A Simultaneous-Modular Approach to Process Flowsheeting and Optimization

Part I: Theory and Implementation

Three basic problem formulations for the simultaneous-modular approach are discussed, as are three alternative methods for computing the flowsheet-level Jacobian. A new algorithm for partitioning and tearing when using the simultaneous-modular approach is also presented. SIMMOD, our implementation of the simultaneous-modular approach is then outlined. In subsequent papers in this series we use SIMMOD to study the performance of the simultaneous-modular approach on process simulation and optimization problems, as well as to perform experiments on different numerical strategies for implementing the simultaneous-modular approach.

HERN-SHANN CHEN
and M. A. STADTHERR
Chemical Engineering Department
University of Illinois
Urbana, IL 61801

SCOPE

Current systems for the computer-aided flowsheeting and optimization of chemical processes are based primarily on the sequential-modular approach. However, there are serious drawbacks to this approach that are today increasingly being recognized. For instance when applied to complex processes with a number of recycle streams, the sequential-modular approach may become extremely slow, depending in part on the degree to which iteration loops are nested in the computation. It is perhaps of more importance to note that the sequential-modular approach does not efficiently handle problems in which a number of design constraints are imposed ("controlled" simulations), nor it is at all well-suited to the solution of optimization problems. For these reasons there has been considerable recent interest in developing alternatives to the sequential-modular approach. Two promising alternatives are the equation-based approach and the simultaneous-modular approach. In this paper we concentrate on the latter.

Several recent studies, including those by Mahalec et al. (1979), Perkins (1979), Jirapongphan (1980), and Biegler and Hughes (1981, 1982a, 1983) have demonstrated the promise of the simultaneous-modular approach for both flowsheeting and optimization problems. These studies, however, are typically fairly limited in scope. For this study, a general-purpose simulator, SIMMOD, based on the simultaneous-modular approach, was developed to provide a critical evaluation of the simultaneous-modular approach for simulation, controlled simulation, and optimization problems, and to provide a means for testing different computational strategies for implementing the simultaneous-modular approach.

This is the first in a series of papers in which we report on the results of this study. This first paper concentrates on the theory underlying the simultaneous-modular approach, and also provides a brief outline of our implementation SIMMOD. Three basic problem formulations for the simultaneous-modular approach are described, as are a number of alternatives for computing the required Jacobian matrix. Also, an algorithm for flowsheet partitioning and tearing in connection with the simultaneous-modular approach is presented. Parts II and III of this series will concentrate on the performance of the simultaneous-modular approach on several benchmark problems of both the simulation and optimization type, and will compare the performance of a number of computational strategies on these benchmark problems.

CONCLUSIONS AND SIGNIFICANCE

The simultaneous-modular approach for process flowsheeting and optimization is a promising alternative to the conventional sequential-modular approach. Our program SIMMOD is a general-purpose flowsheeting and optimization package designed specifically for implementing the simultaneous-modular approach. SIMMOD uses a problem formulation whose computational requirements are closely in line with the

Correspondence concerning this paper should be addressed to M. A. Stadtherr.
Hern-Shann Chen is now at Union Carbide Corporation, South Charleston, WV 25303.

usual sequential-modular simulators. Thus the results of the numerical studies performed using SIMMOD and detailed in subsequent papers can readily be applied in the implementation of the simultaneous-modular approach in existing sequential-modular simulators.

BACKGROUND

In this section we discuss in some detail the relationship between the different approaches to process flowsheeting and optimization, as well as describe their advantages and disadvantages. In subsequent sections we will describe alternative problem formulations for the simultaneous-modular approach, discuss alternatives for the calculation of the required Jacobian matrix, and present an algorithm for partitioning and tearing a flowsheet in the context of the simultaneous-modular approach. Finally, we describe one particular implementation of the simultaneous-modular approach, SIMMOD.

It should be noted at the outset that several extensive reviews of work in process flowsheeting have appeared in recent years, including reviews by Motard et al. (1975), Hlavacek (1977), Rosen (1980), and Evans (1981). The monograph of Westerberg et al. (1979) provides a good introduction to this field. An excellent review in the area of process optimization was given recently by Westerberg (1981), and nonlinear programming algorithms and software have been reviewed recently by Lasdon (1981) and Sargent (1980).

Sequential-Modular Approach

In its most fundamental form, the process flowsheeting problem can be regarded as one of solving a large system of nonlinear equations. The different approaches to process flowsheeting differ most fundamentally in their approach to solving this set of simultaneous equations. The equation system can generally be thought of as consisting of three types of equations:

- Model equations, including process unit models and physical property models
- Flowsheet connection equations that indicate how the units are connected together in the flowsheet
- Specifications

In the sequential-modular approach the model equations are handled using a library of "modules," or subroutines (procedures), each of which performs computations for one of the models used. The process unit modules are typically simulation-oriented; that is, given values for the variables describing the input streams and the equipment, the model equations are solved for the variables describing the output streams. The connection equations are handled implicitly, the executive routine passing output values from one unit module to other unit modules as inputs, as called for by the specified flowsheet topology. Specifications such as input stream data and equipment parameters are easily handled by passing the specified values directly to the proper unit modules. However other specifications, on output streams for instance, may not be nearly so easy to handle, since the unit modules may not accept the specified values as inputs. We refer here to specifications that can be passed directly to the modules as inputs as simple specifications. Other specifications that cannot be input directly to a module are referred to as design specifications. Problems for which all specifications are simple are usually called simulation problems. In this case, one essentially computes the performance of a process given its design. Problems that involve one or more design specifications are generally called controlled simulation problems, or simply design problems. In this case one essentially computes the design of a process given its desired performance.

Now consider a simulation problem for a process involving one or more recycle streams. In order to start the recycle calculations, values for one or more streams must be guessed, or torn. The modules are then called in the sequence indicated by the connection equations until new values for the tear streams are generated. This amounts to using the model equations, connection equations, and simple specifications to generate a set of equations of the form $x = f(x)$ for the tear stream values, which is then solved iteratively by the method of direct substitution (no acceleration) or, more frequently, by some sort of accelerated substitution scheme, such as the well-known Wegstein method or its variations. Thus we essentially have two levels of computation: a module level, in which the module computations are performed, and a flowsheet level, in which direct or accelerated substitution is used to converge the tear streams.

Now consider a controlled simulation problem in which one or more design specifications are involved. These equations cannot be handled on the module level because the specified values cannot be passed directly to the modules as inputs, nor can they be handled directly on the flowsheet level, since specifications are not naturally in the form $x = f(x)$. In the usual sequential-modular approach this sort of problem must be handled by an iterative simulation in which the process is repeatedly simulated until the design specifications are met. The iteration loops so generated are often referred to as control loops. It should be noted that some sequential-modular simulators, ASPEN for example, provide options that allow design specifications and tear streams to be converged simultaneously, thus in effect eliminating the need for control loops. As discussed in more detail below, this may be regarded as one form of the simultaneous-modular approach.

As mentioned above, the sequential-modular approach has been and still is by far the most common approach to process flowsheeting, particularly in industrial use. From a computational standpoint, perhaps its most important strong points are: 1. On the module level, one or more specialized algorithms can be used to solve the model equations within each module; thus the module calculations can be very efficient and robust. 2. On the flowsheet level, both direct and accelerated substitution are generally very reliable solution methods, though even accelerated substitution is generally rather slow. Many other strong points commonly cited can be attributed to the highly structured information flow in the sequential-modular approach. For instance, because the information flow in the program closely resembles the material flow in the process, sequential-modular programs are easily understood by the engineer. The information flow structure also makes error checking fairly easy and allows for generally easy tracebacks in the case of program failure.

We now turn our discussion to two major problems that seriously affect the efficiency of the sequential-modular approach, namely the handling of design specifications and the presence of multiple nested iteration loops. The first problem has already been alluded to above. The handling of design specifications, typically equations of the form x = constant, by introducing additional iteration loops is obviously an inefficient way to handle such simple equations. The second problem can be explained by noting that the accelerated substitution loops used to converge the tear streams, together with any control loops used to handle design specifications, may represent the outermost loops in a hierarchy of nested iteration loops. Within these loops are any iteration loops that may be needed within the process unit modules, and at the innermost level any loops that may exist within the physical property subroutines called by the unit modules. For design problems involving large and complex processes with several recycle streams, it is easy to imagine systems of nested iteration loops that would be very expensive and time-consuming to solve. Finally, a third problem with the sequential-modular approach should be noted, namely that it is not well suited to process optimization. This is because, using the sequential-modular approach, optimization is performed by adding yet another outer iteration loop, further exacerbating the numerical efficiency problems discussed above, thus accounting in part for the great computational expense that has in general been associated with process optimization (Westerberg, 1981).

Because of the fundamental problems described above, which become particularly severe in the case of complex processes, there has been considerable recent interest in two alternative approaches to process flowsheeting and optimization. These approaches, namely the equation-based approach and the simultaneous-modular approach, are discussed in detail below.

Equation-Based Approach

In this case, the process unit model equations, connection equations, and specifications are treated as constituting one very large system of nonlinear equations to be solved simultaneously. As discussed by Stadtherr and Hilton (1982a), physical property model equations may be included in this large equation system, or handled externally in subroutines. Also one may choose to handle the connection equations either explicitly or implicitly (Stadtherr and Hilton, 1982b). In any case, the central problem is now the solution of a very large and sparse system of nonlinear equations.

By solving most or all of the equations decribing a process simultaneously, one can avoid the drawbacks associated with the sequential-modular approach. When all equations are solved simultaneously there need be no nested iteration loops. Design specifications are now just very simple equations within the large system, and are almost trivial to handle. And since the simultaneous equations can be used as constraints in a generalized nonlinear programming problem, this approach has great potential for process optimization (Berna et al., 1980; Locke and Westerberg, 1982).

Today at least five prototype equation-based flowsheeting systems in various stages of development exist: SPEEDUP (Perkins and Sargent, 1982), ASCEND II (Benjamin et al., 1981; Locke, 1981; Locke and Westerberg, 1982), QUASILIN (Gorczynski et al., 1979), FLOWSIM (Shacham et al., 1982), and SEQUEL (Stadtherr and Hilton, 1982a). Many of the aspects involved in equation-based flowsheeting are discussed in detail by Westerberg et al. (1979), Shacham et al. (1982), and Stadtherr and Hilton (1982a).

In comparison to the sequential-modular approach, the equation-based approach offers much greater speed and flexibility, especially when dealing with complex processes and controlled simulations. These potential advantages are generally recognized today. However a number of problems have also been cited that have prevented the widespread industrial adoption of the equation-based approach.

Perhaps the most serious problem from a fundamental standpoint is that the equation-based approach has acquired a poor reputation for reliability. This is due in part to the problem of supplying a good initial guess, and to the need to use a general-purpose equation solver, as opposed to the special-purpose methods that can be applied within the computational modules of the sequential-modular approach. Some recent studies have produced promising results regarding this problem however. For instance, in his studies with ASCEND II, Locke (1981) uses automatic initialization routines along with a so-called evolutionary approach, and finds the initial guesses generated to be good enough to make Newton-Raphson a very reliable solution method.

Some other problems of a less fundamental nature are also cited in connection with the equation-based approach. Among these are large storage requirements and the complexity of the computing routines needed, such as the sparse matrix routines. The performance of the prototype programs listed above, together with new developments in sparse matrix methods for flowsheeting problems (Stadtherr and Wood, 1983a,b), indicate that problems of this type can be overcome. Another problem cited is that the very flexibility of the equation-based approach makes it easier for the user to make inconsistent specifications and makes error checking harder. These and other cited problems that can be categorized as a lack of friendliness to the user are in general not fundamental in nature and should disappear as better "software engineering" is applied to the development of equation-based packages. Finally it should be noted that at least part of the reluctance of industry to adopt this approach is that they have a considerable investment in sequential-modular software, and are unwilling to take the risk or expense of trying something completely different. Along these lines it is interesting to note that in recent years at least one equation-based simulation program, TISFLO-II at Dutch Mines, has been used industrially (Van Meulebrouk et al., 1982).

Simultaneous-Modular Approach

It should be noted first of all that use of the term "simultaneous-modular" to describe the approach discussed in this section has not been universal, Techniques using this sort of approach have been described by a number of terms, such as the two-tier method or simply as a "different" convergence concept for a sequential-modular simulator. Use of the term simultaneous-modular is fairly common today however, and is reasonably descriptive of the general type of computational strategy used.

As in the equation-based approach, in the simultaneous-modular approach the equations describing the entire process are solved simultaneously. In the equation-based approach all process variables, internal and external, are treated as independent. The external variables are those describing the input and output streams for each unit, as well as any equipment parameters to be varied in a controlled simulation or optimization problem. The remaining process variables, including those detailing the performance of particular units or any other nonexternal variables appearing in the individual process models, are referred to as internal. In the equation-based approach the actual rigorous model equations for each unit are then used to solve for the internal and external variables. On the other hand, in the simultaneous-modular approach only the external variables, or an appropriate subset thereof, are treated as independent, and simple, usually linear, models of the units are used, the coefficients in which are generated by perturbation of the same unit modules used in the conventional sequential-modular approach. Thus there are two levels of computation, a module level in which the modules are used, perhaps together with some connection equations, to generate an approximate Jacobian for the process, and a flowsheet level in which these equations are solved simultaneously together with the specification equations and some or all of the connection equations. The various techniques proposed for the simultaneous-modular approach differ in how the approximate Jacobian required on the flowsheet level is generated, in which numerical method is used to solve the nonlinear equations on the flowsheet level, and in whether all connecting streams are iterated on or only an appropriate set of tear streams. This latter case amounts to using some of the connection equations to reduce the size of the approximate Jacobian generated.

An early example of the simultaneous-modular technique is the approach of Rosen (1962), who uses simple linear split-fraction models of the unit operations to generate an approximate Jacobian for a system involving all connecting stream variables. An initial estimate of the split fractions is made, which provides an initial estimate of the Jacobian. The resulting linearized problem is then solved for the connecting stream variables, these results are used with the modules to generate a new set of split fractions, and the process is repeated until convergence. This approach did not meet with much success because the approximate Jacobian generated by split-fraction models is usually rather poor. Mahalec et al. (1979) adopt a strategy similar to that of Rosen. They demonstrate that split-fraction models are not adequate, and they suggest calculating an approximation of the Jacobian by difference split-fraction models and then updating the approximate Jacobian by Schubert's sparse update (Schubert, 1970). They reported good results with this approach. Various techniques employing an approximate Jacobian involving only tear stream variables have also been suggested (Mahalec et al., 1979; Sood et al., 1979a; McLane et al., 1979; Perkins, 1979; Metcalfe and Perkins, 1978). Sood et al. (1979b) and Sood and Reklaitis (1979) have shown this sort of approach to be very effective in solving linear flowsheeting problems. One may also put in this category the methods sometimes referred to as sequential-modular with a "different" convergence block, different in the sense that something other than the usual direct or accelerated substitution is used, typically a Newton or quasi-Newton approach.

To some extent the simultaneous-modular approach can be regarded as an attempt to combine some of the good features of both the sequential-modular approach and the equation-based approach. Since design specification equations can be handled directly on the flowsheet level, there is no need for costly control loops to converge the design specifications as required in the sequential-modular approach. And although there is still some nesting of iteration loops, the problem of slow convergence can be greatly reduced provided an efficient and reliable method is used to solve the flowsheet-level equations. Since the approximate Jacobian required on the flowsheet level will be smaller than that required by the equation-based approach, storage requirements are less, although, depending on the details of the approach used, sparse matrix methods may still be needed. And since already existing modules can be used to perform the module-level calculations, no great additional investment in software would be required.

The most fundamental problems with the simultaneous-modular approach involve the solution of the flowsheet-level equations. As in the equation-based approach, this requires a general-purpose nonlinear equation solver, typically the Newton-Raphson method or some variation thereof, and hence reliability can become a problem. Thus there is a need for good initialization procedures and a nonlinear equation solver with excellent global convergence properties. An additional problem is that the expense associated with computing the approximate Jacobian for the flowsheet level may be large, and could offset the gain in computational speed due to faster convergence. Thus there is a temptation to use very simple approximation methods; these however may provide a poor estimate of the Jacobian, which may exacerbate the problem of reliability. Problems such as these appear to have prevented the widespread adoption of the simultaneous-modular approach, particularly for simulation problems. These same concerns also apply in the case of controlled simulation problems. However, since by using a simultaneous-modular approach the need for comtrol loops can be eliminated, the tradeoffs tend to swing more clearly in favor of the simultaneous-modular approach. In part for this reason, some form of the simultaneous-modular approach, such as the use of quasi-Newton convergence blocks, is available as an option in some industrially used simulation programs (e.g., ASPEN).

Although the basic concept of the simultaneous-modular approach is not very new, this approach has attracted considerable recent interest, motivated by the increasing recognition of the inherent limitations of the sequential-modular approach. Recent studies, such as those by Mahalec et al. (1979) and Perkins (1979), have shown the simultaneous-modular approach to be a very promising alternative for process simulation and controlled simulation problems.

Process Optimization

As mentioned above, while process flowsheeting techniques are widely used in industry, process optimization is not. The reasons for this have been discussed by Westerberg (1981) and Blau (1981) and are summarized briefly here. The major problem, which underlies some of the others, is that using the typical sequential-modular approach process optimization can be computationally very expensive and time-consuming. The elapsed time required to obtain an optimized design may in fact be prohibitive if a plant is to be put on stream in timely fashion. It is often difficult in process optimization to choose a single objective function to optimize, since several competing objectives may have to be considered. Since optimizing a process for even one objective is expensive, it may not be possible to adequately consider other objectives. Another problem is that available process optimization tools typically require some specialized knowledge about how the optimization algorithm works, thus making process optimization difficult for the average engineer. Finally, there is the problem that industrial managers may not be willing to risk adopting a novel, optimized design when a proven, though nonoptimal design is available. It seems clear that considerable progress could be made toward eliminating these problems by developing a general-purpose process optimization strategy that is both very efficient and reliable, as well as easy to use.

The basic problem involved in solving a process optimization problem is the solution of a nonlinear programming problem. There is currently a revived interest in process optimization that has been stimulated by the improvement of nonlinear programming algorithms. In particular, a successive quadratic programming (SQP) algorithm developed by Wilson (1963), improved by Han (1976, 1977), and further improved and implemented by Powell (1978a,b,c) has been shown to be very effective in solving nonlinear programming problems. This algorithm has also been extended to solve large-scale nonlinear programming problems by Berna et al. (1980) and Locke et al. (1982).

In addition to the review articles mentioned above, Jirapongphan (1980) provides an extensive list of previous work in the field of process optimization. Basically, early attempts to apply optimization techniques to process design problems fall into one of two categories. In the first category, a chemical process is represented by a very simple model in order to reduce the computing time, e.g., Adelman and Stevens (1972). With such drastic simplification, the model cannot be expected to represent the actual process to any degree of accuracy. In the second category, an optimization loop is imposed on a sequential-modular simulator to take care of the extra degrees of freedom and design specifications (Friedman and Pinder, 1972; Gaines and Gaddy, 1976; Ballman and Gaddy, 1977). To perform a function evaluation for the optimization loop, the sequential modular simulator has to solve an entire simulation problem. The models employed in this approach are usually more realistic. However, as discussed above, this approach has not been widely accepted due to the

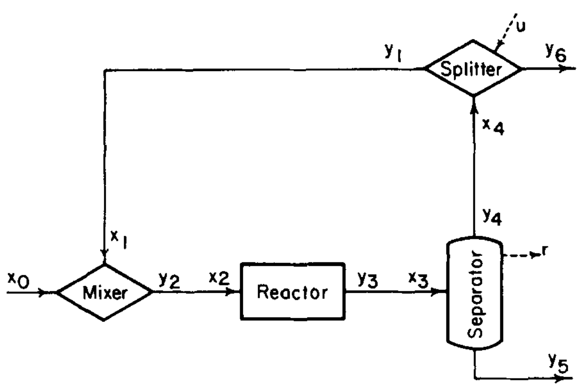


Figure 1. A four-unit process used as an example in describing different problem formulations.

large computing times needed. A typical industrial simulation may use one or more minutes of CPU time. When this is multiplied by the several hundred simulations required for an average optimization study, using this approach the cost may become prohibitive.

Recent work in process optimization has concentrated on the application of the Han-Powell SQP algorithm. Jirapongphan (1980) and Biegler and Hughes (1981, 1982a, 1983) have demonstrated the effectiveness of this method in connection with the simultaneous-modular approach. Techniques for applying the Han-Powell method in connection with the equation-based approach have been described by Berna et al. (1980) and Locke and Westerberg (1982).

As noted above, several recent studies have demonstrated the potential of the simultaneous-modular approach for solving both flowsheeting and optimization problems. These studies tend to be fairly limited in scope however, and leave a number of questions unanswered. Our intention in the work presented here and in Parts II and III is to provide a comprehensive study and critical evaluation of the simultaneous-modular approach for process flowsheeting and optimization, and of the different numerical techniques that may be used to implement it.

FORMULATION OF PROBLEM

It should first be noted that for simulation and controlled simulation problems the flowsheet is assumed to have been partitioned into irreducible blocks of units, a procedure for which is discussed in some detail below. The blocks are then solved one at a time in the appropriate precendence order. A simultaneous-modular formulation is applied only within the irreducible blocks containing more than one unit (i.e., the cyclic parts of the flowsheet). Thus in the discussion of problem formulation here and Jacobian evaluation below, it should be remembered that the problem being formulated and the Jacobian being evaluated describe only one irreducible block and not necessarily the entire flowsheet.

There are a number of possible problem formulations for the simultaneous-modular approach. In general these formulations can be put into one of three categories. In this section we describe these three basic formulations, and discuss some of their advantages and disadvantages. To facilitate the discussion, each formulation will be described with respect to the simple flowsheet shown in Figure 1. This flowsheet comprises four functional units, a mixer, a reactor, a separator, and a splitter. It is also assumed that the splitter has one equipment parameter u that must be adjusted to meet one design specification r imposed on the output of the separator.

Formulation I

In this formulation, all connecting streams are torn and treated as two separate streams, one input and one output stream. The formulation of Mahelec et al. (1979) falls into this category. Using a simultaneous-modular approach the flowsheet-level equations describing the process consist of model equations or approximations thereof, stream connection equations, and design specifications:

Model equations

$$y_2 = g_2(x_1) \qquad (1a)$$

$$y_3 = g_3(x_2) \qquad (1b)$$

$$y_4 = g_4(x_3) \qquad (1c)$$

$$y_1 = g_1(x_4,u) \qquad (1d)$$

Stream connection equations

$$x_2 = y_2 \qquad (1e)$$

$$x_3 = y_3 \qquad (1f)$$

$$x_4 = y_4 \qquad (1g)$$

$$x_1 = y_1 \qquad (1h)$$

Design specification

$$r(x_3) = r^s \qquad (1i)$$

Here r represents an output variable whose value is specified to be r^s. Using this formulation, a general simulation or controlled

$$\begin{array}{c|ccccccccc}
 & y_2 & x_2 & y_3 & x_3 & y_4 & x_4 & y_1 & x_1 & u \\
\hline
1a & 1 & & & & & & & -A_{21} & \\
1e & -1 & 1 & & & & & & & \\
1b & & -A_{32} & 1 & & & & & & \\
1f & & & -1 & 1 & & & & & \\
1c & & & & -A_{43} & 1 & & & & \\
1g & & & & & -1 & 1 & & & \\
1d & & & & & & -A_{14} & 1 & & -A_{1u} \\
1h & & & & & & & -1 & 1 & \\
1i & & & & -B_3 & & & & &
\end{array}$$

Figure 2. Jacobian matrix for formulation I.

$$\begin{array}{c|ccccc}
 & x_2 & x_3 & x_4 & x_1 & u \\
\hline
2a & I & & & -A_{21} & \\
2b & -A_{32} & I & & & \\
2c & & -A_{43} & I & & \\
2d & & & -A_{14} & I & -A_{1u} \\
2e & & -B_3 & & &
\end{array}$$

Figure 3. Jacobian matrix for formulation II.

simulation problem can be represented by $2(c + 2)n_c + n_d$ nonlinear equations, where c, n_c and n_d are respectively the number of chemical species, connecting streams, and design specifications. The expression $c + 2$ occurs because each stream can be characterized by c component flowrates, the enthalpy flowrate, and the pressure. In the specific formulation used by Mahalec et al., nonconnecting input or output stream variables are also treated as independent.

When the variables and equations are reordered appropriately, the Jacobian required to solve Eqs. 1a–i has the bordered unit lower triangular form shown in Figure 2. The A and B submatrices that appear in Figure 2 are matrices of partial derivatives of output variables of modules with respect to input variables. The calculation of these submatrices will be discussed below.

Formulation II

In formulation I, each connecting stream is treated as two separate streams, which results in an unnecessarily large system of nonlinear equations. In formulation II, each connecting stream is treated as one input stream and the model equations are substituted into the stream connection equations. The resulting system of nonlinear equations is given below:

Stream connection equations

$$x_2 = g_2(x_1) \tag{2a}$$

$$x_3 = g_3(x_2) \tag{2b}$$

$$x_4 = g_4(x_3) \tag{2c}$$

$$x_1 = g_1(x_4,u) \tag{2d}$$

Design specification

$$r(x_3) = r^s \tag{2e}$$

Using this formulation, a general simulation or controlled simulation problem can be represented by $(c + 2)n_c + n_d$ nonlinear equations, so the number of flowsheet-level equations is reduced by almost 50%, compared to formulation I. The Jacobian of Eqs. 2a–e is shown in Figure 3. Note that the A and B submatrices to be calculated are the same as those in the first formulation.

Formulation III

For typical applications, the number of equations generated using formulation II can be still on the order of 1,000. Such systems are still too large to be solved by full matrix techniques. To further reduce the number of nonlinear equations that must be solved simultaneously, an appropriate subset of connecting streams are torn instead of all connecting streams. Streams that are not torn can be eliminated using corresponding stream connection equations (e.g., Perkins, 1979). For the simple flowsheet we are considering, stream x_1 can be torn and streams x_2, x_3, and x_4 can be eliminated using Eqs. 2a–c. The resulting system of nonlinear equations is:

Stream connection equations

$$x_1 = g_1(g_4\{g_3[g_2(x_1)]\},u) = \bar{g}_1(x_1,u) \tag{3a}$$

Design specification

$$r\{g_3[g_2(x_1)]\} = \bar{r}(x_1) = r^s \tag{3b}$$

Using this formulation, a general simulation or controlled simulation problem can be represented by $(c + 2)n_t + n_d$ nonlinear equations, where n_t is the number of streams torn in the current irreducible block. Since in most applications, n_t is relatively small, Eqs. 3a–b can be solved by full matrix techniques. The Jacobian for this formulation is usually a full matrix. As noted above, a variety of techniques employing a Jacobian involving only tear stream variables have been suggested, and the use of quasi-Newton convergence blocks within existing sequential-modular packages such as ASPEN can also be thought of as in this category.

Note that in all three formulations, a simulation or controlled simulation problem is described by a system of nonlinear equations $f(x) = 0$, and the evaluation of $f(x)$ requires one pass through all unit modules. It is interesting to note that if we carry the reduction one step further, i.e., we use tear stream connection equations to eliminate tear stream variables, then we get the conventional sequential-modular approach. For example, if Eq. 3a is used to eliminate x_1 then we will have only one equation in one variable left. The problem with this approach is that each evaluation of the design specification equation requires a complete simulation to determine the stream variables.

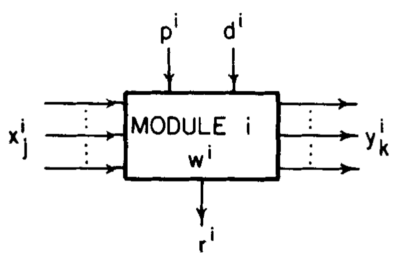
Since for formulation III, the storage requirements are relatively small and full matrix methods can be used, the computational requirements for this formulation are much more closely in line with the usual sequential-modular approach than either of the other two basic formulations. SIMMOD, our implementation of the simultaneous-modular approach, is based on formulation III.

JACOBIAN EVALUATION

The performance of the simultaneous-modular approach depends to a considerable extent on how the Jacobian that is required in the solution of the flowsheet-level equations, or an approximation of it, is calculated. If the computation involved is excessive, the simultaneous-modular approach may be inefficient and not competitive with the sequential-modular approach. Also, if a poor approximation of the Jacobian is used, the simultaneous-modular approach may fail or at least require many iterations to solve the problem, thus again becoming inefficient and perhaps not competitive.

For formulations I and II, several submatrices of partial derivatives of output variables of modules with respect to input variables must be determined, as shown in Figures 2 and 3. Figure 4 shows the information flow of an arbitrary module. The equations describing the module are solved to determine output streams, given input streams and equipment parameters. Besides computing output streams, the module also generates other internal results during the solution process. Some of these results are saved in a special area, the retention vector, and can be manipulated. Other results for the internal variables are usually discarded after the module calculation is done.

The partial derivatives, or approximations of them, that are needed to solve the flowsheet-level equations are the partial derivatives of the connecting output streams and some variables in the retention vector with respect to the connecting input streams and free equipment parameters. There are many ways to



x^i_j	Stream vector for j^th input
p^i	Fixed equipment parameter vector
d^i	Free equipment parameter vector
w^i	Internal variable vector
r^i	Retension variable vector
y^i_k	Stream vector for the k^th output

Figure 4. Information flow structure for an arbitrary module.

calculate or approximate these partial derivatives for formulations I and II. Here, we consider three of them: 1. Analytic differentiation; 2. Full-block perturbation; 3. Diagonal-block perturbation.

Analytic Differentiation

Considering the complexity of the modules, it seems impractical to determine the linearized input-output relations analytically. However, if all the equations describing the module are solved by simultaneous linearization, then it is possible to determine these relations analytically without great difficulty. Note that because the module calculates internal variables, retention variables, and output stream variables, given values of input stream variables and equipment parameters, the equations describing the module can be written as $\boldsymbol{h}(z_1,z_2) = \boldsymbol{0}$, where the vector z_1 includes internal variables, retention variables, and output stream variables, and the vector z_2 includes input stream variables and equipment parameters. Moreover, the number of equations in $\boldsymbol{h}(z_1,z_2) = \boldsymbol{0}$ is the same as the number of variables in z_1. By using Euler's chain relation

$$\left[\frac{\partial z_1}{\partial z_2}\right]_h = -\left[\frac{\partial h}{\partial z_1}\right]_{z_2}^{-1}\left[\frac{\partial h}{\partial z_2}\right]_{z_1}$$

all the necessary partial derivatives can be determined without repeatedly perturbing z_2 and solving the system of nonlinear equations $\boldsymbol{h}(z_1,z_2) = \boldsymbol{0}$ for z_1. This is because the matrices $(\partial \boldsymbol{h}/\partial z_1)_{z_2}$ and $(\partial \boldsymbol{h}/\partial z_2)_{z_1}$ are readily available, if the equations describing the module are solved by simultaneous linearization. This is probably the most suitable technique for evaluating the partial derivatives needed for formulations I and II. However, because most modules, including those in SIMMOD, do not solve nonlinear equations by simultaneous linearization, this approach is not yet practical.

Full-Block Perturbation

Another technique for evaluating the partial derivatives for formulations I and II is to calculate them using difference approximation. This requires a module calculation after each unknown input variable is perturbed by a small amount. There are $(c + 2)n_{ci} + n_{di}$ unknown input variables for module i, where n_{ci} is the number of connecting input streams to module i, and n_{di} is the number of free equipment parameters to module i. Thus, to calculate a Jacobian, the module i calculation must be performed $(c + 2)n_{ci} + n_{di} + 1$ times. When performing the finite difference approximations it is of course important that appropriate care be taken to prevent problems such as round-off or truncation error from causing poor derivative estimations. A good practical discussion of such matters has been provided by Himmelblau and Lindsay (1980).

Diagonal-Block Perturbation

The full-block perturbation technique requires many module calculations to determine the partial derivatives. Mahelec et al. (1979) propose using approximate values for these derivatives to reduce the number of module calculations. More specifically, they used the following approximation

$$\frac{\partial y_k}{\partial x_j} = D_{kj}$$

where y_k is the stream vector of the kth output, x_j is the stream vector of the jth input, and D_{kj} is a diagonal matrix. They perturb all elements of the stream vector x_j at the same time by Δx_j, perform the module calculation to determine Δy_k, the change of the stream vector y_k, and then determine the mm element of D_{kj} from

$$(D_{kj})_{mm} = \frac{(\Delta y_k)_m}{(\Delta x_j)_m} .$$

To evaluate a Jacobian using this technique, the module i calculation must be performed $n_{ci} + n_{di} + 1$ times. Note that in the diagonal approximation, it is assumed that the mth elements of output stream vectors are only affected by the mth elements of input stream vectors. While this assumption may be reasonable for component flowrates and stream pressures, neglecting the interactions between these variables and enthalpy flowrates may result in a very poor derivative approximation.

Direct Difference Approximation

We now turn our attention to Jacobian evaluation methods for formulation III. As we mentioned earlier, using formulation III, a chemical process is represented by a relatively small system of nonlinear equations, and the Jacobian of these equations usually has very few zeros in it. Calculating the required Jacobian directly by difference approximation is the most straightforward approach (e.g., Perkins, 1979). To evaluate a Jacobian using this approach, it is necessary to perform each module calculation at most $(c + 2)n_t + n_d + 1$ times, where n_t is the number of tear streams, n_d is the total number of free equipment parameters, and both n_t and n_d are for the current irreducible block. The actual number of module calculations required may be somewhat less than this figure depending on where in the loop the free variables occur. Compared to formulations I and II, formulation III typically requires considerably less computer storage. However, while the direct difference approximation is a very straightforward approach to obtaining the Jacobian, it may be less efficient than the block perturbation techniques used for formulations I and II, for the following reasons.

1. When the chemical process to be simulated becomes larger or becomes more complicated, the number of tear streams will generally increase. This means that the number of module calculations will increase. However, as we noted earlier, if the Jacobian is calculated by block perturbation techniques, then the number of module calculations will be proportional to the number of connecting input streams for each individual module. Thus since the number of connecting input streams for a given module will remain about the same even when the process becomes larger or

more complex, block perturbation methods may be advantageous.

2. When block perturbation techniques are used to calculate the Jacobian, all the calculations for a given module are performed consecutively. This means that it is easier to temporarily retain the internal variables of the given module and utilize these values to reduce the CPU time for subsequent calculations of the same module.

3. As the development of simultaneous-modular systems continues, it is quite possible that derivatives for some time-consuming modules will be determined analytically, as discussed above. This option can be easily included when block perturbation techniques are used.

Based on the above considerations, we believe that it is desirable when using formulation III to calculate the Jacobian using one of the block perturbation methods usually associated with formulations I and II. This is possible because the only difference between formulations II and III is that in formulation III some equations from formulation II are used to eliminate some of the variables from formulation II. Mathematically, the equations in the second formulation can be partitioned as:

$$f(p,q) = 0 \qquad (4a)$$

$$g(p,q) = 0. \qquad (4b)$$

Here Eq. 4a represents the nontear connecting stream equations, corresponding to Eqs. 2a–c above, and Eq. 4b represents the tear stream connection equations and design specifications, corresponding to Eqs. 2d–e above. The vector q represents tear stream variables and free variables, and p represents nontear stream variables. To obtain formulation III, Eq. 4a is used to determine p as a function of q to reduce the above equations to $g(p,q) = g(p(q),q) = \bar{g}(q)$, corresponding to Eq. 3a above. Using Euler's chain relation, we see that the Jacobian for formulation III can be obtained from the Jacobian for formulation II:

$$\left[\frac{d\bar{g}}{dq}\right] = \left[\frac{\partial g}{\partial q}\right] - \left[\frac{\partial g}{\partial p}\right]\left[\frac{\partial f}{\partial p}\right]^{-1}\left[\frac{\partial f}{\partial q}\right]. \qquad (5)$$

Also note that if the nontear streams are reordered according to the order in which they are calculated in the sequential-modular approach, then the submatrix $(\partial f/\partial p)$ has a unit lower triangular form. This can easily be seen in Figure 3, by referring to the submatrix corresponding to Eqs. 2a–c and stream vectors x_2, x_3, and x_4. Therefore, since this unit lower triangular matrix will be easy to invert, it is fairly easy to compute the Jacobian required for formulation III, given the Jacobian for formulation II.

As noted above, SIMMOD, our implementation of the simultaneous-modular approach, uses formulation III. SIMMOD provides three options for generating the Jacobian approximation whenever required by the nonlinear equation solver used. These options are full-block perturbation, diagonal-block perturbation, and direct difference approximation. Note that while the last option can be applied directly in connection with formulation III, the two block perturbation options require the use of Eq. 5. The relative merits of these three options for evaluating the Jacobian will be considered in detail in Part II.

PARTITIONING AND TEARING

The use of formulation III means that some provision must be made for partitioning and tearing the flowsheet. To partition a flowsheet is to find groups of modules which can be solved independently together. These groups, the irreducible blocks, should each contain the fewest number of modules possible. Typically, partitioning algorithms also perform precedence ordering, i.e., they also determine the proper sequence in which to perform computation on these groups. Tearing a group of units is to determine a set of streams, the tear set, such that once these streams are torn, all the remaining streams related to the group can be computed sequentially. This means that all the remaining nontear streams can be considered as functions of the tear streams. An extensive list of past work in this area can be found in Hlavacek (1977).

Previous work in the area of flowsheet partitioning and tearing has been based on the sequential-modular approach. For the simultaneous-modular approach, the situation is somewhat different because of the presence of free variables and design specifications. In the sequential-modular approach, design specifications and free variables are handled by user-specified control loops and control blocks, which are in effect added to the flowsheet prior to partitioning and tearing. In the simultaneous-modular approach, control loops are not used, so some other method must be devised to account for design specifications and free variables in partitioning and tearing. In devising an algorithm for partitioning and tearing a flowsheet with free variables and design specifications, the following considerations are important:

1. The process flowsheet can be regarded as a directed graph, so one can think of each combination of design specification and free variable as providing a directed information stream. This means that conceptually each design specification will be used to solve for a free variable. However, since explicit control loops are not used in the simultaneous-modular approach, it is not known a priori where these directed information flows are, i.e., it is not known what the proper combinations of design specifications and free variables are.

2. All the information streams arising from the design specifications must be torn because design specifications do not directly generate new values for free variables.

3. In the simultaneous-modular approach, it does not matter which equation is solved for which free variable, as long as the free variable has an effect on the design specification. This is because in the simultaneous-modular approach all the design specifications and stream connection equations in the same irreducible block are converged simultaneously.

In this section we describe an algorithm for partitioning and tearing flowsheets with free variables and design specifications. To facilitate the discussion, we first outline the algorithm and then describe each step in some detail.

Algorithm

1. Partition the flowsheet into groups of modules while ignoring free variables and design specifications.
2. Tear each group of modules while ignoring free variables and design specifications.
3. Determine information streams corresponding to free variables and design specifications.
4. With information streams included, partition the group flowsheet into irreducible blocks.

Step 1: Partition the flowsheet while ignoring design specifications. Partitioning a flowsheet is equivalent to finding the strongly connected components of a directed graph (Tarjan, 1972), or to finding a symmetric permutation to put a matrix into block lower triangular form (Duff and Reid, 1978a). The most efficient algorithm for performing this step is that of Tarjan (1972). A detailed description of this algorithm can be found in Duff and Reid (1978a), and a FORTRAN implementation of this algorithm can be found in Duff and Reid (1978b). Also note that Tarjan's algorithm automatically determines a group precedence order.

Step 2: Tear each group of modules. Many algorithms have been proposed to find a tear set for a group of modules. Lists of some of the algorithms can be found in Hlavacek (1977) and Westerberg et al. (1979). In our implementation we use the

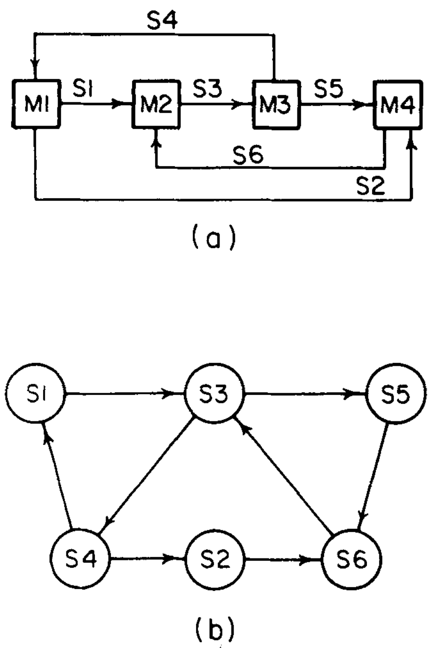


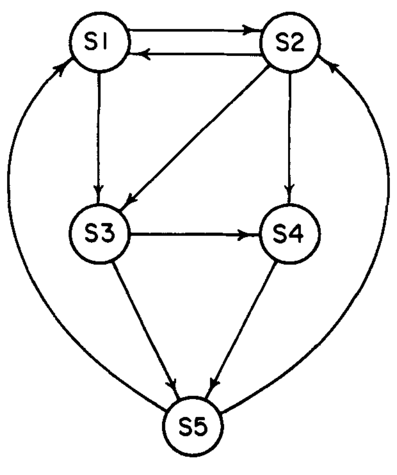
Figure 5. (a) Flowsheet diagram; (b) corresponding signal flow diagram.

algorithm of Pho and Lapidus (1973) to determine the tear set. Given an arbitrary weighting factor for each stream, the algorithm will determine a tear set with the minimum weighted sum. In our implementation, the weighting factor for stream i is determined from

$$w_i = 5 + \frac{5 n_s c_i}{\sum_{i=1}^{n_s} c_i}$$

where n_s is the number of connecting streams associated with the group, and c_i is the number of loops torn by stream i. Without the second term in w_i the algorithm will determine a tear set with the minimum number of tears. Without the first term the algorithm will determine a tear set with the minimum number of loops torn. Upadhye and Grens (1975) show that tear sets that minimize the total number of loops torn give the best convergence for direct substitution. Because SIMMOD uses direct substitution iterations to generate initial guesses for tear stream variables, it is desirable to take this into account. However, this criterion alone sometimes produces tear sets with many tear streams, which is undesirable because the computer storage required by formulation III increases with the number of tear streams. The weighting factor we used is an attempt to compromise between these two objectives.

The algorithm of Pho and Lapidus operates on a signal flow diagram. Figure 5a shows a simple flowsheet diagram, and Figure 5b shows the corresponding signal flow diagram. In the flowsheet diagram, nodes represent modules and directed edges represent streams. In the signal flow diagram, nodes represent streams and directed edges represent information flows. A directed edge from node i to node j in the signal flow diagram indicates that the values of stream i are needed to determine the values of stream j. The algorithm of Pho and Lapidus first applies a basic tearing algorithm that simplifies the signal flow diagram by eliminating streams that can be excluded from the optimal tear set (ineligible streams), and by eliminating streams that must be included in the optimal tear set (essential streams). When this basic algorithm fails to determine a complete tear set, a two-way



Stream	SI	S2	S3	S4	S5
Weighting factor	3	3	4	4	5

Figure 6. Counterexample of two-way edge reduction procedure of Pho and Lapidus (1973).

edge reduction procedure and a branch and bound technique are used to ensure optimality. Streams i and j are said to form a two-way edge if there are edges directed from i to j and from j to i. Pho and Lapidus make the false claim that if streams i and j form a two-way edge then either stream i or stream j must be in the optimal tear set, but not both. Figure 6 shows a counterexample. In this signal flow diagram, a minimum of two streams must be torn. S1 and S2 form a two-way edge, but the optimal tear set includes both S1 and S2. For this reason, only the branch and bound technique of Pho and Lapidus is used to ensure optimality.

After identifying a tear set, a precedence order of modules within the group can easily be determined. In our implementation, we simply disable streams in the tear set, and use Tarjan's algorithm to determine a precedence order of modules.

Step 3. Determine information streams. The basic idea in determining information streams is to conceptually assign a free variable to each design specification. To reduce computational requirements, we make as many such assignments as possible within each group of modules. Because of this, only the excess free variables or design specifications of each group need be considered. To determine information streams, we perform the following operations:

3a. Determine the group flowsheet and number groups according to the group precedence order determined in Step 1. This can be done by examining each stream. If a stream is from a module in group i to a module in group j then a directed edge from group i to group j is included in the group flowsheet.

3b. Count the number of free variables and design specifications associated with each group of modules. Note that a design specification imposed on a stream directed from group i to group j belongs to group i. Figure 7a shows a sample group flowsheet after the first two operations. In this figure, the notation (3*F*, 2*S*) means that there are three free variables and two design specifications associated with that group.

3c. Determine the number of excess free variables or excess design specifications associated with each group, and number these variables and specifications. The results for the sample problem are shown in Figure 7b.

3d. Determine the group adjacency matrix. The (j,i) element of the group adjacency matrix equals one if and only if there is a directed edge from group i to group j. Figure 8a shows the group adjacency matrix for the sample problem.

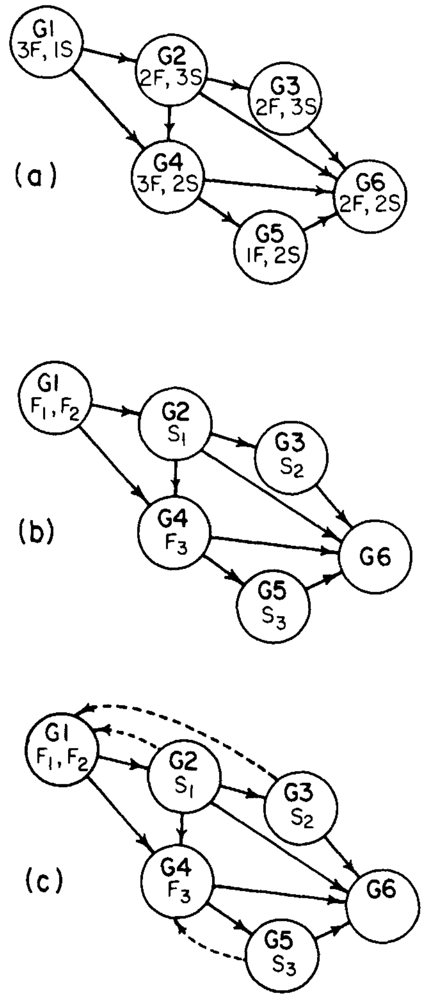


Figure 7. (a) Group flowsheet diagram showing total number of free variables and design specifications for each group; (b) group flowsheet showing excess free variables and design specifications for each group; (c) group flowsheet showing information streams identified using algorithm described in text, and corresponding to the output set assignment shown in Fig. 8c.

	G1	G2	G3	G4	G5	G6
G1						
G2	1					
G3		1				
G4	1	1				
G5				1		
G6		1	1	1	1	

(a)

	G1	G2	G3	G4	G5	G6
G1						
G2	1					
G3	1	1				
G4	1	1				
G5	1	1		1		
G6	1	1	1	1	1	

(b)

	F_1	F_2	F_3
S_1	$\boxed{1}$	1	
S_2	1	$\boxed{1}$	
S_3	1	1	$\boxed{1}$

(c)

Figure 8. (a) Group adjacency matrix for group flowsheet in Fig. 7b; (b) corresponding group reachability matrix; (c) corresponding occurrence matrix showing which excess design specifications are functions of which excess free variables, and indicating the output set assignment.

3e. Determine the group reachability matrix. To determine the jth row of the reachability matrix we start with the jth row of the adjacency matrix, and then take the Boolean sum of this row of the adjacency matrix and all rows i in the adjacency matrix for which there is a one in the (j,i) position of the adjacency matrix. Figure 8b shows the reachability matrix for the sample group flowsheet. Note that a one at the (j,i) position of the reachability matrix means that the results of group j are directly or indirectly affected by results of group i because of stream connections.

3f. Determine functional relations between excess free variables and excess design specifications. Excess design specifications associated with group j are affected by excess free variables associated with groups that have a one in the jth row of the group reachability matrix. Figure 8c shows the results for the sample problem in the form of an occurrence matrix indicating which excess design specifications are functions of which excess free variables.

3g. Determine an output set for the excess specifications and thus identify the information streams. From the occurrence matrix found in Step 3f, we can determine an output set for the excess specifications, i.e., we can determine which excess specification is to be conceptually solved for which excess free variable. If excess specification S_j is to be solved for excess free variable F_i, then an information stream directed from the group that contains S_j to the group that contains F_i is added to the group flowsheet. If an output set cannot be found, the run should be terminated because the free variables are not able to "control" the specifications. Duff (1981a) compares several algorithms for determining an output set and also (Duff, 1981b) gives a FORTRAN routine for performing the operation. For the sample problem, an output set is indicated in Figure 8c, and the corresponding information streams are shown in Figure 7c as dashed edges.

Step 4. Partition group flowsheet with information streams included. This step is similar to Step 1 except that the original flowsheet is now replaced by the group flowsheet with information streams included. For the sample group flowsheet, the irreducible blocks are $\{G1,G2,G3\}$, $\{G4,G5\}$, and $\{G6\}$. The tear set for $\{G1,G2,G3\}$ is simply the union of tear sets for $G1$, $G2$, and $G3$, as found in Step 2.

PROCESS OPTIMIZATION PROBLEMS

Perhaps the most straightforward approach to a process design problem is to formulate it as an optimization problem (Westerberg, 1981). This allows any extra degrees of freedom to be utilized to improve the performance of the process instead of being arbitrarily fixed by the designer. It also allows design specifications to be inequalities as well as equalities. The presence of an objective function to take up extra degrees of freedom and the presence of inequality constraints are the two major differences distinguishing an optimization problem from a simulation or controlled simulation problem. Though there are advantages in formulating a design problem as an optimization problem, it should also be noted that much more information is required to perform a realistic optimization study, not to mention the fact that an optimization problem may be considerably more difficult to solve than a simulation or controlled simulation problem.

One of the most significant advantages of the simultaneous-modular approach for simulation and controlled simulation problems is that it can be much more readily extended to the solution of process optimization problems than the sequential-modular approach. Using the simultaneous-modular approach, the process optimization problem can be stated as a general nonlinear programming (NLP) problem in the following form:

$$\min_{x} F(x)$$
$$\text{s.t.: } g(x) = 0$$
$$h(x) \geqslant 0.$$

The equality constraint $g(x) = 0$ is the same as the set of flowsheet-level equations used to solve a simulation or controlled simulation problem. Thus to make the extension to process optimization, one need only supply an objective function $F(x)$ and any inequality constraints that may be present. The same three types of problem formulations described above for simulation and controlled simulation problems can be used for the optimization problem, as can the techniques described for obtaining the Jacobian. When formulation III is used with block perturbation to compute the derivatives, the Euler relation can again be used to obtain the derivatives of the objective function and the inequality constraints with respect to the tear variables and free variables only. Different methods for using the simultaneous-modular approach for process optimization have been described recently by Jirapongphan (1980) and Biegler and Hughes (1981, 1982a, 1983), all using the Han-Powell SQP algorithm to solve the NLP problems.

The method of Jirapongphan can be classified as using formulation I. The quadratic/linear approximation (Q/LAP) method of Biegler and Hughes (1981) is in one sense a type of sequential-modular approach, since it requires a converged simulation for each function evaluation. One the other hand, these two methods are actually rather similar in principle, differing significantly only in whether to handle the stream connection equations inside or outside the NLP routine. Jirapongphan passes all the stream connection equations to the nonlinear programming routine, so he has to solve a very large nonlinear programming problem. In this case each function evaluation requires only one pass through each module, and the stream connection equations will not be satisfied until the final optimal solution is reached. Thus this is said to be an "infeasible path" method. In the quadratic/linear approximation method, Biegler and Hughes eliminate all the stream variables outside the NLP routine, so they solve a very small NLP problem. However, each function evaluation now requires a complete simulation. Since the stream connection equations are now satisfied at every iteration, this is said to be a "feasible path" method.

The IPOSEQ, RFV, and CFV methods of Biegler and Hughes (1982a, 1983) can be thought of as using formulation III. Again the most significant difference is that IPOSEQ follows an infeasible path, thus requiring only one pass through each module for a function evaluation, while RFV and CFV follow a feasible path, thus requiring a complete simulation for each function evaluation. All three methods obtain the Jacobian by direct difference approximation. While one might expect the need for a complete simulation for every function evaluation to make the feasible path methods less efficient, the results on one problem (Biegler and Hughes, 1982b), suggest otherwise. However, SIMMOD uses an infeasible path approach, and in our experience on several test problems is extremely efficient, as detailed in Part III.

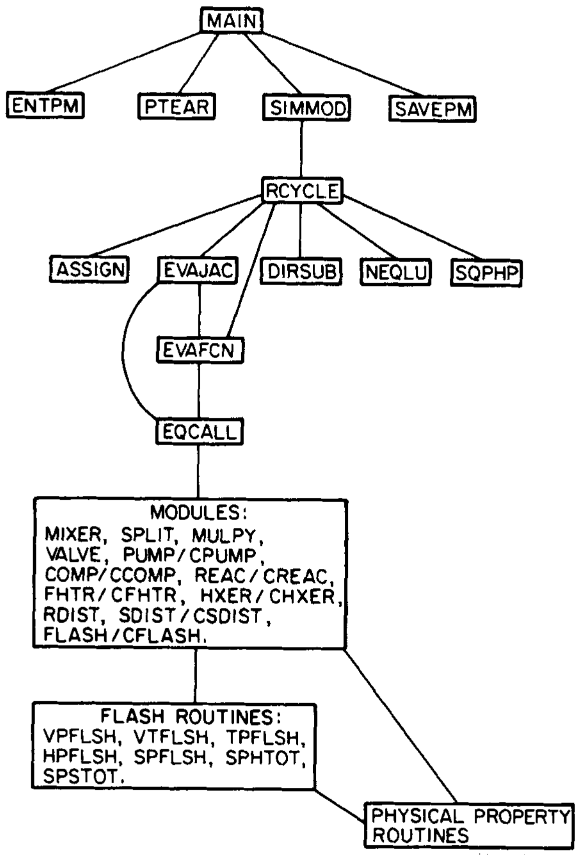


Figure 9. Structure of SIMMOD, a simultaneous-modular flowsheeting and optimization package.

IMPLEMENTATION OF SIMMOD

To perform a critical evaluation of the simultaneous-modular approach, a simultaneous-modular flowsheeting and optimization system, SIMMOD, has been developed. From the user's point of view, the system is similar to the conventional sequential-modular systems. However, the system can handle controlled process simulation problems as efficiently as process simulation problems, and it can also solve process optimization problems very efficiently. The basic structure of SIMMOD is shown in Figure 9. A brief description of some of the blocks in this figure is given.

ENTPM—Read in User-Supplied Data

The input phase is the most critical interface between the user and the flowsheeting program and therefore deserves considerable attention. To use a flowsheeting program, the user has to provide the following data:

1. Process topology.
2. Equipment parameters for the modules.
3. Feed stream information.
4. Physical property data not in the data bank.
5. Convergence criteria.
6. Cost parameters.
7. Optimization criteria.

Note that items 6 and 7 are required only when economic evaluation or optimization is being carried out. The required information may be entered in a variety of ways (Motard et al., 1975). In SIMMOD this information is entered using a problem oriented language (POL) in a logical order. Three sample input files for SIMMOD, one for each of the three types of problems being considered, simulation, controlled simulation, and optimization, are given in the subsequent papers in this series.

To show the simplicity of including design specifications, we consider here the MODEL statement for entering data items 1, 2, and 6. The general format of the MODEL statement is as follows:

MODEL equipment name BY library module name
IN = input streams
OUT = output streams
PARA = equipment parameters
CPAR = cost parameters
FREE = free equipment parameters
SPEC = design specifications

To treat an equipment parameter as a free variable, the user simply specifies the position of the parameter in the equipment parameter list, a scaling factor, a lower bound, and an upper bound for that parameter, or he may use default values for the last three pieces of data. To impose a design specification, the user simply specifies the output variable to be controlled, the constraint type, and the specified value. In the controlled simulation case the only constraint type allowed is equality, while in the optimization case inequality constraints are also allowed. Note that similar options are also provided for stream variables.

PTEAR—Partition and Tear a Flowsheet

Using the information supplied in the MODEL statements, PTEAR performs partitioning and tearing to determine a computational sequence of the modules. The user can also specify a partial tear set through a TEAR-STREAM statement. For optimization, we treat the whole flowsheet as one irreducible block and only perform Steps 1 and 2 of the algorithm presented above. This is done because of the presence of an objective function and inequality constraints.

SAVEPM—Save Final Results

At the end of a run, SAVEPM outputs the final results to a file. These results are written in the POL, so the user can easily modify the file and use it for later runs.

SIMMOD—Calculation Phase

SIMMOD first calls each module to perform data checking. If fatal errors are detected the run is terminated. Otherwise, SIMMOD calls RCYCLE to solve each irreducible block sequentially.

RCYCLE—Recycle Calculation

RCYCLE performs the calculation for one irreducible block. If the irreducible block contains only one module and there are no design specifications, RCYCLE simply calls the module to perform the calculation and then returns to SIMMOD. Otherwise, RCYCLE performs the following calculations:

1. Number variables and equations consecutively. This is done by calling ASSIGN.
2. Perform a small number of direct substitution iterations (DIRSUB) to initialize the tear stream variables. During these iterations, free variables and design specifications are ignored. Also, when the irreducible block consists of several groups of modules, all the direct substitution iterations on one group of modules are performed before proceeding to the next group.
3. Perform simultaneous-modular iterations. In the simulation or controlled simulation case, this is done by calling NEQLU to solve the flowsheet-level system of nonlinear equations. In the optimization case, this is done by calling SQPHP to solve the nonlinear programming problem. The functions and derivatives needed in NEQLU or SQPHP are calculated in EVAFCN and EVAJAC respectively.

MODULES—Perform Unit Operation Calculations

The modules in SIMMOD are similar to those in sequential-modular systems. Each module computes the output streams and retention variables, given input streams and equipment parameters. Some comments about the module calculations are in order:

1. Each module consists of three parts: data checking, module calculation, and report generating.
2. In SIMMOD each stream vector carries the following information: c component flowrates, enthalpy flowrate, pressure, temperature, vapor fraction, and total flowrate. Since only the first $c + 2$ elements are treated as independent variables, each module is only required to determine these elements of the output stream vectors. A flag is used to indicate whether the remaining elements of a particular stream vector have been calculated or not. This allows the system to avoid performing some unnecessary adiabatic flash calculations.
3. Each module is required to calculate the output streams or variables accurate to a relative tolerance specified by the user or set by default in SIMMOD.
4. A flag is used to indicate that the system is performing block perturbation to calculate the Jacobian, so that in this case the module can use internal variable values generated from a previous calculation and retained for subsequent calculations of the same module.
5. Modules are available for the following units: mixer (MIXER), splitter (SPLIT), valve (VALVE), pump (PUMP), compressor (COMP), percent conversion reactor (REAC), fired heater (FHTR), heat exchanger (HXER), rigorous distillation (RDIST), simple distillation (SDIST), and flash (FLASH). A stream multiplication module (MULPY) is available for combining streams from multiple processing trains. The user may write his own modules, ADD1-ADD10, to perform special calculations.

COST MODULES—Perform Unit Costing

The cost modules that are currently available in SIMMOD are shown in Figure 9; for instance, CPUMP is the cost module associated with the PUMP module. These modules perform sizing and costing to determine base costs, fixed investment costs, and utility costs. The cost parameters all have default values but can also be specified by the user in the MODEL statement as described above. Some options are also included to allow the user to tune the cost modules.

Physical Property Routines

SIMMOD is designed so that the physical property routines can be easily replaced. In current implementation, all the thermodynamic properties are calculated using the Peng-Robinson (1976) equation of state.

NEQLU—Nonlinear Equation Solver (flowsheet level)

NEQLU implements a modification of Powell's hybrid (or dogleg) method (Powell, 1970). The modified Powell method used is described in detail by Chen and Stadtherr (1981), who also present performance data for NEQLU on several standard mathematical test problems. These results indicate that NEQLU is more efficient and reliable than the code HYBRD from the MINPACK library (Argonne National Energy Software Center). A recent comparison of several nonlinear equation solvers by Hiebert (1982, 1980; see the latter for more detailed conclusions than the former) rated HYBRD, which also uses a version of Powell's method, as the best performer overall.

SQPHP—Nonlinear Programming Routine

SQPHP implements an enhancement of the Han-Powell algorithm for successive quadratic programming. Details regarding the enhancement of the algorithm and its implementation are given by Chen and Stadtherr (1983), who also present performance data showing that SQPHP is more efficient and reliable than the well-regarded SQP codes VF02AD (Harwell Subroutine Library) and VMCON (Argonne National Energy Software Center).

Scaling

In the modified Powell method used by NEQLU, equation scaling is performed automatically. Therefore, we only need to supply scaling factors for variables. In SIMMOD the magnitude of a variable, x_k, in a stream that goes to module i is estimated from:

$$x_k^m = \begin{cases} \max\{|x_k|, 0.1 * a_i\} \text{ for component flowrate} \\ \max\{|x_k|, 2.0 * a_i\} \text{ for enthalpy flowrate} \\ \max\{|x_k|, 0.1\} \text{ for stream pressure} \end{cases}$$

where a_i is the average total molar flowrate of streams connected to module i. The variable x_k is then scaled by dividing it by its magnitude x_k^m. Note that the units used are: Kg·mol/s for flowrates, MJ/s for enthalpy flowrates, and MPa for pressures. For a free variable, the magnitude can be supplied by the user. Otherwise, the magnitude is the absolute value of the variable or 1.0, whichever is greater.

Because the Han-Powell method is used to solve the nonlinear programming problem, it is not necessary to scale the constraint equations for optimization problems. However, scaling the objective function and the variables may have a strong effect on the performance of the Han-Powell method, as emphasized by Chen and Stadtherr (1983). In SIMMOD the variables are scaled as described above, and the objective function is scaled automatically in SQPHP, using the scaling option $f = 0$, as described by Chen and Stadtherr (1983).

In Parts II and III, SIMMOD is used to study the performance of the simultaneous-modular on several simulation and optimization problems. SIMMOD is also used in several numerical experiments that compare different computational strategies for the simultaneous-modular approach.

ACKNOWLEDGMENT

This work has been supported by the National Science Foundation under grant CPE 80-12428.

NOTATION

a_i	= average total molar flowrate of streams connected to module i
A_{ij}	= Jacobian submatrix relating outlet stream i to inlet stream j
B_j	= Jacobian submatrix relating design specification to inlet stream j
c	= number of chemical species
c_i	= number of loops torn by stream i
D_{kj}	= diagonal approximation of Jacobian submatrix relating outlet stream k to inlet stream j
F_i	= free variable i
n_c	= number of connecting streams in current irreducible block
n_{ci}	= number of connecting input streams to module i
n_d	= number of free variables in current irreducible block
n_{di}	= number of free variables for module i
n_s	= number of connecting streams in group of modules
p	= vector of nontear stream variables
q	= vector of tear stream variables and free variables
r	= variable on which design constraint is imposed
r^s	= specified value of r
S_i	= excess specification i
u	= free variable
w_i	= weighting factor for stream i in tearing algorithm
x_k^m	= estimated magnitude of variable x_k
x_i	= vector of variables describing inlet stream i
y_j	= vector of variables describing outlet stream j
z_1	= vector of internal variables, retention variables, and output stream variables
z_2	= vector of input stream variables and equipment parameters

LITERATURE CITED

Adelman, A., and W. F. Stevens, "Process Optimization by the Complex Method," *AIChE J.*, **18**, 20 (1972).

Ballman, S. H., and J. L. Gaddy, "Optimization of a Methanol Process by Flowsheet Simulation," *I&EC Proc. Des. Dev.*, **16**, 337 (1977).

Benjamin, D. R., M. H. Locke, and A. W. Westerberg, "Interactive Programs for Process Design," *Report #DRC-06-28-81*, Design Res. Ctr., Carnegie-Mellon University (1981).

Berna, T. J., M. H. Locke, and A. W. Westerberg, "A New Approach to Optimization of Chemical Processes," *AIChE J.*, **26**, 37 (1980).

Biegler, L. T., and R. R. Hughes, "Approximation Programming of Chemical Processes with Q/LAP," *Chem. Eng Prog.*, **77**(4), 76 (1981).

Biegler, L. T., and R. R. Hughes, "Infeasible Path Optimization with Sequential Modular Simulators," *AIChE J.*, **28**, 994 (1982a).

———, "Process Optimization: A Comparative Case Study," *Paper No. 23e*, AIChE Ann. Meet., Los Angeles (1982b).

———, "Feasible Path Optimization with Sequential Modular Simulators," *Comput. Chem. Eng.*, in press (1983).

Blau, G. E., "Session Summary: Nonlinear Programming," *Foundations of Computer-Aided Chemical Process Design*, Vol. 1, R. S. H. Mah and W. D. Seider, Eds., Engineering Foundation, New York (1981).

Chen, H. S., and M. A. Stadtherr, "A Modification of Powell's Dogleg Method for Solving Systems of Nonlinear Equations," *Comput. Chem. Eng.*, **5**, 143 (1981).

———, "Enhancements of the Han-Powell Method for Successive Quadratic Programming," *Comput. Chem. Eng.*, in press (1983).

———, "A Simultaneous-Modular Approach to Process Flowsheeting and Optimization. II: Performance on Simulation Problems," *AIChE J.*, in this issue.

———, "A Simultaneous-Modular Approach to Process Flowsheeting and Optimization. III: Performance on Optimization Problems," *AIChE J.*, in this issue.

Duff, I. S., "On Algorithms for Obtaining a Maximum Transversal," *Trans. Math. Software*, **7**, 315 (1981a).

———, "Algorithm 575. Permutations for a Zero-Free Diagonal," *Trans. Math. Software*, **7**, 387 (1981b).

Duff, I. S., and J. K. Reid, "An Implementation of Tarjan's Algorithm for the Block-Triangularization of a Matrix," *Trans. Math. Software*, **4**, 137 (1978a).

———, "Algorithm 529. Permutations to Block-Triangular Form," *Trans. Math. Software*, **4**, 189 (1978b).

Evans, L. B., "Advances in Process Flowsheeting Systems," *Foundations of Computer-Aided Chemical Process Design*, Vol. 1, R. S. H. Mah and W. D. Seider, Eds., Engineering Foundation, New York (1981).

Friedman, P., and K. L. Pinder, "Optimization of a Simulation Model of a Chemical Plant," *I&EC Proc. Des. Dev.*, **11**, 512 (1972).

Gaines, L. D., and J. L. Gaddy, "Process Optimization by Flowsheet Simulation," *I&EC Proc. Des. Dev.*, **15**, 206 (1976).

Gorczynski, E. W., H. P. Hutchison, and A. R. M. Wajih, "Development of a Modularly Organized Equation-Oriented Process Simulator," *Comput. Chem. Eng.*, **3**, 353 (1979).

Han, S. P., "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," *Math. Prog.*, **11**, 263 (1976).

———, "A Globally Convergent Method for Nonlinear Programming," *J. Opt. Theory Appl.*, **22**, 297 (1977).

Hiebert, K. L., "A Comparison of Software which Solves Systems of Nonlinear Equations," *Sandia Tech. Rep. SAND 80-0181*, Sandia National Labs, Albuquerque (1980).

———, "A Comparison of Software which Solves Systems of Nonlinear Equations," *Trans. Math. Software*, **8**, 5 (1982).

Himmelblau, D. M., and J. W. Lindsay, "An Evaluation of Substitute

Methods for Derivatives in Unconstrained Optimization," *Opns. Res.*, **28**, 668 (1980).

Hlavacek, V., "Analysis of a Complex Plant—Steady State and Transient Behavior," *Comput. Chem. Eng.*, **1**, 75 (1977).

Jirapongphan, S., "Simultaneous Modular Convergence Concept in Process Flowsheet Optimization," Ph. D. Thesis, MIT (1980).

Lasdon, L. S., "A Survey of Nonlinear Programming Algorithms and Software," *Foundations of Computer-Aided Chemical Process Design*, Vol. 1, R. S. H. Mah and W. D. Seider, Eds., Engineering Foundation, New York (1981).

Locke, M. H., "A CAD Tool Which Accomodates an Evolutionary Strategy in Engineering Design Calculations," Ph. D. Thesis, Carnegie-Mellon University (1981).

Locke, M. H., and A. W. Westerberg, "The ASCEND-II System—A Flowsheeting Application of a Successive Quadratic Programming Methodology," *Paper No.* 23c, AIChE Ann. Meet., Los Angeles (1982).

Locke, M. H., R. H. Edahl, and A. W. Westerberg, "An Improved Successive Quadratic Programming Optimization Algorithm for Engineering Design Problems," submitted for publication (1982).

Mahalec, V., H. Kluzik and L. B. Evans, "Simultaneous Modular Algorithm for Steady State Flowsheet Simulation and Design," 12th Eur. Sym. Computers in Chem. Eng., Montreux, Switzerland (1979).

McLane, M., M. K. Sood, and G. V. Reklaitis, "A Hierarchical Strategy for Large-Scale Process Calculations," *Comput. Chem. Eng.*, 3, 383 (1979).

Metcalfe, S. R., and J. D. Perkins, "Information Flow in Modular Flowsheeting Systems," *Trans. I. Chem. Eng.*, **56**, 210 (1978).

Motard, R. L., M. Shacham, and E. M. Rosen, "Steady State Chemical Process Simulation," *AIChE J.*, **21**, 417 (1975).

Peng, D. Y., and D. B. Robinson, "A New Two Constant Equation of State," *I&EC Fund.*, **15**, 59 (1976).

Perkins, J. D., "Efficient Solution of Design Problems Using a Sequential-Modular Flowsheeting Program," *Comput. Chem. Eng.*, 3, 375 (1979).

Perkins, J. D. and R. W. H. Sargent, "SPEEDUP: A Computer Program for Steady-State and Dynamic Simulation of Chemical Processes," *Selected Topics on Computer-Aided Process Design and Analysis*, R. S. H. Mah and G. V. Reklaitis, Eds., *AIChE Symp. Ser.*, 78(214), 1 (1982).

Pho, T. K., and L. Lapidus, "Topics in Computer Aided Design. I: An Optimum Tearing Algorithm for Recycle Streams," *AIChE J.*, **19**, 1,170 (1973).

Powell, M. J. D., "A Hybrid Method for Nonlinear Equations," *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, Ed., Gordon and Breach, New York (1970).

———, "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," *Numerical Analysis—Dundee 1977*, G. A. Watson, Ed., *Lecture Notes in Mathematics*, Vol. 630, Springer-Verlag (1978a).

———, "Algorithms for Nonlinear Constraints that Use Lagrangian Functions," *Math. Prog.*, **14**, 225 (1978b).

———, "The Convergence of the Variable Metric Method for Nonlinearly Constrained Optimization Calculations," *Nonlinear Programming Symposium* 3, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Eds., Academic Press, New York (1978c).

Rosen, E. M., "A Machine Computation Method for Performing Material Balances," *Chem. Eng. Prog.*, **58**, 69 (1962).

———, "Steady State Chemical Process Simulation—State of the Art Review," *Computer Applications to Chemical Engineering*, R. G. Squires and G. V. Reklaitis, Eds., *ACS Symp. Ser.*, **124**, 3 (1980).

Sargent, R. W. H., "A Review of Optimization Methods for Nonlinear Problems," *Computer Applications to Chemical Engineering*, R. G. Squires and G. V. Reklaitis, Eds., *ACS Symp. Ser.*, **124**, 37 (1980).

Schubert, L. K., "Modification of a Quasi-Newton Method for Nonlinear Equations with a Sparse Jacobian," *Math. Comput.*, **25**, 27 (1970).

Shacham, M., et al., "Equation-Oriented Approach to Process Flowsheeting," *Comput. Chem. Eng.*, **6**, 79 (1982).

Sood, M. K., and G. V. Reklaitis, "Solution of Material Balances for Flowsheets Modeled with Elementary Modules: The Constrained Case," *AIChE J.*, **25**, 220 (1979).

Sood, M. K., R. Khanna, and G. V. Reklaitis, "A Two-Level Approach Exploiting Sparsity in Flowsheet Material Balancing," *Paper No. 30g*, AIChE Nat. Meet., Houston (1979a).

Sood, M. K., G. V. Reklaitis, and J. M. Woods, "Solution of Material Balances for Flowsheets Modeled with Elementary Modules: The Unconstrained Case," *AIChE J.*, **25**, 209 (1979b).

Stadtherr, M. A., and C. M. Hilton, "Development of a New Equation-Based Process Flowsheeting System: Numerical Studies," *Selected Topics on Computer-Aided Process Design and Analysis*, R. S. H. Mah and G. V. Reklaitis, Eds., *AIChE Symp. Ser.*, **78** (214), 12 (1982a).

———, "On Efficient Solution of Large-Scale Newton-Raphson-Based Flowsheeting Problems in Limited Core," *Comput. Chem. Eng.*, **6**, 115 (1982b).

Stadtherr, M. A., and E. S. Wood, "Sparse Matrix Methods for Equation-Based Chemical Process Flowsheeting. I: Reordering Phase," *Comput. Chem. Eng.*, in press (1983a).

———, "Sparse Matrix Methods for Equation-Based Chemical Process Flowsheeting. II: Numerical Phase," *Comput. Chem. Eng.*, in press (1983b).

Tarjan, R., "Depth-First Search and Linear Graph Algorithms," *SIAM J. Comput.*, **1**, 146 (1972).

Upadhye, R. S., and E. A. Grens, "Selection of Decompositions for Process Simulation," *AIChE J.*, **21**, 137 (1975).

Van Meulebrouk, M. G. G., A. G. Swenker, and J. A. de Leeuw den Bouter, "Using TISFLO-II for the Optimization of Utility Generation and Supply for a Chemical Complex," *IChemE Symp. Ser.*, **74**, 7 (1982).

Westerberg, A. W., "Optimization in Computer Aided Design," *Foundations of Computer-Aided Chemical Process Design*, Vol. 1, R. S. H. Mah and W. D. Seider, Eds., Engineering Foundation, New York (1981).

Westerberg, A. W., et al., *Process Flowsheeting*, Cambridge Univ. Press, Cambridge (1979).

Wilson, R. B., "A Simpical Method for Convex Programming," Ph. D. Thesis, Harvard University (1963).

Manuscript received Apr. 19, 1983; Revision received Sept. 27, 1983 and accepted Jan. 8, 1985.