SIMULTANEOUS MODULAR SIMULATION AND OPTIMIZATION

by

L.T. Biegler

December, 1933

DRC-06-JW-83

SIMULTANEOUS MODULAR SIMULATION AND OPTIMIZATION

by

L.T. Biegler
Carnegie-Mellon University, Pittsburgh, PA 15213

Chemical engineers have long thought of process models in terms of the physical systems they represent* Consequently, the most common approach to plant-vide model description is sequential modular. Here, individual module models relate to distinct physical processes and link together according to flowsheet topology. The calculation flow is thus rigidly fixed by the flowsheet. Although this procedure is reliable, easy to assemble and usually robust, it often lacks the flexibility to perform design and optimization tasks.

The equation based approach offers complete flexibility in specifying design constraints, solving optimization problems, and deriving a solution procedure. However, since the entire equation set bears little resemblance to the process flowsheet, much more work is required to set up and test the process model.

Here we discuss the progress of the simultaneous modular method in capitalizing on the advantages of the above approaches. Simply put, simultaneous modular seeks the flexibility of equation based systems while working with lfblack-boxM process modules.

We review past and current simultaneous modular strategies for design and optimization problems. Areas of current research as well as directions for future work are discussed.

INTRODUCTION

Since its development, chemical engineers have used the computer to gain more insight into chemical processes. The first models were stand-alone computer programs of individual pieces of equipment or simple processes. As computers became more powerful, it seemed quite natural to string these stand-alone programs together to form larger process models. Each model could then be constructed and tested individually, and further developed to include faster algorithms or reflect more complex phenomena. Should an error occur in the module, it is relatively easy to locate, confine and correct without greatly disrupting the entire flowsheet model.

Why then are other simulation modes desired? The main reasons stem from applications to coupled flowsheets with nested recycle loops and the imposition of design specifications that lead to deviations from the normal information flow. An acyclic network with each module calculating its output, given the output of the previous nodule, is perhaps the most efficient way of simulating a process. However, with recycle streams and design specifications, the sequential modular mode can be very inefficient because some level of iteration is required for its solution. The efficiency of the convergence procedure depends greatly on the amount of information available from the flowsheet. It is for this reason alone that sequential modular simulation is inefficient.

Consider a simple single loop flowsheet where we tear a given stream (choose a stream on which to iterate). Defining as a measure of convergence the difference between the guessed and calculated torn stream (y and w, respectively), we pose the simulation problem as:

$$\text{Solve} \quad h(y) = y - w = 0 \quad .$$

A large family of nonlinear equation solvers can be used for this problem. The most common ones have the form:

$$z^{M} - i^{1} - i\,to^{1}) \cdot$$

Several methods applicable to sequential modular flowsheeting are listed in Table 1. Note that there are several ways of calculating

Table 1

Sane Methods for Solving Nonlinear Equations
(Motard et al. (1975))

General Form:

$$y^{i+1} = y^i - J\, h(y^i)$$

$$h(y^i) = y^i - w(y^i)$$

Method	J
Successive Substitution	I
Wegstein	$D = \text{diag}\,\{d_{jj}\}$ $d_{jj} = \dfrac{y_j^i - y_j^{i-1}}{h_j^i - h_j^{i-1}}$
Dominant Eigenvalue	$\left[\dfrac{1}{1-\lambda} I\right]$ $\lambda = \dfrac{\lVert w_i - w_{i-1} \rVert}{\lVert y_i - y_{i-1} \rVert}$
Broyden's Method (1965)	Full matrix quasi-Newton update
Nevton's Method	$\left[\dfrac{\partial h^i}{\partial y}\right]^{-1}$

the matrix $\underline{\underline{J}}$. Successive substitution requires no flowsheet information for $\underline{\underline{J}}$ while Newton's method requires the Jacobian of h(y) at each iteration. Here we come to a major problem of these simulators* In order to apply the simplest of methods, only a flowsheet pass, which evaluates My), is needed. However, even if the process converges, its rate is only linear. A more efficient strategy, like Newton's method, requires a full Jacobian. Here since process modules are essentially input-output black boxes which reveal little information about their constituent equations, the easiest, and often only, way of obtaining derivative information is by perturbation. However, this approach is time-consuming and potentially inaccurate since modules usually contain function and derivative discontinuities.

The more significant problem is therefore the efficient solution of the flowsheet with limited modular information. While Newton's method is relatively independent of flowsheet topology, other algorithms such as simple successive substitution strongly depend on the calculation procedure and often require special care in choosing tear sets.

Thus to summarize the above points, the attributes of sequential modular simulators are:

1) They lead to the formulation of physically meaningfully solution strategies and, thus, a process simulation is relatively easy to construct, debug, analyze and interpret.

2) Any sophisticated convergence algorithm applied to the tear set needs flowsheet information that usually requires prohibitive computational effort. Thus, the simplest convergence methods must be used.

3) Because of the black box nature of the process modules, the solution procedure is rigidly determined by the flowsheet topology. Also, since these modules are constructed to calculate output streams and calculated parameters from input streams and a prespecified set of input parameters, any constraints the engineer wishes to impose on the simulation must be handled as additional iteration levels (control loops) that further complicate the flowsheet topology. Here, only simple convergence methods can be used, so special care must be taken in determining tear sets.

I. Simultaneous Modular Strategies

As mentioned by Perkins (1983), equation based flowsheeting systems are not hindered by the shortcomings of sequential modular simulators. Because these simulators can derive as much information as they need from an equation-based model they have great flexibility in deriving solution procedures and applying efficient convergence algorithms. The disadvantage to the equation-based approach is mainly due to implementation and the engineer's need to understand the simulation process. In theory there is no question that an equation-based approach is superior to the sequential modular strategy. In practice, sequential modular simulators will continue to be used because

1) they are much easier to construct and understand;

2) they presently require less core storage and thus can be extended to much larger simulation problems;

3) incorporation of new modules or more complex versions for unit operations can be made easily without changing the overall solution strategy;

4) process simulations are currently easier to program and debug in the sequential modular mode;

5) they are available, they work and they are most familiar to engineers.

In time, statements 1) to 4) will become invalid. Perhaps the greatest inertia in converting to equation-based modes is due to statement 5). On the other hand, many researchers have sought to combine the best of both strategies. Although their efforts have taken remarkably different forms, they deserve to be classified under the general heading of simultaneous modular strategies. Broadly speaking, the simultaneous modular approach can be defined as the art of flexibly solving simulation problems made up of black box process modules.

Not surprisingly, the development of simultaneous modular strategies parallels that of equation-based approaches. In the first section the stages of development will be summarized as follows:

A) Development of optimal tear strategies for sequential modular simulators

B) Simultaneous convergence applied to design constraints and tear streams

C) Linear module information calculated from the flowsheet for faster convergence

D) Nonlinear models derived from the flowsheet.

Here, we begin by highlighting graph theoretic techniques that were originally applied to systems of equations and conclude by discussing strategies that use simplified flowsheet models that lend themselves to the advantages of equation-based simulators.

A. Optimal Tear Strategies

In the early sixties, as stand-alone modules were linked to form process flowsheets, the question of deriving efficient calculation sequences became important. This was one of the first applications of graph theory to chemical processes (Mah, 1982). Here, the flowsheet is treated as a digraph that could be analyzed to determine an optimal tear set. This problem is similar to equation ordering in that nodes (modules) must be partitioned and ordered while any remaining irreducible sets are solved iteratively. For process modules the algorithmic decomposition procedure is easier than with equations because the output set (streams calculated by the module) is already fixed by the flowsheet. However, the concept of an "optimal" tear set is still as difficult to define as with equation-based simulators. Moreover, equations with analytic derivatives can be partitioned into irreducible sets and solved simultaneously. Sequential modular simulators, on the other hand, require simpler solution algorithms where performance is strongly determined by the tear set. Here, determination of the "optimal" tear set has been represented conceptually as a set covering problem (Pho and Lapidus, 1973; Rosen, 1980):

$$\text{Min} \quad \sum_{j=1}^{N} p_j x_j$$

$$\text{s.t.} \quad \sum a_{ij} x_j \geq 1$$
$$x_j = 0,1 \qquad j=1,N$$

where the binary variables x_j correspond to whether stream j is torn (1) or not (0). The coefficients a_{ij} can be assembled into a (usually) sparse loop matrix. A value of one for a_{ij} indicates that stream j appears in loop i. Otherwise $a_{ij}=0$. Finally, p_j are user defined weights on the tear stream. The integer program now finds a minimal weighted tear set that breaks each loop at least once. In order to obtain an optimal set, several criteria were proposed for selecting the weights p_j.

1) Tear minimum of streams, $p_j=1$ (Barkeley and Motard, 1972).

2) Tear minimum of stream variables, p_j = variables in stream j.

3) Tear as many loops as possible only once (nonredundant tears) (Upadhye and Grens, 1975).

4) Select optimum tear set adaptively by obtaining a measure of the eigenvalues for convergence (Genna and Motard, 1975).

Comprehensive reviews of these strategies are given in Mah (1982) and Gundersen and Hertzberg (1982). However, none of these strategies explicitly addresses the problem of design constraints. Here, a design constraint is a process specification that cannot be entered explicitly as a flowsheet input parameter. In the conventional sequential modular mode, an additional iteration level, in the form of a control loop, is thus imposed to meet this constraint. Needless to say, the presence of control loops greatly influences tear set selection and drastically affects algorithmic performance if not considered.

B. Application to Design Constraints

Metcalfe and Perkins (1978) considered this controlled simulation (or design) problem by explicitly reversing the flow of information in the digraph and altering the levels of iteration. After deleting "undefined" and "overdefined" streams, sequencing can be performed using existing tearing methods. Equations describing the design constraints as well as the tear streams are then solved simultaneously.

Chen (1982) also addressed the controlled simulation problem by including design constraints after the flowsheet streams were torn. Control loops are imposed by determining an output set for the design problem (by assigning the manipulated variables to design constraints) and then tearing the control loops separately.

Using the approach of Perkins or Chen, the controlled simulation can be greatly improved. To better illustrate this approach, consider the design or controlled simulation problem written as:

$$\text{(CSP)} \quad h(y_f z) \gg \mathbf{y} - v(y,z) - 0$$
$$\mathbf{c(y,z)} - 0$$

In the sequential modular mode, control loops create two iteration **levels where** only **one** of **the above** relationships is converged at a **time.** For example, if the design constraint is contained within a **recycle** loop, the flowsheet simulation strategy is:

Solve Outer Iteration: $\mathbf{h(y,\ll(y) - y - w(y,z(y)) - 0}$

Solve Inner Iteration: $\mathbf{c(\bar{y},z) \ll 0}$

where $\bar{y}$ is held constant in the inner iteration and z(y) is chosen **in the** outer iteration so that the design constraint, c, is satisfied.

Both Chen and Perkins (1979) propose the simultaneous solution of the controlled simulation problem (CSP). This simple concept had **not been** employed before because most common flowsheet convergence **methods** fail if design constraints are included. Instead, Perkins **used** Broyden's method while Chen used a modification of Powell's dogleg algorithm coupled with Broyden's method. The latter method helps the system of equations converge from poor starting points. Both studies reported significant improvement over conventional sequential modular strategies.

C. Linear Information Derived from the Flowsheet

Instead of deriving calculation strategies and selecting "optimal*9 tear sets, a large body of simultaneous modular literature deals with obtaining more information from the flowsheet so that more efficient convergence techniques can be applied.

Again, these strategies parallel another approach to equation-based simulation: simultaneous solution by Newton's method. With a flowsheet made up of black boxes, however, the approach to this problem is slightly different. Here, evaluation of the J matrix by perturbation or some other approximation can be time consuming, but the flowsheet structure does allow us to reduce the size of the problem. In this section, one can identify two classes of simulation strategies:

1) Newton's method applied to the flowsheet tear set.

2) Newton's method applied to all process streams.

Needless to say, the controlled simulation problem can be handled in both cases simply by simultaneously converging the design constraints with the process stream equations.

1) Tear Set Convergence. For the simulation problems:

$$h(y) - y - w(y) \quad ,$$

shown in Fig. 1, Newton's method gives us the relation

$$\begin{array}{l} \quad i+1 \qquad i \quad /L \quad dvV^{1} \;, \quad , \; XN \\ \; y \qquad -y \quad \bullet v j - s r j \quad (yw(y)) \\ 1 \qquad\qquad\qquad x \qquad oy \; y \end{array}$$

with (I-3w/3y)~ as the J matrix. The use of a more sophisticated strategy now allows us to solve nested recycle problems simultaneously; if 3w/3y is available, Newton's method coupled with more robust methods (e.g. dogleg, continuation, etc.) could be used. The advantages of this approach have led to the development of numerous simultaneous modular strategies. They can all be summarized as attempts to find good approximations to 3w/3y.

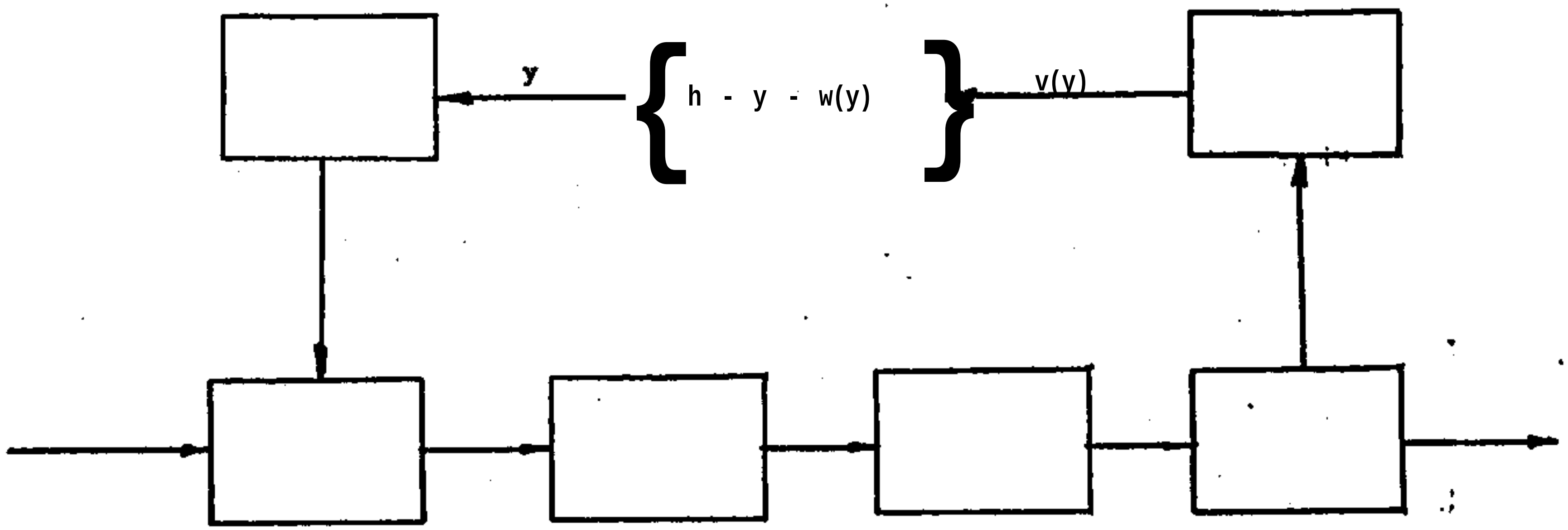


Figure 1. Simple Simulation Problem

The first such attempt was the method of split fractions (Nagiev, 1957; Vela, 1961; Rosen, 1962). Here the input-output Jacobians of each module are approximated by a diagonal matrix. Each diagonal element is evaluated merely by dividing the outlet component flowrate by its corresponding inlet component flowrate. Reactor modules that do not have diagonally dominant Jacobians and create or consume chemical species are treated by pseudo-feeds and outputs. Rosen (1962) mentioned that this procedure worked well on his example problems but offered no guarantees for efficiency or stability. Other studies (Klemes, et al., 1976; Lutcha, et al., 1975; Umeda and Nishio, 1972; Vasek, et al., 1974) have reported mixed results, depending on the size and complexity of the problem.

Several studies (Komatsu, 1968; Umeda and Nishio, 1972) used simplified linear Jacobian approximations based on Nishimura's (1967) approach. Here modules that have diagonally dominant Jacobians such as separators, splitters and mixers are described by diagonal models. Reactor Jacobians are approximated with off-diagonal terms corresponding to the components participating in the reaction sequence. Nishio and Umeda (1972) compared this approach to the conventional sequential modular and Rosen's split fraction strategies. In addition to showing superior performance on several example problems, they also demonstrate the stability of their method over split fractions. It appears the major reason for this is due to the inclusion of off-diagonal terms in reactor Jacobians. Finally, they derive sufficient conditions when their simultaneous modular approach converges faster than direct substitution.

Reklaitis and coworkers (1979) used a similar approach to solve linear mass balance problems. To obtain a solution the flowsheet is first described in terms of three basic linear modules: splitters, reactors and mixers. To solve the linear system only a subset of the mixer equations need be written explicitly. All other stream variables can be eliminated by chain-ruling the Jacobians of the linear modules. This approach has been demonstrated on large linear mass balance problems and was extended to design problems with linear or nonlinear design constraints. The advantage of this approach is that . linearity is fully exploited and iterative calculations are eliminated except for the nonlinear constraints. This approach was later included in a hierarchical strategy that interfaced with black box process modules (MeLane, et al., 1979). To solve the simulation, different levels of module aggregation are employed starting from individual linearized models for internal recycle streams to completely aggregated models for the entire plant.

In their description of FLOWPACK-II Berger and Perris (1979) mention a similar concept called "subnetworking.[11] Here process modules are decomposed into smaller modules that can relate flows of streams, enthalpies and pressures independently. In a manner similar to equation-based simulation, the decomposed flowsheet now offers greater flexibility in deriving solution procedures and handling design constraints. The drawback with "subnetworking,[11] however, is that the decomposed flowsheet now becomes much more complex and determination of the solution strategy becomes more difficult.

2) " All Stream Convergence. To avoid determination of tear sets that reduce the size of the system, several researchers have adopted the "all-stream[11] tear strategy. As shown in Fig. 2 on a simple flowsheet, this strategy deals with the satisfaction of two sets of equations:

Input-output relationships

$$\Delta y_i = A_i (\Delta x_i)$$

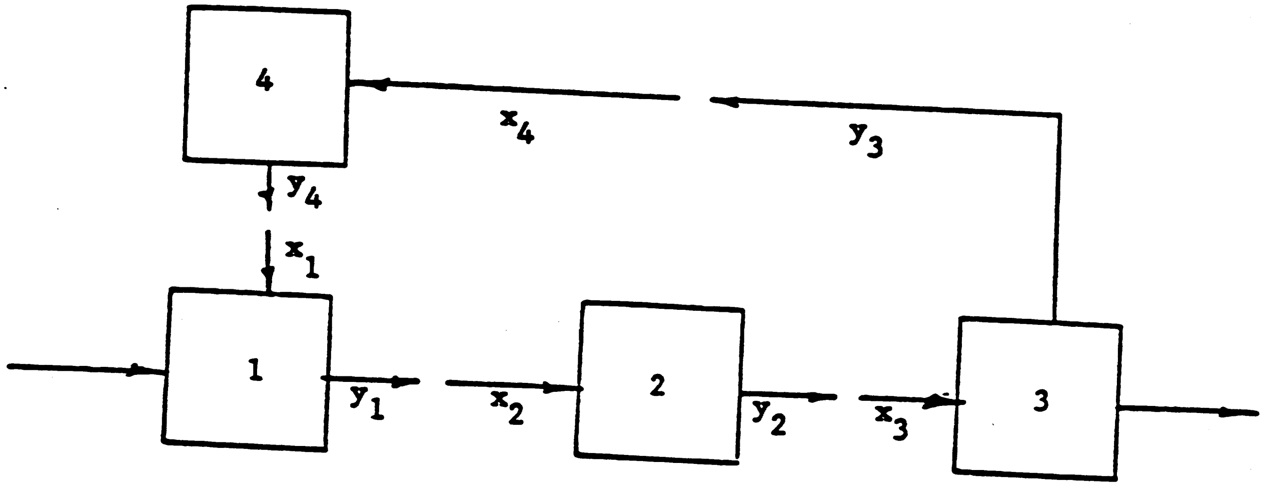
and stream connections

where y_i are output streams for module i

x_i are input streams

and module j is immediately downstream of module i

Mahalec, et al. (1979) combined these relationships into a large linear system that was solved by sparse matrix decomposition. Several methods were compared for evaluating the input-output Jacobian, $\underline{\underline{A}}$.



$$\begin{bmatrix} -A_1 & & & & I & & & \\ & -A_2 & & & & I & & \\ & & -A_3 & & & & I & \\ & & & -A_4 & & & & I \\ I & & & & & & & -I \\ & I & & & -I & & & \\ & & I & & & -I & & \\ & & & I & & & -I & \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta x_4 \\ \Delta y_1 \\ \Delta y_2 \\ \Delta y_3 \\ \Delta y_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ y_4 - x_1 \\ y_1 - x_2 \\ y_2 - x_3 \\ y_3 - x_4 \end{bmatrix}$$

"All-Stream" Tear Formulation

$$\begin{bmatrix} I & & & -A_4 \\ -A_1 & I & & \\ & -A_2 & I & \\ & & -A_3 & I \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta x_4 \end{bmatrix} = \begin{bmatrix} y_4 - x_1 \\ y_1 - x_2 \\ y_2 - x_3 \\ y_3 - x_4 \end{bmatrix}$$

Reduced Formulation

Figure 2. Decomposition Strategies

For their simulation problems, they found that by initially choosing diagonal Jacobian approximations (except for reactors) and applying Schubert (1970) updates at every iteration, significantly better performance was observed over the sequential modular approach.

An obvious simplification to the above approach would be the elimination of the stream connection equations (Sargent (1979)). This is the method used in the flowsheet condensation steps of the QAP (Parker and Hughes, 1981) and the Q/LAP (Biegler and Hughes, 1982) optimization algorithms. As seen in Fig. 2, this simplification essentially halves the size of the linear system and reduces the effort required to solve the linear equations.

All of these methods require some information from the simulator at each iteration. Regardless of the convergence method, the major effort for simulation is still governed by evaluation of black box process modules. All of the above Newton-based methods, including those of Chen (1982) and Perkins (1979), can be viewed as approximating the flowsheet as a linear mass (and energy) balance problem at each iteration. Since even the fastest "linear" method requires several iterations of the nonlinear simulation problem, any gains in computational efficiency can only result if <u>more</u> than gradient information is obtained from the flowsheet. Thus, the most recent developments for simultaneous modular simulation have dealt with solving <u>nonlinear</u> mass and energy balance approximations at each iteration.

Working within an optimization framework, Hughes (1975) and Isaacson (1975) approximated modular behavior by quadratic models of the form:

$$\left\{\begin{matrix} \Delta w^i{}_j \\ \Delta r^i_j \end{matrix}\right\} = a_j^T \Delta x^i + b_j^T \Delta y^i + \Delta x^{i^T} A_j \Delta y^i + \Delta x^{i^T} B_j \Delta x^i + \Delta y^{i^T} C_j \Delta y^i$$

where	y^i_j	- input stream variable j for module i
	w^i_j	- output stream variable j for module i
	x^i	- module design variable set
	r^i_j	- module retention variable j (calculated parameters)

However, model coefficients usually cannot be evaluated analytically and calculation by finite difference is expensive and potentially inaccurate. In Isaacson's optimization study, the $Ay^{iT} C_j Ay^i$ term did not contribute much flowsheet information; omitting this term led to significant reductions in model construction. The results of this study indicate that nonlinear flowsheet approximations are no more efficient if they are constructed from higher order derivatives. Instead, the most fruitful developments for nonlinear approximation have directly employed simplified flowsheet models to promote convergence.

D. Nonlinear Models Derived from the Flowsheet

The concept of representing the rigorous model in terms of simplified models at each iteration stems from a similar strategy for physical property evaluation. In order to avoid complex calculations at every iteration, simple models were proposed (Barrett and Walsh, 1979; Leesley and Heyen, 1977) with adjustable parameters that could be matched with rigorous physical properties at selected points. Barrett and Walsh (1979) also developed measures for evaluating the magnitude of error for the simplified models in order to determine when the rigorous physical properties should be' re-evaluated.

Boston and Britt (1978) extended this concept to flash calculations by totally redefining the solution procedure. Here the "primitive" variables that are usually chosen for convergence are solved explicitly using simplified correlations for K values and vapor and liquid enthalpies. New variables chosen for convergence are the adjustable parameters in the simplified models. The calculation strategy is thus:

Outer iteration:

solve $S(3_f\bar{y}) - R(Yt\bar{7})$

Inner iteration

Find $\bar{y}$ that solves the simplified flash calculation with 8 constant.

Here 6 - adjustable parameters for simplified models
R - rigorous models evaluated in the outer iteration
y - variables calculated by flash calculation
Y - parameters used in rigorous property evaluations
S - simplified models that are functions of 0

The inner iteration represents a nonlinear approximation of the rigorous flash problem* After rearrangement of the simplified equations, this simply becomes a one variable convergence problem. Moreover, the outer iteration performs smoothly because the parameters 6 are not as sensitive to changes in temperature or liquid-vapor ratios. The algorithm, termed inside-out, works well even for highly nonideal systems or wide and narrow-boiling systems.

Boston and coworkers (1980, 1974) have also extended this concept to multistage flashes and staged operations. Again, less frequent evaluation of rigorous physical property routines has resulted in significant reductions in computational effort.

The extension of this approach to flowsheeting is straightforward. For simulation one can treat the black box process modules in the same manner as the physical property calculations. The inner iteration could then be performed using simplified engineering models. These concepts were recently proposed by Pierucci, et al. (1982). Here, one pass of the flowsheet was evaluated to determine starting values for internal stream vectors and parameters for simplified models. The simplified (or evolutionary) models for each unit are then used to converge the flowsheet. The calculation sequence for the inner iteration is the same as the one specified for the flowsheet, but now the tear equations can be solved faster because the evolutionary models have analytic gradients. This scheme requires no special equation solving technique and serves only to update the value of the tear set. Because of the form of the evolutionary modules, this strategy can also be extended to handle controlled simulation problems.

Jirapongphan, et al. (1980) proposed a similar strategy that was also applied to flowsheet optimization. Here the nonlinear approximation interacts even less with the flowsheet and follows more the strategy of the inside-out algorithm. Flowsheet "black box^{11}

modules are used only to calculate the adjustable parameters for the simplified models (denoted as Reduced Analytical Modules (RAM)), which are then solved simultaneously as a large sparse system. Convergence is obtained when the RAM parameters, not the tear streams, approach a stationary point. More will be mentioned about this method in the optimization section.

Any problems with nonlinear approximation simultaneous modular methods are mainly due to the form and accuracy of the simplified models. Because of their nonlinearity, the models must frequently be solved iteratively, either by sequencing and solving the tear set or by simultaneously converging a sparse system of nonlinear equations. The inner iteration is thus identical to solving (possibly over-) simplified simulation problems. Here, questions that must be addressed are:

1) Do simplified models exist that accurately represent complex model (i.e. realistic process) behavior?

2) Are there advantages to solving the simplified model (probably) more often than the complex model?

3) What are the necessary and sufficient conditions on the form of the simplified models that govern convergence of the outer iteration.

The examples solved by Pierucci et al and Jirapongphan indicate that these methods are clearly superior to the conventional sequential modular approaches and can handle problems much larger than those attempted by equation-based simulators. However, future work with these strategies must also deal with complex reactors and distillation models where the choice of limited or inaccurate simplified models can cause convergence failure in the outer iteration (Chimowitz, et al., 1983).

Another case that must be addressed is the wide use of discontinuous and nondifferentiable models in sequential modular simulators. These manifest themselves in sizing and cost correlations, in nonideal distillation columns, in reactors, and as embedded optimization problems (Clark and Westerberg, 1982) that describe multiphase flash calculations. As yet, no simultaneous modular approach has considered this problem even though it can be a major cause of convergence failures. More will be said about this in the next section.

In reviewing simultaneous modular strategies, we see how this simulation mode has "bridged the gap" between an inflexible, but robust method (sequential modular) and the equation-based strategy that offers great flexibility for design and simulation.

The development of this simulation mode has been spurred by the advances of equation-based modes in solving large sets of linear and nonlinear equations. However, it has always been tied to "black box" modules which can model process behavior in as much detail and rigor as necessary.

For the present, application of this philosophy will lead to very efficient, frequently-used and successful simulation strategies. The simultaneous modular approach combines the efforts of detailed model representation embodied in sequential modular strategies with powerful equation-based strategies and makes them both immediately useful*

II• Simultaneous Modular Optimization

Process improvement is a major activity among chemical engineers. Faced with obvious economic and logistical incentives, the use of successful and efficient optimization techniques is clearly desirable. Unfortunately, the explicit development and use of these techniques is not widespread in industry. At the last FOCAPD conference several reasons were cited (Blau, 1980; Westerberg, 1981) for this fact. To motivate the discussion of simultaneous modular optimization, it is useful to consider some of these reasons.

Put concisely they can be listed as:

1) It is often difficult to accurately define an objective function and formulate a mathematical program for the engineering problem.

2) Current simulation and optimization tools are too expensive (in terms of the engineer's and the computer's time) and not especially "user-friendly."

3) There is so much uncertainty in the design and operation of the process that the information gained from an optimization study may not be meaningful.

4) The penalty for failure to meet project deadlines clearly increases the reluctance toward implementing these tools.

The second and fourth reasons can be overcome by development of more efficient, robust and friendly tools. In time, I feel that these reasons will no longer be valid provided that new strategies are implemented well and the user is conscious of the algorithm's limitations.

Uncertainty in design is often a problem that precludes a rigorous optimization study. Presently, however, this is due to the prohibitive effort required by many brute-force optimization techniques. An efficient optimization tool that can be applied frequently and easily is very useful for sensitivity analyses. Moreover, one can also pose optimization problems that include and exploit process uncertainty (see Grossmann and Morari (1983)).

The first reason deals more frequently with process synthesis but also applies to certain aspects of process optimization as well. Because defining an "optimal" plant is difficult and formulation of the mathematical programming problem can include continuous and discrete.decisions and discontinuous and nondifferentiable functions, the designer is often left with a problem that cannot be solved. This realization has led several researchers (Linnhoff, 1981; Morari, 1982; Stephanopoulos, 1981) away from numerical solutions to the clever application of heuristics. Here, several problems such as special cases of energy recovery and separation systems can be solved efficiently merely by representing the problem compactly and applying sound engineering concepts (e.g. thennodynamic targets).

It should be mentioned, though, that heuristic strategies may offer no guarantee of optimality and additional process restrictions can become difficult to deal with. Here, the general framework is also not as well defined as with mathematical programming. Finally, one should realize that although heuristics often lead to good solutions, there is much improvement that can be realized through manipulation of continuous variables. This is especially true in the optimization of simulated processes, where minor variations in continuous variables have sometimes realized process improvements that would justify the use of even the crudest automatic optimization strategies (Gaines and Gaddy, 1976). Since chemical processes contain complex interactions, it is difficult, if not impossible, to develop heuristic strategies that will yield these optimal results.

Thus, while the engineer should not be reduced to the level of a technician tied to an automatic tool, he must be aware of the power of optimization strategies. To this end the problem of formulating meaningful optimization problems is as important as determining thermodynamic targets; judicious application of state-of-the-art optimization strategies is equally as important as the application of heuristics.

In this spirit we review optimization strategies for process flowsheeting and discuss the recently developed infeasible path algorithm. We divide these methods into three classes:

A) Two-level Optimization Techniques

B) Feasible Path Strategies

C) Infeasible Path Strategies

Here, nonlinear programming algorithms will only be described as they pertain to optimization strategies since space precludes comprehensive treatment. Excellent reviews of algorithms and software are available elsewhere (Gill, et al., 1981; Lasdon, 1981; Sargent, 1980).

The optimization problem can be written as:

$$\text{(P1)} \qquad \min_{z} \ \phi(z)$$
$$\text{s.t.} \quad g(z) \leq 0$$
$$h(z) = 0$$
$$c(z) = 0$$

where z represents all of the variables in a given process, ϕ is an objective function, typically of economic form, and the inequalities $g(z) \leq 0$ represent physical or "designed" limits on process operation. The equality constraints are divided into two sets: $h(z) = 0$ are equations needed to simulate the process while $c(z) = 0$ are conditions imposed by the designer. The variables are usually partitioned into two sets denoted as x and y. Here the vector x represents the process decisions; its dimension is the number of degrees of freedom in the process. The y variables can be calculated from c and h once the x variables have been chosen.

Perhaps the most appealing ideas for process optimization stem simply from the modularity of chemical processes. From the classical study of unit operations and countless hours of observing equipment performance, it becomes inevitable that individual process units will be optimized. The question then remains, how does our knowledge of these units impact on plant-wide process optimization? During the sixties and early seventies, this led to the development of several elegant decomposition techniques.

A. Two-Level Optimization Techniques

The first attempts at flowsheet decomposition used dynamic programming (Aris, 1964) for acyclic processes. Jackson (1964) and Brosilow and Lasdon (1965) were the first to propose the two-level strategy for cyclic process flowsheets. Here, the optimization problem must have separable objective and constraint functions. Equality constraints are further classified as equations that represent the module model and connectivity equations that link the modules.

Because the optimization problem is separable, sub-lagrangians can be written for each module (or stage). The coupling between these functions occurs through adjoint variables (Lagrange multipliers on the connectivity equations). At the first level the sub-lagrangians are minimized with adjoint variables held constant. The adjoints are then updated at the second level to help satisfy the interconnection equations. One serious drawback of this approach is that unless the optimization problem is convex, the region around the solution may have a dual gap and the algorithm will not converge. Stephanopoulos and Westerberg (1975) proposed a remedy to this problem, but sacrificed the separability properties.

However, even when this strategy succeeds, it performs inefficiently (Jung, et al., 1972) and requires prohibitive computational effort* Consequently, this approach has found little use outside of the academic world and is currently obsolete.

B. Feasible Path Strategies

A more straightforward approach to optimization derives from parametric studies directly applied to the simulator. Since the process can be modeled and simulated for a wide variety of cases with few changes in input data, it becomes easy to let an optimization algorithm automatically perform the case studies. However, this approach suffers from the same limitations that apply to sequential modular simulators. Here the optimization problem becomes:

(P2) WLn rf(x)

s.t. g(x) £ 0

c(x) - 0 .

Note that the process equations (h(x,y) » 0) need to be solved every time the objective function is evaluated. Since the flowsheet consists of black-box modules, simulation will usually be performed by slow convergence techniques. Moreover, the application of efficient gradient-based optimization techniques is hindered for two reasons. First, derivatives can only be evaluated by perturbing (and re-simulating) the entire flowsheet with respect to the decision variables. This process is not only time consuming but requires relatively tight convergence tolerances in order to minimize perturbation error. The second reason is due to the nature of the black boxes. Process module behavior is often described by discrete and discontinuous relations or by functions that may be nondifferentiable at certain points. Thus, even if there is no perturbation error "gradients" still may be inaccurate.

In this case only direct search algorithms offer any "guarantee" of success (Westerberg, 1981). Numerous studies with adaptive random (Friedman and Pinder, 1972; Gaines and Gaddy, 1976; Ballman and Gaddy, 1977), complex and pattern search algorithms tied to sequential modular simulators have demonstrated that this approach is inefficient and too expensive to be done frequently. It may be largely due to this reason that widespread use of process optimization techniques is not common in industry even though there is a clear need for them.

In order to use more efficient gradient-based algorithms for simulator optimization, Hughes (1975) and Isaacson (1975) constructed quadratic module models at each iteration that could be combined and reduced to form the quadratic optimization problem:

$$\text{(P3)} \qquad \text{WLn} \quad a^T Ax + Ax^T A\Delta x$$

$$\text{s.t.} \quad b^T \pounds x + AX^{\prime} BAX \pounds 0$$

$$c^T Ax + Ax^T CAx - 0$$

The solution of this problem then determines the next base point for model construction. At first glance this strategy appears cumbersome, but it eliminates the need for perturbation and repeated simulation of the entire flowsheet. Parker and Hughes (1981) applied this strategy to a FLOWTRAN simulated ammonia process with 8 degrees of freedom and found the optimum in the equivalent time of 65.4 simulations. Biegler and Hughes (1981) used only linear module models to obtain reduced gradient information and applied an SQP algorithm (Han, 1977; Powell, 1977) to (P2). Here the same ammonia synthesis process was solved in only 12.6 simulation time equivalents. However, this strategy still requires the simulation of the flowsheet at every iteration.

C. <u>The Infeagible Path Strategy</u>

Here we need to consider the structure of sequential modular simulators and how it can be exploited for optimization. The simulation problem can be written simply in terms of the tear equations:

$$h(y) * 7 - w(y)$$

since w is directly determined from the tear variables, y, by evaluating the process loop. The optimization problem (PI) is thus reduced to the form:

$$
\text{(RP)} \quad \text{Min} \quad \phi(x,y)
$$
$$
\begin{aligned}
\text{s.t.} \quad & g(x,y) \le 0 \\
& c(x,y) = 0 \\
& h(x,y) = y - w(x,y) = 0
\end{aligned}
$$

In previous optimization strategies the last constraint was converged either in the outer loop by the two-level algorithm or in an inner loop by the feasible path approach. In the infeasible path strategy this constraint is converged simultaneously with the optimization problem. Here the problem is small enough that no decomposition techniques are required. Also the work required per iteration is not excessive because the flowsheet converges only as the optimum is found. To discuss this strategy we present three components that account for its algorithmic performance and also represent areas for future research. These are:

1) Choice of Optimization Algorithm

2) Gradient Calculation Strategy

3) Attributes of Process Modules

1) Choice of Optimization Algorithm. This is the most important aspect in judging the performance of the infeasible path approach. The nonlinear programming strategy must be able to handle nonlinear equality constraints; thus we are limited to the following methods:

a) Generalized Reduced Gradients (GRG)

b) Successive Quadratic Programming (SQP)

These two methods are currently regarded as the most efficient and robust nonlinear programming techniques. Other methods that handle nonlinear equalities, such as penalty functions (Fiacco and McCormick, 1968) or augmented Lagrangians (Bertsekas, 1976) require far too much computational effort and too many function evaluations to perform efficiently with process modules.

Among the generalized reduced gradient methods, the two most popular algorithms are GRG2 (Lasdon, et al., 1978) and MINOS (Murtagh and Saunders, 1978). Both use active set strategies and optimize the nonlinear function in the subspace of the active constraints. However, GRG2 requires converged equality constraints at each iteration* While this approach enforces stable algorithmic performance for equation-based chemical process optimization strategies (Lasdon 1981; Sarma and Reklaitis, 1982), it defeats the purpose of the infeasible path strategy. MINOS, on the other hand, does not need to follow a feasible path for convergence but works in linear subspaces. Also, Murtagh (1982) described how MINOS can be adapted to equation-based chemical process optimization. Performance on large-scale problems have been very efficient with this approach.

However, the optimization problem formulated for the infeasible path strategy has a different structure. First, it is small (about 10-50 variables) and has nonlinear equality constraints. Therefore, a full optimization in a linear subspace is probably too inefficient, especially in terms of function evaluations. Moreover, the characteristics of MINOS are best applied to large problems. Thus, we consider the SQP (Han, 1977; Powell, 1977) strategy.

Instead of optimizing a nonlinear function in the linear subspace of the active set, SQP constructs a quadratic objective function and linearizes the constraints. The resulting quadratic program (QP) can be solved easily and requires only one function and gradient evaluation. The QP also finds the active constraint set and determines a search direction for the next iteration. Several studies have verified (Lasdon, 1981; Powell, 1977; Schittkowski, 1982b) that the SQP algorithm requires very few function evaluations for nonlinearly constrained optimization problems, while the CPU time expended depends largely on the efficiency of the quadratic programming step.

SQP is thus ideally suited for infeasible path optimization because function and gradient calculations require flowsheet module calculations and represent the most time-consuming part of the optimization. However, the algorithm itself is still evolving; a few of its properties still require improvement.

As mentioned above, SQP solves a quadratic program at each iteration. Since the Hessian matrix of the QP is a positive definite approximation of the Hessian of the Lagrangian, the search direction is unique. However, taking full steps along this direction does not guarantee global convergence of the nonlinear problem. To ensure convergence, various methods have been proposed (Chamberlain, et al., 1982; Han, 1977; Powell, 1978; Schittkowski, 1982a). These find a stepsize along the search direction by minimizing a function that reflects the magnitude of the objective function and infeasibility of the constraints. As yet, however, each stepsize strategy still has its drawbacks.

Another problem lies in initializing and maintaining the accuracy of the approximated Hessian. While the quadratic program and the updating procedure are themselves scale invariant, the performance of the algorithm depends greatly on how the variables are scaled or, equivalently, how the Hessian is initialized. Only heuristic scaling strategies have been proposed so far (Biegler and Hughes, 1982; Chen, 1982). Further numerical studies such as recent ones by Schittkowski (1982b) and Stadtherr and Chen (1983) should help to resolve these problems.

2) Gradient Calculation Strategy. At each iteration, the QP used in the infeasible path strategy takes the form:

$$
\begin{aligned}
\text{(QP)} \quad \text{Min} \quad & \nabla\phi(x_i,y_i)^T d + \frac{1}{2} d^T B d \\
\text{s.t.} \quad & g(x_i,y_i) + \nabla g(x_i,y_i)^T d \le 0 \\
& h(x_i,y_i) + \nabla h(x_i,y_i)^T d = 0 \\
& c(x_i,y_i) + \nabla c(x_i,y_i)^T d = 0
\end{aligned}
$$

At present the gradients $\nabla\phi(x_i,y_i)$, $\nabla g(x_i,y_i)$, $\nabla h(x_i,y_i)$, $\nabla c(x_i,y_i)$ are evaluated by perturbing the flowsheet. Since the optimization problem can be written explicitly in terms of design variables, x, and retention variables (selected flowsheet parameters), r, gradient information is given by:

$$\nabla_x \psi = \frac{\partial\psi}{\partial x} + \frac{\partial r}{\partial x}\frac{\partial\psi}{\partial r}$$

$$\nabla_y \psi = \frac{\partial r}{\partial y}\frac{\partial\psi}{\partial r} \qquad \text{for } \psi = \phi, g \text{ or } c$$

and

$$\nabla_x h = -\,\partial w/\partial x$$

$$\nabla_y h = I - \frac{\partial w}{\partial y}$$

Since $\partial\psi/\partial x$ and $\partial\psi/\partial r$ can be evaluated explicitly, only $\frac{\partial r}{\partial x}$, $\frac{\partial r}{\partial y}$, $\frac{\partial w}{\partial x}$ and $\frac{\partial w}{\partial y}$ need be evaluated by perturbation. Two perturbation strategies (Biegler and Hughes, 1982; Chen, 1982; Hutchison, et al., 1983; Shivaram and Biegler, 1983;) have been tested so far. Both can be used to advantage in modular environments.

With direct loop perturbation, the recycle convergence block is replaced by an "optimization" block. Nothing else in the simulator executive need be changed and the same calculation order used for convergence is executed for perturbation. Here, design variables do not affect upstream units so only partial loop perturbations are required. The total number of block evaluations is thus given by:

$$NBE = \sum_{i=1}^{NTOT} \sum_{j=1}^{VCS} \ell_{ij}$$

where

(NCP+2)	-	number of stream elements (no. of component flows plus pressure and enthalpy).
NCS	-	number of connecting streams in calculation sequence.
ND	-	number of design variables
$\ell_{ij} (\geq 0)$	-	number of blocks from design or tear variable i to the jth terminus in the calculation sequence.
VCS	-	number of calculation sequence termini (locations of w_j or the last retention vector downstream of all w_j) which are downstream of variable i.
NT	-	number of tear streams
NTOT	-	(NCP+2)NT + ND

This procedure tends to be somewhat wasteful for design variable perturbations and nested loops but requires no additional interface or manipulations to calculate the gradient. Also most information flow reversals embedded in the flowsheet can be handled quite naturally with this procedure. Finally, gradients of highly nonlinear modules tend to behave smoothly because direct perturbation responses of w and r are used. This allows for consistent intermediate variable responses to a given perturbation size for y (Biegler and Hughes, 1982).

The second strategy uses chainruling to evaluate gradients after each module has been perturbed by itself. This is, in principle, the same as the condensation step used by Biegler and Hughes (1981). Here the number of block evaluations is:

$$NCS\ (NCP + 2) + ND$$

For flowsheets where design variables are at the beginning of the calculation sequence, this method is clearly more efficient. Also, since each module is perturbed independently less computational effort can be expected for intramodule convergence (Chen, 1982). However, this strategy is more difficult to implement on sequential modular simulators and special provisions must be made for information flow reversal. Chen (1982) applied this approach on a simultaneous modular simulator and demonstrated its superiority. On small problems solved with sequential modular simulators, Shivaram and Biegler (1983) also reported fewer block evaluations, but some additional CPU time was required for procedural overhead.

Both strategies require calculation loops that include design and retention variables. Determination of the calculation sequence is straightforward; tear streams should be chosen that break all loops with the fewest tear variables. This heuristic keeps the optimization problem small and allows all loops to be converged simultaneously.

Stream candidates that satisfy this criterion should then be chosen to minimize the effort of gradient calculation. This is especially important if direct loop perturbation is used since it influences the l_{ij} terms for the number of block evaluations.

From discussion of gradient calculation strategies, one must also consider efficient and accurate methods for evaluating modular Jacobians. This leads directly to the next section.

3) Attributes of Process Models. Since gradient information is presently constructed by perturbation, it is important to consider how much error results from this procedure. Here, the accuracy of the gradients is affected by higher order Taylor series terms and by noise resulting from iterative calculations subject to a finite tolerance.

Because tear streams are not converged at each iteration, the gradients are not affected by recycle tolerances. However, many intra-modular procedures require iterative calculations and thus the size of the perturbation becomes important. As the perturbation size increases, the error due to higher order Taylor series terms becomes significant. On the other hand, the response of a small perturbation may be corrupted by convergence noise.

Biegler and Hughes (1982) experimented with various perturbation sizes and found little difference provided that iterative calculations had tight tolerances. Hutchison, et al. (1983) chose perturbation sizes that balance and thus minimize the error due to noise and higher order terms. Crowe (1983) eliminated the noise problem entirely by simply executing a fixed number of iterations for the function evaluation and perturbation steps.

Though perturbation was used for Jacobian information, the chain-ruling strategy also allows inclusion of analytic modular Jacobians if they can be specified. Some units such as mixers and splitters have analytic Jacobians for mass and enthalpy balances. Others such as heaters, compressors and pumps may require only a few perturbations for this information. Of course, the most difficult and complex units, such as tubular reactors and rigorous separators, have Jacobians that are almost impossible to construct analytically. For this reason an approach that uses simplified models can be deceivingly attractive.

For example, in a more radical departure from evaluating the flowsheet, Jirapongphan, et al. (1980) describe an optimization strategy that uses rigorous process models at the outer iteration only to fit parameters for more simplified models. For the inner iteration the simplified nonlinear models are optimized for the next outer iteration. While this approach is very efficient because it reduces the number of modular evaluations, there are some disadvantages relating especially to optimization that must be considered.

First, it is not always possible to find suitable nonlinear simplified models for the more rigorous ones. For some complex process and physical property models, simplified correlations cannot describe their behavior accurately enough.

The second disadvantage is more insidious. Because the optimum is usually determined by the gradients that make up the Kuhn-Tucker conditions, it is probable that the optimum found by Jiraponghpan's algorithm is <u>not</u> the optimum of the rigorous process model. Instead, this algorithm finds the optimum of the simplified models at the point where properties calculated by the rigorous and simplified models match. The following example illustrates why this solution may not be the true optimum.

Consider the optimization problem pictured in Fig. 3. The rigorous optimization problem is given by:

$$\text{Min} \quad a^2 + x^2$$

$$\text{s.t.} \quad \alpha - (x^3 + x^2 + 1) - 0$$

$$\alpha \geq 0$$

If we want to approximate the property a (defined by x^3+x^2+1) by a simpler model: a = x+8 (where B is determined by matching a with the nonlinear model), then the "inside-out" problem becomes:

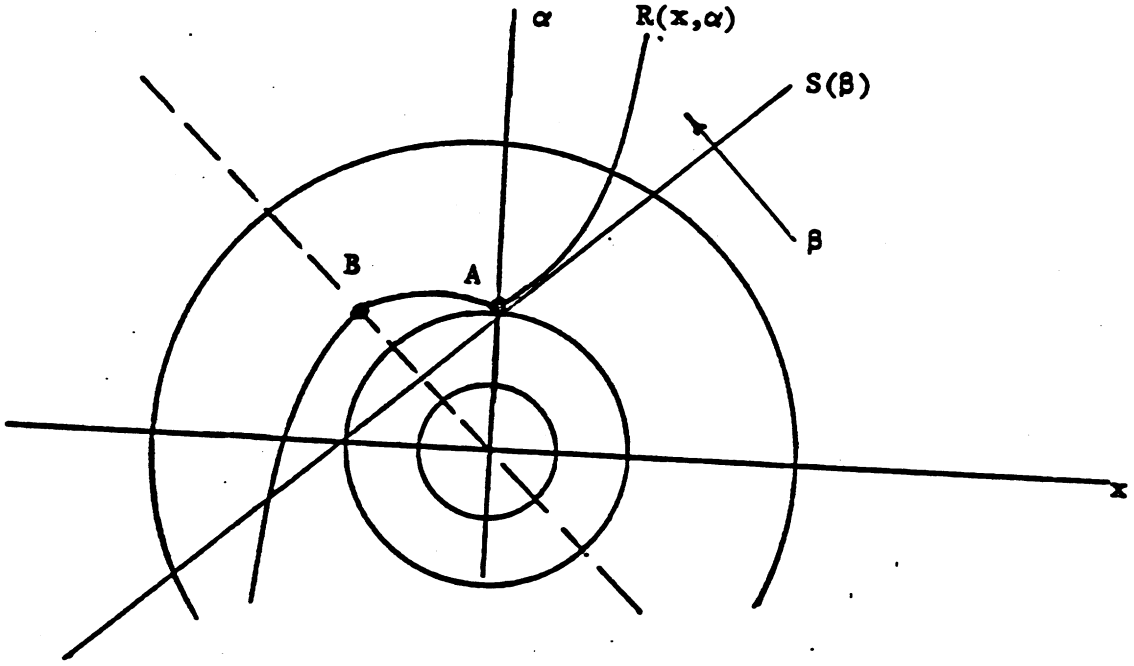


Figure 3. Simplified Models in Optimization

Outer iteration

$$\overline{\text{or}} \bullet (X + X + 1) \ll \overline{X} + 0$$

Inner problem

$$\bar{x} \bullet \arg \left\{ \begin{array}{l} \text{Min} \quad or^2 + x^2 \\ \text{s.t.} \quad a \bullet (x + B) = 0 \\ \qquad\qquad a \wedge 0 \end{array} \right\}$$

From Fig. 3, it can easily be seen that the optimum of the original problem is at point A. Starting from this point, the latter approach will actually move away toward point B. The broken line is the locus of optima found by the inner problem for different values of 8. At point B this locus crosses the equality constraint of the original problem.

It is interesting to note that, in this case, Jirapongphan's algorithm moves away from the true optimum (point A) to point B, which is actually a local maximum of the original problem. This shows that determination of the optimum by Kuhn-Tucker conditions is directly related to the accuracy of the gradients. Since many simplified models have different derivatives than their rigorous counterparts, different optima will be found.

This situation also applies to the use of simplified models for Jacobians in the chain-ruling procedure. Chen (1982) reported good results with simplified models if reasonably accurate approximations were made. However, he suggests further study before drawing any conclusions. Again, if simplified models have different gradients, they will probably lead to different "optimal^{11} solutions.

Instead, one should probably develop a framework that includes information that can be obtained analytically and uses the structure of the module to obtain missing information by perturbation. Only if the perturbations are prohibitive should simplified models then be considered. Even in this case, these models should match rigorous model gradients as well as their functions. This is the approach we are taking in refining our optimization strategy.

Lastly, one encounters the most difficult aspect about sequential modular optimization. Because the process modules often incorporate some level of decision-making in their calculation strategy (e.g. changing phases in flash calculations) as well as certain cost and sizing functions, we often encounter gradients and functions that are not uniquely defined at certain points.

In this case, the infeasible path strategy will calculate the wrong "gradients" and possibly jam into a corner and fail. Using smooth, simplified models may alleviate this problem but, as mentioned above, the solution will probably be far from the true optimum.

Here the biggest drawback is that these functions are hidden within process modules. Thus, we can only address those problems that are known to occur during the optimization. However, even if the source of these functions is known, there still are no easy ways of handling them.

Discontinuous functions can sometimes be considered by introducing binary (0 or 1) variables into the optimizations. The problem now becomes a mixed-integer nonlinear program which is difficult to solve, especially if many binary variables are present. Duran and Grossmann (1983), however, have developed an efficient strategy for MINLP's if the problem is separable for the binary variables and convex for the continuous variables.

Some nondifferentiable functions, if known, can be represented by the epigraph of differentiable functions (Han, 1981):

$$\max_{i} \{e_i(z)\}$$

In an optimization problem these can be incorporated by noting the equivalence

6

$$\max_{i} \{e_i(z)\} > \Longleftrightarrow < e_i(z) \leq \delta$$

where δ is added as a new variable. Now, nondifferentiable objective functions can be written as:

$$\begin{aligned} \text{Min}\quad & \delta \\ e_i(z) &\le \delta \\ g(z) &\le 0 \\ h(z) &= 0 \end{aligned}$$

and nondifferentiable inequality constraints can be given by:

$$\left.\begin{array}{rl} \text{Min} & \phi(z) \\ e_i(z) \le \delta & \\ \delta \le 0 & \\ h(z) = 0 & \end{array}\right\rangle \Longleftrightarrow \left\langle\begin{array}{rl} \text{Min} & \phi(z) \\ e_i(z) \le 0 & \\ h(z) = 0 & \end{array}\right.$$

Nondifferentiable equalities can be represented by:

$$\begin{aligned} \text{Min}\quad & \phi(z) \\ g(z) &\le 0 \\ \delta &= 0 \\ e_i(z) &\le \delta \end{aligned}$$

but <u>only</u> if we force one or more of the $e_i(z)$ to be active at the solution. The combinatorial problem encountered here can be difficult. This is related to a more insidious nondifferentiable problem given by:

$$\begin{aligned} \underset{x_1,x_2}{\text{Min}}\quad & \phi_1(x_1,x_2) \\ \text{s.t.}\quad & g_1(x_1,x_2) \le 0 \\ & h_1(x_1,x_2) = 0 \\ \underset{x_2}{\text{Min}}\quad & \phi_2(x_1,x_2) \\ \text{s.t.}\quad & g_2(x_1,x_2) \le 0 \\ & h_2(x_1,x_2) = 0 \end{aligned}$$

This problem is typical of process optimization where the inner minimization may represent an equilibrium calculation (minimize Gibbs[1] free energy). Clark and Westerberg (1982) addressed this problem by writing the Kuhn-Tucker conditions for the inner problem and applying active set and relaxation strategies to the inner problem inequalities.

These types of problems have been studied in equation-solving environments. They are just as frequent in sequential modular modes and should be recognized when formulating the optimization problem. Easy ways of handling them are to restrict the region of investigation so that discontinuities are not encountered (Biegler and Hughes, 1983) (e.g. compressors must have positive Ap) or by fixing calculated variables that may exhibit discontinuous behavior as input parameters or design variables. One example is stream pressure which can be nondifferentiable around the loop and should thus be removed from the tear set, y, in the optimization problem (Biegler and Hughes, 1982).

CONCLUSIONS AND SIGNIFICANCE

Current strategies for simultaneous modular simulation and optimization have been reviewed and compared.

Simultaneous modular simulation arose because of the need to solve flowsheets more efficiently and flexibly while using "black box[11] process modules. Its development closely parallels that of equation-based simulators in the following areas:

1) Flowsheet decomposition algorithms that find "optimal" tear sets for recycle and control loops and converge them simultaneously.

2) Gradient approximation strategies that allow simultaneous solution by Newton's method.

3) Use of approximate nonlinear models that represent the flowsheet in a simplified form and can be solved using equation-based strategies.

The optimization problem can be regarded as an extended simulation problem where optimization and recycle convergence occur simultaneously. This infeasible path strategy has worked well in equation-based modes (Berna et al. (1980)) as well as with the simultaneous modular approach. The main points of this approach can be summarized as:

1) The SQP algorithm is appropriate for this strategy because it can handle small sets of nonlinear equalities efficiently. Some work, however, still needs to be done to improve performance and robustness.

2) Two types of gradient calculation strategies can be used for the SQP algorithm:

   a) Direct loop perturbation is easy to implement on any sequential modular simulator but may be inefficient for design variable perturbations.

   b) Chain-ruling usually requires fewer perturbations but extensive modification of the simulator's executive is needed for implementation.

   The choice depends on the structure of the simulator.

3) The flowsheet should be decomposed by choosing the fewest variables that tear all loops. This keeps the optimization problem small. If several tear candidates exist, the ones that minimize the effort of gradient calculation should be chosen.

4) The only current reliable way of evaluating gradients is through perturbation unless analytic information is known. Here, use of simplified models may cause convergence to the wrong solution.

5) Nondifferentiable and discontinuous functions embedded in process modules remain serious obstacles to efficient process optimization strategies. There are some ways of overcoming them if the functions can be identified.

As mentioned at the beginning of this paper, the simultaneous modular strategy bridges the gap between reliable and detailed sequential modular models and efficient equation-based solution strategies. As such, it has immediate applicability to real problems while using state-of-the-art solution strategies. Therefore, this mode will play a very important role in process engineering in the near future. Further improvements in this area will enhance its already high level of flexibility and applicability.

Acknowledgement

Financial support from the National Science Foundation under Grant #CPE-8204366 is gratefully acknowledged.

LITERATURE CITED

Aris, R., Discrete Dynamic Programming, Blaisdell, New York (1964).

Ballman, S.H. and J.L. Gaddy, "Optimization of Methanol Process by Flowsheet Simulation," IEC Proc. Des. Dev., 16, No. 3, 337 (1977).

Barkeley, R.W. and R.L. Motard, "Decomposition of Nets," Chem. Eng. J., 3, 265 (1972).

Barrett, A. and J.J. Walsh, Computers and Chem. Eng., 2, 397 (1979).

Berger, F. and F.A. Perris, "FLOWPACK II - A New Generation of Process Flowsheeting Systems," Computers and Chem. Eng., 3, 309 (1979).

Berna, T.J., M.H. Locke and A.W. Westerberg, "A New Approach to Optimization of Chemical Processes," AIChE J., 26, No. 1, 37 (1980).

Bertsekas, D., "Multiplier Methods: A Survey," Automatica, 12, 133 (1976).

Biegler, L.T. and R.R. Hughes, "Approximation Programming of Chemical Processes with Q/LAP," Chem. Eng. Prog., 77, No. 4, 76 (1981).

Biegler, L.T. and R.R. Hughes, "Infeasible Path Optimization of Sequential Modular Simulators," AIChE J., 28, No. 6, 994 (1982).

Biegler, L.T. and R.R. Hughes, "Process Optimization: A Case Study Comparison," Computers and Chem. Eng., to appear (1983).

Blau, G.E., "Session Summary: Nonlinear Programming," in Foundations of Computer-Aided Chemical Process Design (Mah and Seider, ed«), p.219, Engineering Foundation, New York (1980).

Boston, J.F., "Inside-Out Algorithms for Multicomponent Separation Process Calculations," in Comp. Applications to Chemical Engineering (Squires and Reklaitis, ed.), ACS Monograph (1980).

Boston, J.F. and H.I. Britt, "A Radically Formulation and Solution of the Single Stage Flash Problem, Computers and Chem. Eng., 2, 109 (1978).

Boston, J.F. and S.L. Sullivan, Can. J. Chem. Eng., 52, 52 (1974).

Brosilow, C and L. Lasdon, AIChE-IChE Symposium Series, #4, 75 (1965).

Broyden, C.G., "A Class of Methods for Solving Nonlinear Simultaneous Equations," Math. Comp., 19, 577 (1965).

Chamberlain, R.M., M.J.D. Powell, C. LeMarechal and H.C. Pedersen, "The Watchdog Method for Forcing Convergence in Algorithms for Constrained Optimization," Math. Prog. Study, 16, 1 (1982).

Chen, H.-S., Ph.D. Thesis, University of Illinois, Urbana (1982).

Chimowitz, E.H., S. Macchietto, T.F. Anderson and L.F. Stutzman, "Local Models for Representing Phase Equilibria," I&EC Proc. Des. Dev., to appear (1983).

Clark, P.A. and A.W. Westerberg, "Optimization for Design Problems Having More than One Objective," Process Systems Engineering Symposium, Tokyo (1982).

Crowe, C., personal communication (1983).

Duran, M.A. and I.E. Grossmann, "An Efficient Algorithm for a Special Class of Mixed-Integer Nonlinear Programs," TIMS/ORSA Meeting, Chicago (1983).

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

Friedman, P. and K.L. Pinder, "Optimization of a Simulation Model of Chemical Plants," I&EC Proc. Des. Dev., 11, No. 4, 512 (1972).

Gaines, L.D. and J.L. Gaddy, "Process Optimization by Flowsheet Simulation," I&EC Proc. Des. Dev., 15, No. 1, 206 (1976).

Genna, P.L. and R.L. Motard, "Optimal Decomposition of Process Networks," AIChE J., 21, No. 4, 656 (1975).

Gill, P.E., W. Murray and M.H. Wright, Practical Optimization, Academic Press, London (1981).

Grossmann, I.E. and M. Morari, "A Dialogue on Resiliency, Flexibility and Operability," this conference (1983).

Gundersen, T. and T. Hertzberg, "Partitioning and Tearing Chemical Process Flowsheets," Process Systems Engineering Symposium, Tokyo (1982).

Han, S-P, "A Globally Convergent Method for Nonlinear Programming," J. Optimization Theory and Control, 22, No. 3, 297 (1977).

Han, S-P., Math Programming, 20, 1 (1981).

Hughes, R.R., "Optimization Methods for Block Simulation," presented at VI Interamerican Congress of Chemical Engineering, Caracas, Venezuela, July 1975.

Hutchison, H.P., S. Kaijaluoto and W. Morton, "Process Optimization Using a Serial Cyclic Flowsheet Simulator," 3rd International Congress on "Computers and Chemical Engineering," Paris (1983).

Isaacson, R.A., Ph.D. Thesis, University of Wisconsin, Madison (1975).

Jackson, R., "Some Algebraic Properties of Optimization Problems in Complex Chemical Plants," Chem. Engr. Science, 19, 19 (1964).

Jirapongphan, S., J.F. Boston, H.I. Britt and L.B. Evans, "A Non-linear Simultaneous Modular Algorithm for Process Flowsheet Optimization," presented at 80th AIChE Meeting, Chicago (1980).

Jung, B.S., W. Mirosch and W.H. Ray, "A Study of Large Scale Optimization Techniques," 71st National AIChE Meeting, Dallas, XX (1972).

Klernes, J., J. Lutcha and V. Vavek, "Recent Extension and Development of Design Integrated Systems - DIS," Computers and Chem. Eng., 3, 357 (1976).

Komatsu, S., "Application of Linearization to Design of a Hydro-dealkylation Plant," I&EC Oper. Res. Symp., 60, No. 2, 36 (1968).

Lasdon, L.S., "A Survey of Nonlinear Programming Algorithms and Software," in Foundations of Computer Aided Process Design (Mah and Seider, ed.), p.185 (1981).

Lasdon, L.S., A.D. Waren, P. Jain and M.W. Ratner, ACM Trans, on Math Software, 4, 34 (1978).

Leesley, M.F. and G. Heyen, Computers and Chem. Eng., 1, 109 (1977).

Linnhoff, B., "Entropy in Practical Process Design," in Foundations of Computer Aided Process Design (Mah and Seider, ed.), Vol. 2, p.537 (1981).

Lutcha, J., J. Klemes, J. Klemsa, V. Vasek, M. Dohnal, and C. Verroouzek, Computer Aided Design, 2, No. 4, 229 (1975).

Mah, R.S.H., "Application of Graph Theory to Process Design and Analysis," Process Systems Engineering Symposium, Tokyo (1982).

Mahalec, V., H. Kluzik and L.B. Evans, "Simultaneous Modular Algorithm for Steady State Flowsheet Simulation and Design," CACE '79, EFCE, Montreux (1979).

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

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

Morari, M., "Flexibility and Resiliency of Process Systems," Process Systems Engineering Symposium, Tokyo (1982).

Motard, R.L., M. Shacham and E.M. Rosen, "Steady State Process Simulation," AIChE J., 21, No. 3, 417 (1975).

Murtagh, B.A., "On the Simultaneous Solution and Optimization of Large-Scale Engineering Systems," Computers and Chem. Eng., 6, No. 1, 1 (1982).

Murtagh, B.A. and M.A. Saunders, Math Prog., 14, 41 (1978).

Nagiev, M.F., Chem. Eng. Prog., 53, 297 (1957).

Nishimura, H«, Y. Hiraizumi and S. Yagi, Kagaku Kogaku, 31, No. 2, 183 (1967).

Parker, A.L. and R.R. Hughes, "Approximation Programming of Chemical Processes - 1," Computers and Chem. Eng., £» No- 3> *23 (1981).

Perkins, J.D., "Efficient Solution of Design Problems Using a Sequential Modular Flowsheeting Programme," Computers and Chem. Eng., 3, 375 (1979).

Perkins, J.D., "Equation Oriented Flowsheeting," this conference (1983).

Pho, T.K. and L. Lapidus, "Topics in Computer Aided Design. Part I - An Optimal Tearing Algorithm for Recycle Streams," AIChE J., 1£, No. 6, 1170 (1973).

Pierucci, S.J., E.M. Ranzi and G.E. Biardi, "Solution of Recycle Problems in a Sequential Modular Approach," AIChE J., 28, No. 5, 820 (1982).

Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," presented at the 1977 Dundee Conference on Numerical Analysis (1977).

Powell, M.J.D., "Algorithms for Nonlinear Constraints that Use Lagrangian Functions," Math Prog., 14, No. 224 (1978).

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

Rosen, E.M., "Steady State Chemical Process Simulation: A State-of-the-Art Review," in Computer Applications to Chemical Engineering (Squires and Reklaitis, ed«), p.3, ACS Monograph 124 (1980).

Sargent, R.W.H., "Rapporteur's Review: Flowsheeting," Computers and Chem. Eng., 3, 17 (1979).

Sargent, R.W.H., "A Review of Optimization Methods for Nonlinear Problems,[11] in Comp. Applications to Chem. Eng. (Squires and Reklaitis, ed.), ACS Monograph (1980).

Sarma, P. and G.V. Reklaitis, "Optimization of a Complex Chemical Process Using an Equation Oriented Model," Math Prog. Study, 20, 113 (1982).

Schittkowski, K., "The Nonlinear Programming Method of Wilson, Han and Powell with an Augmented Lagrangian Type Line Search, Part 1: Convergence Analysis," Numer. Math., 38, 83 (1982b).

Schittkowski, K., "The Nonlinear Programming Method of Wilson, Han and Powell with an Augmented Lagrangian Type Line Search, Part 2: An Efficient Implementation with Linear Least Squares Subproblems," Numer. Math., 38, 115 (1982a).

Schubert, L.K., "Modification of A Quasi-Newton Method for Nonlinear Equations with a Sparse Jacobian," Math Comp., 24, 27 (1970).

Shivaram, S. and L.T. Biegler, "Improved Infeasible Path Methods for Sequential Modular Optimization," 3rd International Congress on "Computers and Chemical Engineering," Paris (1983).

Sood, M.K. and G.V. Reklaitis, "Solution of Material Balances: The Constrained Case," AIChE J., 25, 220 (1979).

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

Stadtherr, M.A. and H.S. Chen, "Numerical Techniques for Process Optimization by Successive Quadratic Programming," 3rd International Congress on "Computers and Chemical Engineering," Paris (1983).

Stephanopoulos, G., "Synthesis of Process Flowsheets: An Adventure in Heuristic Design or a Utopia of Mathematic Programming?" in Foundations of Computer Aided Process Design (Mah and Seider, ed.), Vol. 2, p.439 (1981).

Stephanopoulos, G. and A.W. Westerberg, "Use of Hestenes' Method to Resolve Dual Gaps in Engineering System Optimization," JOTA, 15, No. 3, 285 (1975).

Umeda, T. and M. Nishio, "Comparison Between Sequential and Simultaneous Approaches in Process Simulation," IEC Proc. Des. Dev., 11, No. 2, 153 (1972).

Upadhye, R.S. and A.E. Grens, "Solution of Decompositions for Chemical Process Simulation," AIChE J., 21, No. 1, 136 (1975).

Vasek, V., J. Klemes, C. Vermouyek and M. Dohnal, Collect. Czech., Chem. Commun., 39, 2772 (1974).

Vela, M.A., Hydrocarbon Processing, 40, No. 5, 247 (1961).

Westerberg, A.W., "Optimization in Computer Aided Design," in Foundations of Computer Aided Process Design (Mah and Seider, ed.), p.149 (1981).