STRATEGIES FOR SIMULTANEOUS-MODULAR FLOWSHEETING AND OPTIMIZATION

Mark A. Stadtherr and Hern-Shann Chen
Chemical Engineering Department
University of Illinois, Urbana, IL 61801

ABSTRACT

The performance of the simultaneous-modular approach for process flowsheeting and optimization is studied using several process simulation and process optimization problems. Results of numerical experiments involving techniques for computing a flowsheet-level Jacobian, for performing the line search in the Han-Powell method, and for scaling the optimization problem are presented. The simultaneous-modular approach, as implemented in our new program SIMMOD, is found to be very effective on both simulation and optimization problems.

SCOPE

The limitations of the traditional sequential-modular approach for process flowsheeting and optimization are today increasingly being recognized. For instance, the sequential-modular approach does not efficiently handle problems in which a number of design constraints are imposed ("controlled" simulations), nor is it at all well-suited to the solution of optimization problems. Thus 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. Here we concentrate on the latter.

While the simultaneous-modular approach is not a new concept, it has attracted significant attention recently. 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, especially for controlled simulation and optimization problems. We have recently developed (Chen and Stadtherr, 1983b) a general-purpose simulator, SIMMOD, based on the simultaneous-modular approach, 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.

For simulation and controlled simulation problems we employ a recently developed modification of Powell's dogleg method (Chen and Stadtherr, 1981) for solving the nonlinear equations on the flowsheet level. Three options for providing an initial flowsheet-level Jacobian are considered, namely full-block perturbation, diagonal-block perturbation (e.g., Mahalec et al., 1979), and direct difference approximation (e.g., Perkins, 1979).

For optimization problems, we follow an infeasible path approach, using the popular Han-Powell method to solve the nonlinear programming problem by successive quadratic programming. However, SQPHP (Chen and Stadtherr, 1983a), the nonlinear programming routine used by SIMMOD, contains several enhancements of the algorithm to make it more efficient and reliable in the context of the simultaneous-modular approach. Here we concentrate on the performance of two such enhancements, the use of the basic watchdog technique (Chamberlain et al., 1979) in the line search, and the use of an automatic scaling procedure for the variables and the objective function.

CONCLUSIONS AND SIGNIFICANCE

Numerical experiments using SIMMOD show that: 1. For performing derivative calculations, the full-block perturbation approach provides a good combination of efficiency and reliability, and is preferable to the more commonly used direct difference approximation technique. 2. The basic watchdog technique (Chamberlain et al., 1979) for the line search procedure in the Han-Powell method is very effective. 3. The automatic scaling procedure used in SIMMOD for optimization problems improves its reliability and usually its efficiency.

The simultaneous-modular approach is an attractive approach for flowsheeting and optimization problems, at least when the computational strategies incorporated in SIMMOD are used. For simulation problems, SIMMOD is competitive with and sometimes significantly better than the sequential-modular approach in terms of computational efficiency, and it is at least as reliable as well. For optimization problems SIMMOD is much better than the sequential-modular approach, and in fact provides a very significant improvement over other versions of the simultaneous-modular approach when it comes to computational speed.

BACKGROUND

Many of the theoretical aspects involved in implementing the simultaneous-modular approach have recently been discussed in detail by Chen and Stadtherr (1983b). Some of these aspects are briefly summarized here.

In solving a process flowsheeting problem there are basically three types of equations to be considered: 1. model equations, including process unit models and physical property models; 2. flowsheet connection equations that indicate how the units are connected together in the flowsheet; 3. specifications. As in the equation-based approach (e.g., Shacham et al., 1982; Stadtherr and Hilton, 1982), in the simultaneous-modular approach equations describing the entire process are solved simultaneously. In this sense the equation-based approach and the simultaneous-modular approach can be thought of as two extremes of the same basic idea. In the equation-based approach all process variables, internal and external, are treated as independent, and the actual rigorous model equations for each unit are used. Generally one can think of the computation as occurring on two levels, one in which the equations are linearized, and another in which the linearized equations are solved simultaneously and a correction step determined. 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 using the same unit modules use in the conventional sequential-modular approach. Thus there are again 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 variou 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.

Formulation of Problem

There are a number of possible problem formulations for the simultaneous-modular approach. In general these formulations can be pu into one of three catagories, which we denote as formulations I, II, a III.

In formulation I, all connecting streams are torn and treated as two separate streams, one input and one output stream. The formulatio of Mahelec et al. (1979) falls into this category, as does the formulation used by Jirapongphan (1980) for process optimization problems. In this formulation, each connecting stream is treated as tv separate streams, which results in an unnecessarily large system of equations. In formulation II, each connecting stream is treated as on input stream and the model equations are substituted into the stream connection equations. Using this formulation the number of flowsheet-level equations is reduced by almost 50%, compared to formulation I.

For typical applications, the number of equations generated usin 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 can be torn instead of all connecting streams. Streams that are not torn can be eliminated using corresponding stream connection equations. Various techniques employing a flowsheet-level Jacobian involving only tear stream variables have been suggested (e.g., Mahalec _et al._, 1979; Sood _et al._, 1979; McLane _et al._, 1979; Perkins, 1979; Metcalfe and Perkins, 1978). The IPOSEQ, RFV, and CFV optimization methods of Biegler and Hughes (1982a,1983) can also be thought of as using this formulation. 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 accelerated direct substitution is used, typically a Newton or quasi-Newton approach. Since in most applications, the number of tear stream variables is relatively small, the Jacobian for formulation III is usually a full matrix, and the flowsheet-level problem can be solved by full matrix techniques.

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. This is desirable because one of the attractions of the simultaneous-modular approach is that it is possible to implement it in connection with an existing sequential-modular simulator. SIMMOD, our implementation of the simultaneous-modular approach, is based on formulation III.

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, especially in comparison to the usually very robust, though slow, performance of the direct substitution approach. 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 for some problems, which may exacerbate the problem of reliability. Concerns 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 it is possible to eliminate costly "control" loops, the tradeoffs tend to swing more clearly in favor of the simultaneous-modular approach. It should be noted that, in part for this reason, some form of the simultaneous-modular approach, such as the use of quasi-Newton convergence blocks, is available, at least optionally, in some industrially used simulation programs.

At this point it should be emphasized that for simulation and controlled simulation problems we assume that the flowsheet has been partitioned into irreducible blocks of units, and that the blocks are solved one at a time in the appropriate precedence order. Acyclic parts of the flowsheet, corresponding to irreducible blocks containing only one unit, are handled in the usual sequential fashion. The simultaneous-modular formulation is applied only within the cyclic parts of the flowsheet (irreducible blocks containing more than one unit).

Jacobian Evaluation

The modified dogleg algorithm used by SIMMOD to solve the flowsheet-level equations uses Broyden's method to update the Jacobian; thus it requires a Jacobian calculation initially, but thereafter only when necessary to maintain a good rate of convergence. The efficiency of the simultaneous-modular approach depends to a considerable extent on how the Jacobian that is required, 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.

Figure 1 shows the information flow for an arbitrary module. For formulations I and II, a block of partial derviatives corresponding to each module must be calculated. 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 of each module and some variables in the retention vector with respect to the connecting input streams and free equipment parameters. There are many ways to calculate or approximate these partial derivatives for formulations I and II. Here, we consider two of them.

The most straightforward way to obtain the block of partial derivatives for each module is to use a standard 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 c is the number of components, 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. We refer to this approach as full-block perturbation.

The full-block perturbation technique requires many module calculations to determine the partial derivatives. To reduce the number of module calculations one may use a diagonal approximation of the Jacobian (e.g., Mahelec _et al._, 1979). This assumes that the mth elements of output stream vectors are only affected by the mth elements of input stream vectors. Note that this assumption is made only for iterations in which a Jacobian must be evaluated, i.e. the first iteration and perhaps some subsequent ones if necessary to maintain a good rate of convergence. To evaluate a Jacobian using this technique, the module i calculation must be performed $n_{ci} + n_{di} + 1$ times. We refer to this approach as diagonal-block perturbation. Since this will often provide a very poor approximation for reactor units, we follow Mahalec _et al._ and use full-block perturbation for reactor units.

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 on the flowsheet level, and the Jacobian of these equations usually has very few zeros in it. Again the most straightforward approach for obtaining the Jacobian is to compute it directly by difference approximation (e.g., Perkins, 1979). In this case one does not perform perturbation on each module individually, but instead performs perturbation on a sequence of modules. 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 and n_d are respectively the number of tear streams and the total number of free equipment parameters associated with 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. Alternatively one could instead perform perturbation on each module individually using either full-block or diagonal-block perturbation, and then simply use Euler's chain relation to obtain the flowsheet-level Jacobian required for formulation III. Thus the Jacobian for formulation III could be obtained by direct difference approximation, full-block perturbation, or diagonal-block perturbation. Results of studies comparing these three approaches are presented below. It should be noted that the use of Euler's rule in this context appears not to have been studied extensively until this study and one by Shivaram and Biegler (1983). The results of Shivaram and Biegler differ from those obtained here however, as discussed below.

SIMMOD--A Simultaneous-Modular Flowsheeting and Optimization Program

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 simulation problems as efficiently as ordinary simulation problems, and it can also solve process optimization problems very efficiently. The basic structure of SIMMOD has been described in detail elsewhere (Chen and Stadtherr, 1983b). Some of its more important features are listed briefly below:

1. User input is entered using a problem oriented language (POL)

in a logical order.

2. For simulation and controlled simulation problems, partitioning and tearing are performed using an algorithm (Chen and Stadtherr, 1983b) designed specifically for the simultaneous-modular approach.

3. For simulation or controlled simulation problems, the flowsheet-level system of equations is solved using NEQLU, a very efficient and reliable nonlinear equation solver that implements a modification of Powell's dogleg method (Chen and Stadtherr, 1981).

4. For optimization problems, the nonlinear programming problem is solved using SQPHP, a very efficient NLP routine, that implements an enhancement of the Han-Powell method (Chen and Stadtherr, 1983a). An infeasible path approach is used.

5. A small number of direct substitution iterations are performed to initialize the tear stream variables. While the numerical routines used have generally very good global convergence properties, we find this initialization procedure to provide an extra margin of safety that is needed on some problems.

6. The modules in SIMMOD are similar to those in sequential-modular systems. However, 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 previous calculations and saved for subsequent calculations of the same module. Also a flag is used to prevent the system from performing some unnecessary adiabatic flash calculations on module output streams. The user may write his own modules to perform special calculations.

7. Cost modules are included that perform sizing and costing to determine base costs, fixed investment costs, and utility costs.

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

9. For simulation and controlled simulation problems, the equations and variables are automatically scaled. For optimization problems, the objective function and the variables are automatically scaled.

The performance of SIMMOD on several simulation and optimization problems is discussed in detail elsewhere (Chen and Stadtherr, 1983c,d). We present here the results of three numerical experiments that compare different computational strategies for the simultaneous-modular approach.

COMPARISON OF JACOBIAN EVALUATION METHODS FOR SIMULATION PROBLEMS

In this section we use a number of simulation test problems to compare the three approaches considered for computing the Jacobian. We also compare the overall efficiency and reliability of the simultaneous-modular approach as implemented in SIMMOD with that of the sequential-modular approach.

Test Problems

Five benchmark problems and variations thereof are used in this study. Detailed descriptions of these problems are given elsewhere (Chen and Stadtherr, 1983c). The five basic problems and their variations are:

1. Problems 1a, 1b, and 1c all involve the well-known four flash-unit system studied by Cavett (1963). The difference between the three problem variations is the tear set used. In Problems 1a and 1b there are two tear streams; in Problem 1c there are three.

2. Problems 2a and 2b involve the cyclopentadiene recovery process used as Example 2 in the CHESS User's Guide (Motard and Lee, 1971). The computation time for this problem is dominated by a rigorous distillation module. The difference between the two problem variations is that in Problem 2b approximate physical property models are used in connection with the rigorous distillation module for evaluating the flowsheet-level Jacobian, while in Problem 2a rigorous models are used. The approximate models are two-constant, composition independent models for K values and enthalpy departures.

3. Problem 3 is the simple ethylene process used as Example 3 in the CHESS User's Guide.

4. Problems 4a and 4b involve the raw product recovery section of a natural gasoline plant, as taken from Example 4 in the CHESS User's

Guide. The difference between Problems 4a and 4b is the same as that between Problems 2a and 2b.

5. Problem 5 is a light hydrocarbons recovery process. This is exercise 25 in the FLOWTRAN exercise book (Clark, 1977).

Of the five problems studied, Problem 3 is a very easy problem for the sequential-modular approach, Problem 2 is slightly harder, while the remaining three problems are fairly difficult.

In solving these test problems, the following procedures are used in SIMMOD unless specified otherwise: 1. All thermodynamic properties are calculated using the Peng-Robinson equation of state with all binary parameters set equal to zero. 2. Before performing simultaneous-modular iterations on each irreducible block, we set all tear stream variables equal to zero, ignore free variables and design specifications, and perform three sequential-modular iterations. 3. The relative convergence tolerance in the module calculations is 10^{-6}, the relative convergence tolerance in the flowsheet-level calculation is 10^{-4}, and the perturbation factor in the Jacobian evaluation is 10^{-3}. 4. The computer used was a CDC Cyber 175.

Results

Some results comparing the performance of the three Jacobian evaluation techniques on these test problems are shown in Tables 1 and 2, which show the number of simultaneous-modular iterations required and the Jacobian evaluation time. We can make the following observations:

1. Calculating the Jacobian by diagonal-block perturbation is clearly faster than the other two alternatives, however it is not as reliable. Because it does not provide a particularly good approximation of the Jacobian it may take more iterations to converge (Problem 2), or it may fail (Problems 1 and 4). Sometimes, however, it works surprisingly well (Problems 3 and 5). Diagonal-block perturbation cannot be recommended for use in a general-purpose implementation of the simultaneous-modular approach.

2. Calculating the Jacobian either by direct difference approximation or by full-block perturbation gives reliable convergence, but the Jacobian evaluation time for the former is excessive. This is

different than the result obtained by Shivaram and Biegler (1983), who find that the full-block perturbation approach requires somewhat more CPU time. It appears however that this result was obtained because their implementation of full-block perturbation was not a particularly efficient one. Our recommendation is that the Jacobian be calculated by full-block perturbation. It is reliable and the Jacobian evaluation time is moderate.

3. In looking at the relative efficiency of full-block perturbation and direct difference approximation, two factors are primarily responsible for the difference: the number of perturbations required and the efficiency with which the perturbations can be performed. For the direct difference approximation case the number of perturbations required is roughly proportional to the number of tear streams in the current irreducible block, while for full-block perturbation the number of perturbations required is roughly proportional to the number of unknown connecting input streams per unit. This can be seen clearly in the results for Problem 1a. Here the computation time is dominated by the flash units, each of which has one connecting input stream; however there are two tear streams, so direct difference approximation requires twice as many perturbations of the flash units as full-block perturbation. This accounts for most of the difference in CPU time between the two approaches; the remaining difference is due to the better perturbation efficiency of the full-block perturbation approach, as discussed below. Also note that in Problem 1c there are three tear streams, while in Problems 1a and 1b there are only two, so when direct difference approximation is used the Jacobian evaluation time increases by roughly 50% on Problem 1c. When a chemical process to be simulated becomes larger or becomes more complicated, the number of tear streams will generally increase. On the other hand, since most flowsheets will comprise predominantly the same types of units, the number of connecting input streams is less likely to increase significantly, though there may be exceptions. Thus increasing problem size and complexity will generally degrade the performance of the direct difference approximation approach, while having a lesser effect on the full-block perturbation approach.

4. Even on problems for which full-block perturbation requires

more perturbations than direct difference approximation, full-block perturbation may be preferred because it can be implemented more efficiently. When block perturbation techniques are used to calculate the Jacobian, all the calculations for a given module are performed consecutively. This means that the internal variables of the module can be temporarily retained and easily utilized to reduce the CPU time for subsequent calculations of the given module. When direct difference approximation is used, calculations of the same module are not performed consecutively, and thus savings of this type cannot be fully realized; internal variables can be saved from one pass through the loop to the next (this was done here only on Problems 2a and 2b), but for complex units with large numbers of internal variables the I/O expense involved may be significant. This can be seen clearly on Problem 2a. Here the CPU time is dominated by the rigorous distillation module. When full-block perturbation is used 21 perturbations of the distillation module are required, while for direct difference approximation only 14 are required. Nevertheless, full-block perturbation still requires less CPU time, because the perturbation calculations can be made more efficiently. Even though in applying direct difference approximation on this problem, the internal variables for the rigorous distillation were saved from one perturbation to the next, the savings this made possible were largely offset by the I/O expense involved. Thus better perturbation efficiency is another advantage of the full-block perturbation approach, and accounts in part for the good performance shown in Table 2.

5. The use of simple physical property models for Jacobian evaluation in Problems 2b and 4b significantly decrease the Jacobian evaluation time, and has little or no effect on reliability. It is interesting to note that when direct difference approximation is used on Problem 4b the Jacobian is apparently somewhat less accurate than when full-block perturbation is used. This appears to be due to the fact that when direct difference approximation is used, results from the approximate models are passed downstream to other units, thus propagating some inconsistencies. Concerning the accuracy of the Jacobian, it is also worth noting that the use of full-block perturbation gives one control over the size and direction of the

perturbations to each unit, while in using direct difference approximation there is no such control over the perturbations entering the downstream units. In principle one might thus expect direct difference approximation to sometimes give a poor estimate of the Jacobian for this reason; however we have not observed this problem with direct difference approximation to date.

When the Jacobian is calculated by full-block perturbation, the simultaneous-modular approach as implemented in SIMMOD is competitive to or sometimes better than the usual sequential-modular approach in terms of reliability and computational efficiency. This can be seen in the results presented in Table 3. In preparing this table, the total computational time required for the simultaneous-modular approach, including initialization time, function evaluation time, Jacobian evaluation time, and NEQLU overhead time, was converted to an equivalent number of sequential-modular iterations by dividing by the average time for a sequential-modular iteration for each problem, as measured by the average time required by SIMMOD to make one pass through all the modules. The overhead time for NEQLU on these problems and for SQPHP on the optimization problems ranges from about 1 to 15 percent of the total time. The FLOWTRAN and CHESS results are taken from Rosen and Pauls (1978), Motard and Lee (1971), and Clark (1977), and are for the case in which a Wegstein acceleration procedure is used.

For Problem 1, Rosen and Pauls (1978) find that when the sequential-modular approach is used the convergence behavior of this problem can be very strongly affected by the tear set used, and also by the acceleration frequency used in the direct substitution solution scheme. For some combinations of tear set and acceleration frequency FLOWTRAN fails to solve the problem. As shown in the results for Problems 1a, 1b, and 1c, all of which use different tear sets, the simultaneous-modular approach is not sensitive to the tear set used, at least when block perturbation derivatives are used. Insensitivity to the tear set used is an advantage of the simultaneous-modular approach.

PERFORMANCE OF BASIC WATCHDOG TECHNIQUE ON PROCESS OPTIMIZATION PROBLEMS

In the usual implementation of the Han-Powell method, a single penalty function is used as the objective in the line search. As noted by Powell (1980), on some problems this approach may lead to a very slow rate of convergence. Many techniques (Mayne, 1980; Fletcher, 1981; Yamashita, 1982; Schittkowski, 1981; Chamberlain _et al._, 1979,1982) have been proposed for overcoming this problem. In SQPHP (Chen and Stadtherr, 1983a), the NLP routine used in SIMMOD, we use the basic watchdog technique of Chamberlain _et al._ (1979). In this method the primary line search objective is still the penalty function, but a secondary line search function, namely the Lagrangian, is added and used if necessary to maintain a good convergence rate. A generalized version of this technique has been described more recently (Chamberlain _et al._, 1982). In order to study the effect of using the basic watchdog technique we disable it for some runs with the following two optimization test problems.

Test Problems

Two optimization problems and variations thereof are used in this study. Detailed descriptions of these problems are given elsewhere (Chen and Stadtherr, 1983d). The test problems used are:

1. Problems 6a and 6b involve the two flash-unit system studied by Jirapongphan (1980). In Problem 6a the design variables are the flash temperatures and the common pressure of the two flash units. In Problem 6b the design variables are the pressure and the fractions vaporized in each flash.

2. Problems 7a and 7b involve the three flash-unit system studied by Jirapongphan (1980). The difference between the two problem variations is the choice of design variables, just as in the case of Problems 6a and 6b.

In solving these test problems, the following procedures are used in SIMMOD unless specified otherwise: 1. All thermodynamic properties are calculated using the Peng-Robinson equation of state with all binary parameters set equal to zero. 2. Before performing simultaneous-modular iterations on each irreducible block, we set all tear stream variables equal to zero, ignore free variables and design specifications, and perform three sequential-modular iterations. 3. The relative convergence tolerance in the flash module calculations is 10^{-10}, the relative convergence tolerance in the flowsheet-level calculation is 10^{-4}, and the perturbation factor in the Jacobian and gradient evaluation is 10^{-5}. 4. The Jacobian and gradient are generated by full-block perturbation. 5. The computer used was a CDC Cyber 175.

Results

Some results comparing the performance of SIMMOD with and without the use of the watchdog technique are shown in Table 4, which shows the number of simultaneous-modular iterations and number of line searches required, as well as the equivalent number of sequential-modular iterations, determined as described above. We can make the following observations:

1. By following the progression of the objective function toward it optimal value, it can be determined that the poor performance of SIMMOD without the basic watchdog technique is indeed due to the line search technique used. Most of the progress of the objective function toward its optimum occurs in the first several iterations, while very little improvement is made in the many iterations that follow. Although good correction steps are found in the quadratic programming subproblems for these latter iterations, full correction steps are not accepted due to the line search technique used.

2. The watchdog technique appears to be very effective. It is very easily and cheaply implemented and leads to a dramatic improvement in the rate of convergence on these problems.

3. On other process optimization problems, such as the Problems 8 and 9 described below, the use of the watchdog technique has little effect. However, since the use of the watchdog technique is very beneficial in some problems, and causes no loss of computational efficiency when it is not needed, we recommend its use in any general-purpose process optimization program using the Han-Powell method.

PERFORMANCE OF AUTOMATIC SCALING ON PROCESS OPTIMIZATION PROBLEMS

One advantage of the Han-Powell method is that constraints need not be scaled. However, the scaling of the objective function and of the variables will still affect the performance of the method, perhaps significantly (Chen and Stadtherr, 1983a). In SIMMOD we scale the objective function and variables by setting the initial Hessian approximation to be a non-identity diagonal matrix, the elements in which are found using simple heuristic techniques described in detail elsewhere (Chen and Stadtherr, 1983a,b). In order to study the effectiveness of the automatic scaling scheme used in SIMMOD, we disable the scaling procedure for some runs on three optimization test problems and variations thereof.

Test Problems

Problems 6a, 6b, 7a, and 7b are again used. Two additional problems, detailed specifications for which are given elsewhere (Chen and Stadtherr, 1983d), are also used. These two problems are:

1. Problem 8 is the ammonia synthesis optimization problem studied recently by by Parker and Hughes (1981) using a quadratic approximation programming (QAP) approach, and by Biegler and Hughes (1981) and Jirapongphan (1980) using the simultaneous-modular approach.
2. Problem 9 is the gasoline polymerization process studied by Friedman and Pinder (1972) and Gaines and Gaddy (1976), using the sequential-modular approach. This problem is not used in the scaling study but is included in a summary of the overall performance of SIMMOD.

The procedures used in SIMMOD on these two additional problems are the same as for Problems 6 and 7, except that the relative convergence tolerance for the module calculations is 10^{-6}, the perturbation factor in the Jacobian and gradient evaluations is 10^{-3}, and the relative convergence tolerance for the flowsheet-level calculation is 10^{-3}. Also, for Problem 8, ten sequential iterations are used to initialize the tear variables.

Results

Some results comparing the performance of SIMMOD with and without the use of the automatic scaling procedure are shown in Table 5. We can make the following observations:

1. The scaling procedure generally leads to an improvement in computational efficiency, but the improvements are quite moderate, and in one case there is actually a loss of computational efficiency.

2. The scaling procedure improves the reliability of SIMMOD, as can be seen by considering Problem 8. Without scaling the true optimum for Problem 8 was not found. As a further check, we ran the problem again using the solution from the unscaled run as the initial guess. Without scaling, the NLP rountine terminated in one iteration and claimed the solution. With scaling, the NLP routine performed four more iterations and converged to a solution representing roughly a 1% improvement in the objective function. This indicates that the solution found without scaling is not even a local minimum.

Finally, in order to consider the overall performance of SIMMOD on optimization problems, using both the watchdog technique and the automatic scaling procedure, we present Table 6, which compares the performance of SIMMOD with the reported performance of some other optimization programs used in other studies. Regarding these results we make the following observations:

1. To compare the CPU times given, we note that the speed of the CDC Cyber 175 used in this study is about three times that of a CDC 6600, and is about the same as an IBM 370/168. After taking these factors into account, the CPU time used by SIMMOD is still an order of magnitude less than that used in all the previous studies. One reason for the very short execution time is that the NLP routine SQPHP uses an enhanced version (Chen and Stadtherr, 1983a) of the Han-Powell method, and is significantly more efficient than the NLP routines used in other studies. Another factor is that SIMMOD does not perform any unnecessary adiabatic flash calculations on the output streams of modules. Also the different studies use different modules, different physical property routines, and different executive routines, making direct quantitative comparisons of CPU time difficult. What is clear is that using the

strategies incorporated in SIMMOD it is possible to implement the simultaneous-modular approach very efficiently on optimization problems.

2. As mentioned by Jirapongphan (1980), for Problem 8 a process simulation at optimal conditions using the sequential-modular simulator FLOWTRAN requires 63 sequential iterations. SIMMOD requires the equivalent of only about 60 sequential iterations to optimize the process. It should be noted that the results of Jirapongphan quoted here are for the case in which rigorous models were used in making derivative evaluations. Jirapongphan also quotes results indicating that efficiency can be improved by making use of simple, nonlinear, approximate models in this context.

3. A recent comparison (Biegler and Hughes, 1982b) of the IPOSEQ, CFV, and RFV methods on a propylene chlorination process indicates that the feasible path methods CFV and RFV are significantly more efficient than the infeasible path method IPOSEQ. As implemented in SIMMOD, the infeasible path approach appears to be very efficient. It is difficult to imagine that the performance of SIMMOD could be improved by using a feasible path approach. In our experience, however, it is possible that on some problems the overall performance may be improved by doing a few more sequential iterations during initialization. For instance, on Problem 8 the program fails if only three sequential iterations are used initially, but is successful if five sequential iterations are used. As the number of initial sequential iterations is increased from 5 to 15, the number of simultaneous-modular iterations needed to solve the problem is reduced from 6 to 4. It is also worth noting that Jirapongphan's solution of Problem 8 requires 18 simultaneous-modular iterations; this relatively large number indicates that the one-sequential-iteration initialization scheme he recommends is not suitable for this problem.

Acknowledgement

This work has been supported by the National Science Foundation under grant CPE 80-12428. Dr. Hern-Shann Chen is now at the Union Carbide Corporation, South Charleston, West Virginia.

LITERATURE CITED

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).

Biegler, L. T., and R. R. Hughes, "Process Optimization: A Comparative Case Study," pres. at AIChE Annual Meeting, Los Angeles (1982b).

Biegler, L. T., and R. R. Hughes, "Feasible Path Optimization with Sequential Modular Simulators," Comput. Chem. Eng., in press (1983).

Cavett, R. H., "Application of Numerical Methods to the Convergence of Simulated Processes Involving Recycle Loops," Amer. Petroleum Inst. Preprint No. 04-63 (1963).

Chamberlain, R. M., C. Lemaréchal, H. C. Pedersen, and M. J. D. Powell, "The Watchdog Technique for Forcing Convergence in Algorithms for Constrained Optimization," pres. at Tenth International Symposium on Mathematical Programming, Montreal (1979).

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

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).

Chen, H. S. and M. A. Stadtherr, "Enhancements of the Han-Powell Method for Successive Quadratic Programming," submitted for publication (1983a).

Chen, H. S., and M. A. Stadtherr, "A Simultaneous-Modular Approach to Process Flowsheeting and Optimization: I. Theory and Implementation," submitted for publication (1983b).

Chen, H. S., and M. A. Stadtherr, "A Simultaneous-Modular Approach to Process Flowsheeting and Optimization: II. Performance on Simulation Problems," submitted for publication (1983c).

Chen, H. S., and M. A. Stadtherr, "A Simultaneous-Modular Approach to Process Flowsheeting and Optimization: III. Performance on Optimization Problems," submitted for publication (1983d).

Clark, J. P., Exercises in Process Simulation Using FLOWTRAN, CACHE Corp., Cambridge, Mass. (1977).

Fletcher, R., "Second Order Corrections for Non-Differential Optimization," in Numerical Analysis, Dundee 1981, Lecture Notes in Mathematics, Vol. 912, Springer-Verlag, Berlin (1982).

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).

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

Mahalec, V., H. Kluzik and L. B. Evans, "Simultaneous Modular Algorithm for Steady State Flowsheet Simulation and Design, pres. at 12th European Symposium on Computers in Chemical Engineering, Montreux, Switzerland (1979).

Mayne, D. Q., "On the Use of Exact Penalty Functions to Determine Step Length in Optimization Algorithms," in Numerical Analysis, Dundee 1979, Lecture Notes in Mathematics, Vol. 773, Springer-Verlag, Berlin (1980).

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., and H. M. Lee, CHESS User's Guide, 3rd Ed., University of Houston (1971).

Parker, A. L., and R. R. Hughes, "Approximation Programming of Chemical Process, Part 1: Optimization of FLOWTRAN Models, Comput. Chem. Eng., 5, 123 (1981).

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 Programme," Comput. Chem. Eng., 3, 375 (1979).

Powell, M. J. D., "Optimization Algorithms in 1979," in Optimization Techniques, Lecture Notes in Control and Information Science, Vol. 22, Springer-Verlag, Berlin (1980).

Rosen, E. M., and A. C. Pauls, "Computer Aided Chemical Process Design: The FLOWTRAN System," Comput. Chem. Eng., 1, 11 (1977).

Schittkowski, K., "The Nonlinear Programming Method of Wilson, Han, and Powell with an Augmented Lagrangian Type Line Search Function," Numer. Math., 38, 883 (1981).

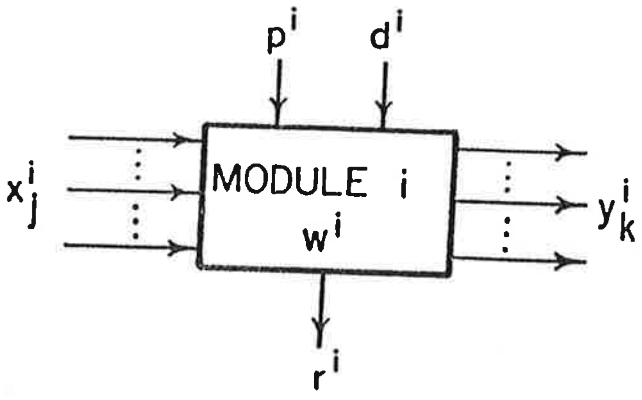
Shacham, M., S. Macchietto, L. F. Stutzman and P. Babcock, "Equation Oriented Approach to Process Flowsheeting," Comput. Chem. Eng., 6, 79 (1982).

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

Sood, M. K., R. Khanna and G. V. Reklaitis, "A Two Level Approach Exploiting Sparsity in Flowsheet Material Balancing," pres. at AIChE National Meeting, Houston (1979).

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

Yamashita, H., "A Globally Convergent Constrained Quasi-Newton Method with an Augmented Lagrangian Type Penalty Function," Math. Prog., 23, 75 (1982).



x^i_j	Stream vector for jth 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 kth output

Figure 1. Information Flow in an Arbitrary Module.

Table 1.

Number of Simultaneous-Modular Iterations Required for Three Jacobian Evaluation Techniques

Problem	Full-Block Perturbation	Direct Difference Approximation	Diagonal-Block Perturbation
1a	4	4	Fail
1b	5	5	--
1c	4	4	--
2a	2	2	10*
2b	2	2	10*
3	2	2	2*
4a	13	12	Fail
4b	8	14	Fail
5	3	3	3

*Full-block perturbation was used in reactor units instead of diagonal-block perturbation.

Table 2.

Jacobian Evaluation Time (Seconds) Required for Three Jacobian Evaluation Techniques

Problem	Full-Block Perturbation	Direct Difference Approximation	Diagonal-Block Perturbation
1a	0.819	2.006	Fail
1b	0.898	2.034	--
1c	0.867	3.104	--
2a	0.829	1.441	0.284*
2b	0.419	1.052	0.209*
3	0.490	0.830	0.201*
4a	8.855	11.939	Fail
4b	3.272	5.516	Fail
5	3.530	7.568	0.719

*Full-block perturbation was used in reactor units instead of diagonal-block perturbation.

Table 3.

Comparsion of Simultaneous-Modular Approach and Sequential-Modular Approach on Process Simulation Problems

Problem	Equivalent Number of Sequential-Modular Iterations Using SIMMOD with Full-Block Perturbation	Number of Sequential-Modular Iterations Using a Sequential-Modular Simulator
1a	25	23 (FLOWTRAN)
1b	27	48 (FLOWTRAN)
1c	28	
2a	13	22 (CHESS)
2b	9	
3	17	6 (CHESS)
4a	31	>40 (CHESS)
4b	18	
5	24	"many" (FLOWTRAN)

Table 4.

Effect of Using the Watchdog Technique on the Performance of the Simultaneous-Modular Approach for Process Optimization

Problem	No Watchdog			Watchdog		
	SI	LS	EI	SI	LS	EI
6a	30	70	297	25	30	147
6b	46	92	414	15	21	129
7a	42	91	395	21	37	194
7b	50*	122	462	15*	30	140

Notes: SI = number of simultaneous-modular iterations required.
LS = number of line searches required.
EI = equivalent number of sequential-modular iterations.
Automatic scaling was not performed.
*SQPHP terminated abnormally but returned a good solution.

Table 5.

Effect of Using an Automatic Scaling Technique on the Performance of the Simultaneous-Modular Approach for Process Optimization

	No Scaling			Scaling		
Problem	SI	LS	EI	SI	LS	EI
6a	25	30	147	11	14	100
6b	15	21	129	14	21	121
7a	21	37	194	25	53	248
7b	15[*]	30	140	16	22	133
8	5[†]	5	60	5	5	60

Notes: SI = number of simultaneous-modular iterations required.
LS = number of line searches required.
EI = equivalent number of sequential-modular iterations.
The watchdog technique was used in the line search.
[*]SQPHP terminated abnormally but returned a good solution.
[†]True optimum was not found (objective function value was 1% from the true optimum objective function value).

Table 6.

Comparison of SIMMOD and Other Optimization Studies

	SIMMOD		
Problem	Number of Equivalent Sequential-Modular Iterations	CPU Time (CDC Cyber 175)	Results of Other Studies
6a	100	1.7 sec.	13.3 sec.[1]
6b	121	2.3 sec.	
7a	248	5.7 sec.	108 sec.[1]
7b	133	3.7 sec.	
8	60	6.5 sec.	5.1 min.[1] 14.4 min.[2] 34.88 min.[3]
9	51	17.4 sec.	50 complete simulations[4]

Notes: The watchdog technique was used in the line search.
Automatic scaling of variables and objective function was used.
[1]Results of Jirapongphan (1980) on an IBM 370/168.
[2]Results of Biegler and Hughes (1981) using Q/LAP on an CDC 6600.
[3]Results of Parker and Hughes (1981) on an IBM 370/168.
[4]Results of Friedman and Pinder (1972).