empty G_2. A solution order is then F_{RE}, (f_2, F_{RP}), T, F_{RC}, V, F_G, (f_8, F_R), F_{RA}, (f_9, F_{RB}), (f_3, F_D), f_4, f_7, (f_6, F_B), (f_1, F_A), f_5.

It is apparent that the number of tears required to index on a sequence of nodes is given by

$$k_s = \sum_{r=1}^{s} \rho(X_r)_{r-1} - 1 \tag{10}$$

where $\rho(X_r)_{r-1}$ is the local degree of index node X_r in subgraph G_{r-1}. Thus, in the above example, $k_2 = (2 - 1) + (3 - 1) = 3$.

Equation (10) may also be expressed recursively by letting k_r be the number of tears required to index on the sequence $(X_1, \ldots, X_r)$; then

$$k_r = k_{r-1} + \rho(X_r)_{r-1} - 1 \tag{11}$$

where $1 \leqq r \leqq s$, and $k_0 = 0$ by definition.

The minimal tear problem may now be restated as follows. Find an index-node sequence $(X_1, \ldots, X_s)$ which transforms G_0 into an empty graph for which the quantity k_s defined in (10) is a minimum k^* over all possible sequences.

THE FORWARD PASS, DUALITY, AND INCONSISTENCY

The number of possible index-node sequences may often be reduced by assigning an output to any f node which has just one edge and by deleting the (f, x) pair from the graph. If the x node were not assigned as the output of the f node in a minimal tear assignment, then

TABLE 1. THE ORIGINAL SYSTEM OF DIBELLA AND STEVENS (*1*)

Equation	Variables
f_1	F_A, F_B, F_G, F_D
f_2	F_{RP}, F_{RE}
f_3	$F_{RB}, F_{RC}, F_R, V, F_D, F_{RE}, F_G, T$
f_4	$F_{RB}, F_{RC}, F_{RP}, V, F_R, T, F_D, F_G$
f_5	$F_{RA}, F_{RB}, V, F_R, F_D, F_G, F_A, T$
f_6	$F_{RA}, F_{RB}, F_{RC}, V, F_R, F_D, F_G, F_B, T$
f_7	$F_{RA}, F_{RB}, F_{RC}, F_{RP}, V, F_R, T, F_D, F_G$
f_8	$T, F_{RC}, F_{RP}, V, F_R, F_G$
f_9	$F_{RA}, F_{RB}, F_{RC}, F_{RE}, F_G, F_{RP}, F_R$

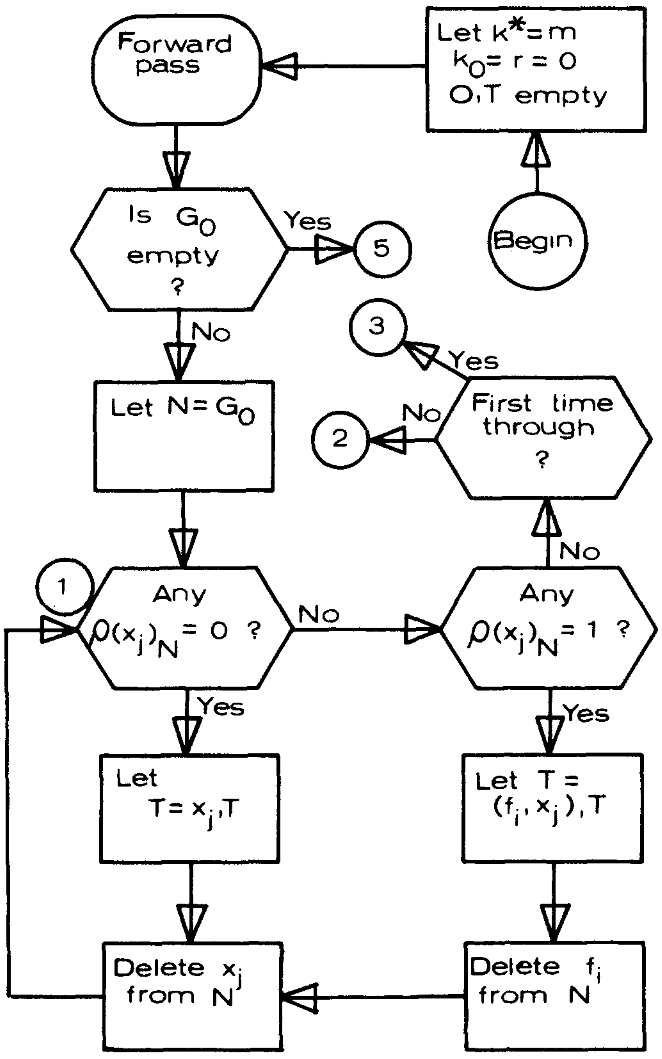

Fig. 4a. Flow chart of algorithm II-T.

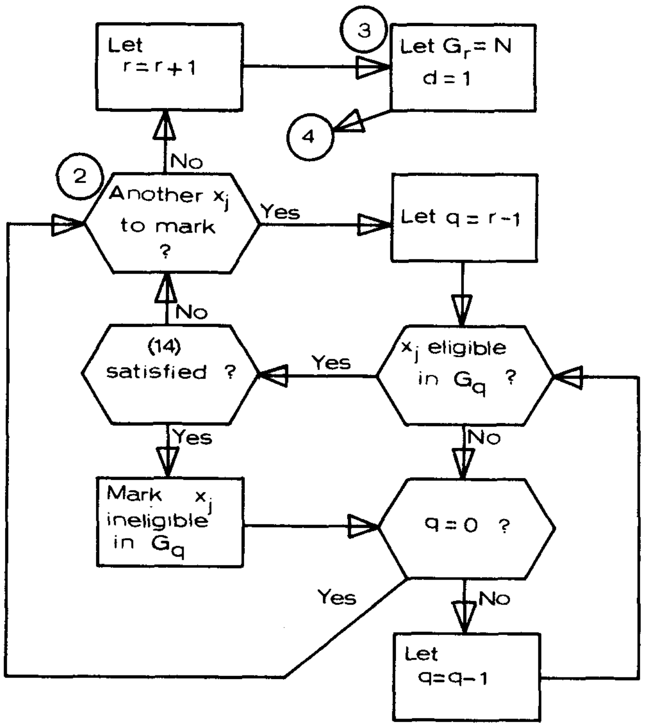

Fig. 4b. Flow chart of algorithm II-T.

the f node would have to be in the tear set. But then the x node would have to be an index variable and could always be reassigned as an output of the f node without changing the minimality of the assignment. The deletion process may be repeated until there are no f nodes left with just one edge.

This is the first part of algorithm I in reference 7 and will be called the *forward pass*, since deletion proceeds in the same order in which the equations may be solved, in contrast to the reverse pass of algorithm I-T. The forward pass need not be applied to any subgraph after G_0, since deletion of an index set does not decrease the local degree of any f node remaining in the subgraph.

This procedure is the equivalent of applying algorithm I-T to the graph of the dual problem (3, pp. 221 to 39), that is, a graph in which the x nodes are relabeled as *f nodes* and vice versa. One may thus encounter the dual of a design variable assignment, that is, an f node with no outputs. This means that the portion of the graph so assigned contains fewer x nodes than f nodes, as illustrated in reference 7. Thus, Hall's diversity criterion (9) is not satisfied, and the f node so assigned is a priori algebraically redundant or inconsistent.

AN EXCLUSION CRITERION

The number of sequences to be examined can be drastically reduced by excluding from consideration those which cannot possibly produce fewer tears than sequences which have already been considered. A criterion for this exclusion can be developed as follows. Let $(X_{q+1}, \ldots, X_r)$ be a sequence of index nodes which transforms subgraph G_q into G_r, and let X_{r+1} be an index node transforming G_r into G_{r+1}. Then, from Equation (11)

$$k_{r+1} = k_r + \rho(X_{r+1})_r - 1 \qquad (12)$$

Now, if indexing on X_{r+1} transforms G_q into G'_{q+1}, then obviously all the nodes of G_{r+1} are also contained in G'_{q+1} $(G_{r+1} \in G'_{q+1})$; hence, at least as many tears are required to transform G'_{q+1} into an empty graph as to transform G_{r+1} into an empty graph. Thus

$$k'_{q+1}{}^* \geqq k^*_{r+1} \qquad (13)$$

where k_p^* represents the number of tears required to transform an arbitrary subgraph G_p into an empty graph. Note that by this definition, $k^* = k_0^*$.

Now, if

$$\rho(X_{r+1})_q - \rho(X_{r+1})_r \geqq k_r - k_q \qquad (14)$$

then solving (12) for k_r, substituting the result into (14), and rearranging we get

$$\rho(X_{r+1})_q - 1 \geqq k_{r+1} \qquad (15)$$

Adding this expression to (13) we get

$$\rho(X_{r+1})_q - 1 + k'_{q+1}{}^* \geqq k_{r+1} + k^*_{r+1} \qquad (16)$$

This means that if condition (14) is met, then G_q cannot be transformed into an empty graph by any sequence beginning with X_{r+1} with fewer tears than by the sequence $(X_q, \ldots, X_r, X_{r+1})$. X_{r+1} may thus be marked *ineligible* for indexing on G_q.

The criterion (14) may also be applied to nodes X_{r+1}, which are not actually in X_r, by setting $\rho(X_{r+1})_r = 1$, since indexing on such a node would give $k_{r+1} = k_r$. Such a procedure is adequate for marking X_{q+1} itself ineligible in G_q, which is necessary to prevent repeated examinations of the same sequence.

ALGORITHM II-T FOR MINIMUM TEARS

The above considerations are incorporated into algorithm II-T for finding a minimum tear assignment by indexing, shown in Figures 4*a*, *b*, *c*. The bottom half of Figure 4*a* will be recognized as the application of algorithm I-T to transform subgraph N from G_r to G_{r+1}.

Figure 4*b* shows the use of criterion (14) in the learning step of the algorithm which encodes the experience just obtained by eliminating unprofitable sequences from future consideration. Note that the testing of a variable for eligibility in higher subgraphs can cease as soon as the criterion fails; that is

$$\rho(x_j)_q - \rho(x_j)_N < k_{r+1} - k_q \tag{17}$$

for, if the test were to succeed in G_p with $p < q$

$$\rho(x_j)_p - \rho(x_j)_N \geqq k_{r+1} - k_p \tag{18}$$

Then, taking the negative of (17) and adding, we obtain

$$\rho(x_j)_p - \rho(x_j)_N > k_q - k_p \tag{19}$$

and x_j would already have been marked ineligible in G_p by a previous step.

The right half of Figure 4*c* represents the search for a new indexing step which might yield fewer tears than the minimum found to date, while the left half is the procedure for resuming the search at the next higher level. This is done either when a new minimum k^* is found or when a search on the present subgraph cannot possibly find a smaller k^*. The search is terminated at step 5 when a search on G_0 cannot further improve k^*.

This algorithm meets the need expressed in reference 7

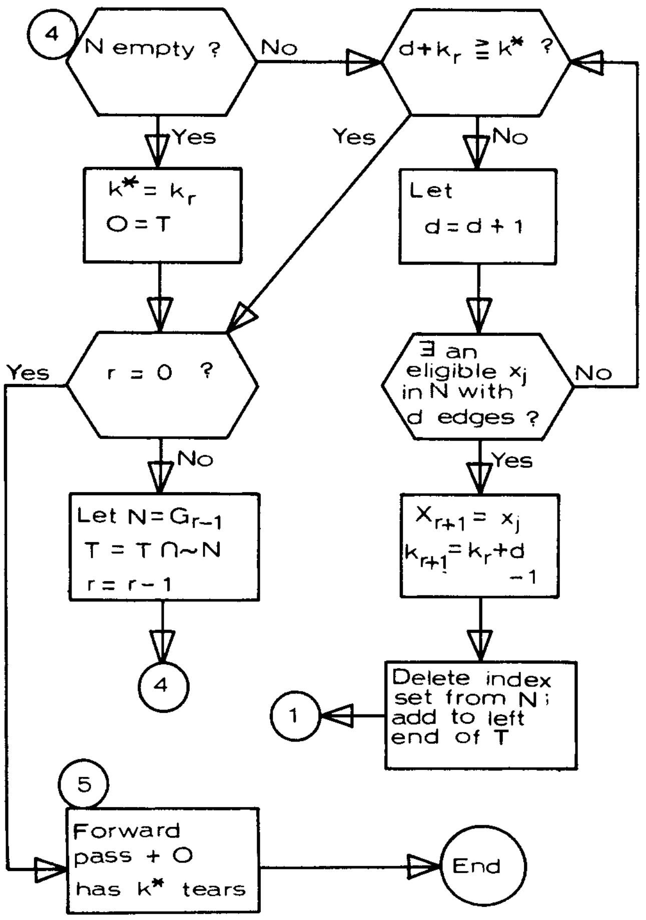

Fig. 4c. Flow chart of algorithm II-T.

Table 2. Application of Algorithm II-T to the System of Table 1

Step 1: Index F_A on G_0
Assignments: f_1, (f_5, F_A), (f_6, F_B)
Nodes marked in G_0: F_A, F_B, F_D, V, T, F_R, F_G, F_{RC}, F_{RB}, F_{RA}

Step 2: Index F_{RA} on G_1
Assignments: f_7, (f_9, F_{RA})
All nodes marked in G_1
F_{RP} marked in G_0

Step 3: Index F_D on G_2
Assignments: f_3, (f_4, F_D), F_{RB}, (f_8, V), T, F_R, F_G, F_{RC}, (f_2, F_{RP}), F_{RE}
All nodes marked in G_2
G_3 is empty; set $k^* = 3$; ascend twice

Step 4: Index F_{RE} on G_0
Assignments: f_2, f_3, (f_9, F_{RE})
F_{RE} marked in G_0
All nodes marked in G_0; can't ascend; end of algorithm

Final order:
F_{RE}, (f_2, F_{RP}), F_{RC}, F_G, F_R, T, (f_8, V), F_{RB}, (f_4, F_D), f_3, (f_9, F_{RA}), f_7, (f_6, F_B), (f_5, F_A), f_1

for an efficient means of directing the search for a minimal tear assignment. Memory requirements are not large, since relatively few subgraphs, namely G_0 through G_r, need be retained. For example, the application of the algorithm to the system of Table 1 is shown in Table 2. Note that a maximum of four subgraphs had to be retained in memory, and that only four trial indexing steps were needed to find that $k^* = 3$ tears. The optimal index-node sequence (F_A, F_{RA}, F_D) produces the same number of tears as the sequence (F_A, F_D) examined previously, because $\rho(F_D)$ is reduced by indexing on F_{RA} in going from G_1 to G_2.

Algorithm II-T has been programmed in SNOBOL and FORTRAN II for the IBM-1620 computer and used to solve example problems from a variety of sources (for example, *11*, *12*). No more than four tears are required in any of these problems which have up to forty-five equations and variables each. Tearing is thus a powerful tool for simplifying the algebraic and numerical analysis of models of systems with large numbers of interacting units, such as those encountered in chemical process design and economic modeling.

COMPARISON WITH PREVIOUS WORK

The inventor of the term *tearing* was Kron, one of the earliest practitioners of the piece-by-piece solution of large systems (*13*). His insight appears limited to systems which can be reduced to equivalent electrical network problems and hence lacks the generality of the algebraic approach taken in this paper.

The method of tearing presented here is less strict in its output set requirements than previous methods (*7*, *12*) in that it is not required that the equations in the tear set have as outputs k of the design variables, the tear variables. This often allows an acyclic assignment with fewer tears than the latter method. For instance, in the countercurrent-reactor example, it is difficult to envision an order using tear variables which involves fewer than N tears, compared with only two torn equations independent of the number of stages N.

PARTITIONING

Steward (*12*) notes that when $m = n$, the solution of the system of equations may be speeded up by partitioning the equations into blocks such that iteration takes place within but not between the blocks. This is similar to the partitioning of recycle computations discussed previously (8). Such blocks are easily detected in the final order given by algorithm II-T; an iterative block is found whenever the number of torn equations is equal to the number of design variables. Because of the relaxed output set assignment criterion, the partition is not unique over all possible output set assignments as is Steward's. Because of this nonuniqueness, partitioning before tearing, as Steward recommends, may yield blocks which cannot be torn to give the true minimum total tears. Thus, algorithm II-T gives a single block with $k^* = 2$ tears for the twenty-six equation example in which Steward finds two blocks with one and three tears, respectively.

ELIMINATING TRANSCENDENTAL EQUATIONS

If algorithm II-T assigns a variable as output of an equation in which it appears transcendentally, then the equation would have to be solved iteratively. This may greatly increase the time required to solve the problem because of the additional computation needed to perform this internal iteration within the nonlinear program, and because of the truncation errors which may be introduced.

This situation can be avoided by making all the equations nontranscendental through the introduction of dummy variables. For instance, the equation

$$f = y - xe^x = 0 \tag{20}$$

can be solved algebraically for y but not for x. If, however, the single Equation (20) is replaced by the two equations

$$f' = z - e^x = 0; \quad f = y - xz = 0 \tag{21}$$

then either equation can be solved algebraically for any of the variables appearing in it. Note that the first equation serves to define the dummy variable z, and that the constraint $z - d \geqq 0$, where $d > 0$, must be added to the inequality constraints. Also, since the function xe^x attains a minimum at $x = -1$, y must be constrained by $y - e^{-1} \geqq 0$.

REDUCING TEARS BY ALGEBRAIC SUBSTITUTION

The representation of the system of diBella and Stevens was chosen in order to provide clarity and compactness in displaying the equations. However, it turns out that this system requires only one tear, as shown previously, in comparison with three for the original system shown in Tables 1 and 2. Algebraic substitutions such as those in Equations (4) $f_{10\text{-}14}$ can thus reduce k^* by decoupling the equations through a reduction of the number of variables per equation.

There does not appear to be any general rule for predicting when substitution of a new variable for a combination of variables will reduce k^*. However, it certainly will not increase k^* and hence is advisable whenever the combination appears in two or more equations. This requires a good eye for combinations and at present can be done more efficiently by a human being in formulating the problem than by a computer in analyzing it.

ELIMINATING REDUNDANT CONSTRAINTS

In solving Equations (4) numerically in the order indicated earlier, one finds that the torn function f_1 is equal to zero for any value of the design variables; thus, the equation $f_1 = 0$ is redundant. This is due to the fact that diBella and Stevens, in formulating the problem, used both an overall material balance (f_1) and balances for each of the components ($f_{3\text{-}9}$, f_{14}), which constitute a redundant subset of equations.

The elimination of the redundant torn equation $f_1 = 0$ gives a zero tear ordering which contains no redundant equations. This is obvious, since the Jacobian matrix $(\partial \mathbf{f}/\partial \mathbf{x}_s)$ of the ordered system is triangular, and, since each equation can be solved for its output, the diagonal elements are nonzero. Hence, the Jacobian is nonsingular, and $\mathbf{x}_s$ can be determined as the vector function $\mathbf{g}(\mathbf{x}_d)$ (14).

Solving the equations algebraically in the order given by algorithm II-T will show if any of the inequality constraints on the state variables are redundant and can be eliminated. Thus, rearranging the order slightly to simplify the algebra, we get

$$\left.\begin{array}{ll}
\text{Design variable:} & F_{RE} \\
f_2: & g_1 = F_{RP} = F_P + 0.1\ F_{RE} \\
\text{Design variable:} & \phi \\
f_3: & g_2 = R_2 = (M_B/M_E)F_{RE}\phi \\
f_4: & g_3 = R_3 = (M_C/M_P) \\
 & \qquad (R_2 - F_P - F_{RP}\phi + F_P\phi) \\
\text{Design variable:} & F_{RC} \\
f_7: & g_4 = R_1 = (M_B/M_C)(R_3 + F_{RC}\phi) + R_2 \\
\text{Design variable:} & T \\
\text{Calculate} & k_i = A_i e^{-B_i/T}, \quad i = 1, 2, 3 \\
f_{13}: & g_5 = t = R_3/k_3 F_{RP} F_{RC} \\
f_{12}: & g_6 = F_{RB} = R_2/k_2 F_{RC} t \\
f_6: & g_7 = F_B = R_1 + R_2 + F_{RB}\phi \\
f_{11}: & g_8 = F_{RA} = R_1/k_1 F_{RB} t \\
f_8: & g_9 = F_G = (M_G/M_C)\ R_3 \\
f_9: & g_{10} = F_R = F_{RA} + F_{RB} + F_{RC} \\
 & \qquad + F_{RE} + F_{RP} + F_G \\
f_{14}: & g_{11} = F_D = \phi(F_R - F_G - F_P) \\
f_5: & g_{12} = F_A = R_1 + F_{RA}\phi \\
f_{10}: & g_{13} = V = F_R{}^2 t/\rho
\end{array}\right\} \tag{22}$$

Examination of these equations shows that the following set of inequality constraints is sufficient to guarantee that all the constraints (5) are satisfied:

$$\left.\begin{array}{l}
h_1 = F_{RE} \geqq 0 \\
h_2 = \phi \geqq 0 \\
h_3 = 1 - \phi \geqq 0 \\
h_4 = R_3 \geqq 0 \\
h_5 = F_{RC} \geqq 0 \\
h_6 = T - 580 \geqq 0 \\
h_7 = 680 - T \geqq 0
\end{array}\right\} \tag{23}$$

The original optimization problem in seventeen variables with fourteen equality and nineteen inequality constraints has now been converted into one in four variables with no equalities and seven inequalities. Furthermore, all the constraints except h_4 are rectangular in the design variables, which greatly simplifies the task of finding an initial feasible solution and carrying out the optimization.

STABILITY

One drawback of minimal tear iteration schemes is that the tear equations may be so far removed from the design variable set that the functions involved become extremely nonlinear. Thus, Frank (*10*) reported that the two tear iteration scheme could not be made to converge for initial guesses that were not close to the true solution for the countercurrent-reactor example.

The relative stability of the iteration scheme developed for this example in reference *8* suggests that the situation might, in general, be improved by forcing equations in the middle of the system into the tear set, if the initial ordering proves too unstable. This can be done by attaching to each such equation a fake x node with only one edge. The f node corresponding to the equation would then be removed in the first pass, before any indexing occurs. This procedure could form the basis of an algorithm which would start with a minimal tear ordering and modify it step-by-step until satisfactory stability is obtained.

EXTENSION TO MINIMUM RECYCLE PARAMETERS

Algorithm II-T may be modified to solve the problem of finding minimum recycle parameters (*8*) by rewriting criterion (14) as

$$w_+(X_{r+1})_q - w_+(X_{r+1})_r \geqq k_r - k_q \qquad (24)$$

where $w_+(X_{r+1})_q$ is the number of incoming parameters to node X_{r+1} in subgraph G_q, which is zero if X_{r+1} is not in G_q.

Indexing is performed by using those portions of algorithm II-R of reference *8* which deletes nodes to head and tail strings instead of algorithm I-T. Tears are counted by modifying Equation (11) to

$$k_r = k_{r-1} + w_+(X_r)_{r-1} \qquad (25)$$

As an example of the application of this algorithm, consider a complete graph (*9*) on five nodes. Each node has two input and output parameters; hence, the first pass produces no simplification. Indexing on node 1 produces a graph in which nodes 2 and 3 each have only one input parameter. Indexing on node 2 gives the final order (1, 2, 3, 4, 5), with $k = 3$ recycle parameters. It is obvious from symmetry that this is the minimum, but the algorithm would have to try four more indexing sequences in order to be sure. However, this does not seem an excessive number of sequences; in problems with simpler structures, many fewer sequences would be required in relation to the number of nodes.

CONCLUSIONS

The use of recycle parameters and tearing is a potent tool for simplifying the analysis of models of large systems of interacting units, such as those found in chemical process design. Tearing gives more insight into the algebraic structure of systems of equations and optimization problems and is therefore more generally applicable to a wide variety of systems.

The concept of indexing leads to computationally efficient algorithms for determining an order of solution for the system which yields the minimum number of tears or of recycle parameters. These algorithms should be highly useful in computer programs for the automatic solution of complex problems.

ACKNOWLEDGMENT

This work was made possible by the advice and assistance of Associate Professor Dale F. Rudd and by grants and graduate fellowships of the National Science Foundation.

Listings and example runs of the SNOBOL and FORTRAN II-D versions of algorithms I-T and II-T may be found on pages 98 to 127 of the author's Ph.D. dissertation at the University of Wisconsin, Madison, Wisconsin, copies of which are available from University Microfilms, Ann Arbor, Michigan.

NOTATION

$\mathbf{f}$	= m dimensional vector of equality constraint functions
$\mathbf{f}_d$	= k dimensional vector of torn equation functions
G_q	= q^{th} index subgraph
G_0	= initial graph of the system
$\mathbf{g}$	= $(m - k)$ dimensional vector of equality constraint functions
$\mathbf{h}$	= m_2 dimensional vector of inequality constraint functions
k	= number of torn equations
k^*	= minimum value of k for a system, that is, k_0^*
$k_{\min}$	= lower bound for k^*
k_q	= number of tears required to reduce graph G_0 to subgraph G_q
k_q^*	= minimum number of tears to reduce subgraph G_q to the empty graph
N	= total number of countercurrent reactor stages; also, temporary storage for graph operations
$T \cap \sim N$	= set of nodes in graph T but not in graph N
$w_+(A)_q$	= number of input parameters to unit A in index subgraph G_q for recycle calculations
X_q	= q^{th} index variable
$\mathbf{x}$	= n dimensional vector of process variables
$\mathbf{x}_d$	= $(n - m + k)$ dimensional vector of design variables
$\mathbf{x}_s$	= $(m - k)$ dimensional vector of state variables
$z(\mathbf{x})$	= (scalar) objective function
$\rho(A)$	= local degree of node A in subgraph G_0, that is, $\rho(A)_0$
$\rho(A)_q$	= local degree of node A in subgraph G_q

LITERATURE CITED

1. diBella, C. W., and W. F. Stevens, *Ind. Eng. Chem. Process Design Develop*, 4, No. 1, 16-20 (Jan., 1965).
2. Williams, T. J., and R. E. Otto, *Trans. Am. Inst. Elect. Engrs.*, **79**, 458-73 (Nov., 1960).
3. Hadley, G., "Linear Programming," Addison-Wesley, Reading, Mass. (1963).
4. ———, "Nonlinear and Dynamic Programming," Addison-Wesley, Reading, Mass. (1964).
5. Hens, A. M., paper presented at AIChE-IChemE Joint Meeting, London, England (1965). Preprint 4.14.
6. Ornea, J. C. and G. G. Eldredge, *ibid.*
7. Lee, Wooyoung, J. H. Christensen, and D. F. Rudd, *AIChE J.*, **12,** No. 6, 1104-10 (Nov., 1966).
8. Christensen, James H., and Dale F. Rudd, *ibid.*, **15,** No.1, 94-100 (Jan. 1969).
9. Ore, O., "Graphs and Their Uses," Random House, New York (1963).
10. Frank, Andrew, *Chem. Eng. Progr. Symposium, Ser. No. 72,* **63,** 54-58 (1967).
11. Rosen, E. M., *Chem. Eng. Progr.*, **58**, No. 10, 69-73 (Oct., 1962).
12. Steward, D. V., *SIAM Journal,* **2,** Series B, No. 2, 345-65 (1965).
13. Kron, G., "Diakoptics: The Piecewise Solution of Large-Scale Systems," Macdonald, London, England (1963).
14. Hildebrand, F. B., "Advanced Calculus for Applications," pp. 340-46, Prentice-Hall, Englewood Cliffs, N.J. (1963).

Manuscript received April 12, 1968; revision received July 26, 1968; paper accepted July 29, 1968.