RECENT PROGRESS IN EQUATION-BASED PROCESS FLOWSHEETING

Mark A. Stadtherr and James A. Vegeais
University of Illinois
Chemical Engineering Department
Urbana, IL 61801

ABSTRACT

The equation-based approach is an attractive alternative to the usual sequential-modular approach to process flowsheeting. In this paper we discuss the current status of equation-based flowsheeting and discuss recent progress in numerical methods for its implementation. We also discuss the development of techniques for implementing the equation-based approach in connection with advanced computer architectures, in particular vector machines such as the Cray-1 or Cray X-MP supercomputers.

BACKGROUND

In this section, we discuss the relationships between the different approaches to process flowsheeting, as well as some of their perceived advantages and disadvantages.

It should be noted at the outset that two comprehensive reviews of equation-based process flowsheeting [1,2] have appeared in recent years. The review by Perkins [2] is particularly valuable and covers many aspects of equation-based flowsheeting in more detail than possible here.

In its most fundamental form, the process flowsheeting problem can be regarded as one of solving a large system of nonlinear equations. The different approaches to process flowsheeting differ most fundamentally in their approach to solving this set of simultaneous equations. The equation system can generally be thought of as consisting of three types of equations: 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; and 3. specifications.

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

For a simulation problem involving one or more recycle streams, values for one or more streams must be guessed, or torn. The modules are then called in the sequence indicated by the connection equations until new values for one of the tear streams are generated. This amounts to using the model equations, connection equations, and simple specifications to generate a set of equations of the form $\underset{\sim}{x} = \underset{\sim}{f}(\underset{\sim}{x})$ for the tear stream values, which is then solved iteratively, usually by the method of substitution, either directly, or most often in connection with some sort of acceleration scheme. Thus we essentially have two levels of computation: a module level, in which the module computations are performed, and a flowsheet level, in which direct or accelerated substitution is used to converge the tear streams. (As used here we restrict the term SeqM to this traditional type of iteration scheme. Other types of flowsheet-level iteration schemes are available, but we prefer to categorize their use under the term "simultaneous-modular" as discussed in more detail below.)

When one or more design specifications are involved, a difficulty arises. These equations cannot be handled on the module level because the specified values cannot be passed directly to the modules as inputs, nor can they be handled directly on the flowsheet level, since specifications are not naturally in the form $x = f(x)$. In the SeqM approach this sort of problem must be handled by an iterative simulation in which the process is repeatedly simulated until the design specifications are met. The iteration loops so generated are often referred to as "control" loops.

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

Unfortunately, the SeqM approach also has a reputation for being very slow and costly, for two fundamental reasons. The first of these is the need

to handle design specifications, typically equations of the form x = constant, by introducing additional iteration loops; this is obviously an exceedingly inefficient way to handle such simple equations. The second reason can be explained by noting that the control loops needed to handle any design specifications may represent the outermost loops in a hierarchy of nested iteration loops. Immediately inside the control loops are the substitution loops used to converge the tear streams. Within these loops are any iteration loops that may be needed within the process unit modules, and at the innermost level any loops that may exist within the physical property subroutines called by the unit modules. Such systems of nested iteration loops can be very expensive and time consuming to solve. Finally, it is important to note that the SeqM approach is not well suited to process optimization, since this adds yet another outer level of iteration.

Because of the fundamental problems described above, there has been considerable recent interest in alternative approaches to process flowsheeting, namely the equation-based (EB) approach and the simultaneous-modular (SimM) approach. As discussed in more detail below, it is interesting to note that these two approaches can be regarded as two extremes of the same basic idea.

In the EB case, the process unit model equations, connection equations, and specifications are treated as constituting one very large system of nonlinear equations to be solved simultaneously. Though there are different ways of formulating the equations, the central problem is still the solution of a very large and sparse system of nonlinear equations.

By solving most or all of the equations describing a process simultaneously, one can avoid the drawbacks associated with the SeqM approach. When all equations are solved simultaneously there need be no nested iteration loops. Design specifications are now just very simple equations within the large system, and are almost trivial to handle. And since the simultaneous equations can be used as constraints in a generalized nonlinear programming problem, this approach has great potential for process optimization.

Today at least six EB flowsheeting systems in various stages of development exist. Two, SPEEDUP (Imperial College) [e.g., 3] and TISFLO II (Dutch Mines) [e.g., 4] have received considerable commercial use or testing. The others, ASCEND II (Carnegie-Mellon) [e.g., 5], QUASILIN (University of Cambridge) [e.g., 6], FLOWSIM (University of Connecticut/Control Data) [1], and SEQUEL (University of Illinois) [7] are perhaps best regarded as experimental prototype systems.

As in the EB approach, in the SimM approach the equations describing the entire process are solved simultaneously. In this sense the EB approach and the SimM approach can be thought of as two extremes of the same basic idea. In the EB approach the actual rigorous model equations for each unit are used, while in the SimM approach simple, usually linear, models of the units are used, the coefficients in which are generated by perturbation of the same unit modules used in the conventional SeqM approach. Thus there are two levels of computation, a module level in which the modules are used, perhaps together with some connection equations, to generate an approximate Jacobian for the process, and a flowsheet level in which these equations are solved simultaneously together with the specification equations and some or all of the connection equations. The various techniques proposed for the SimM approach differ in how the approximate Jacobian is generated on the module level, 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. One can include in this category the methods sometimes referred to as SeqM but which do not use the usual accelerated substitution scheme on the flowsheet level (there is a nomenclature problem here--such methods are sometimes referred to as SeqM and sometimes as SimM). For instance one can regard the use of Broyden convergence blocks in programs such as ASPEN, as being a rudimentary SimM feature in a basically SeqM program.

To some extent the SimM approach can be regarded as an attempt to combine some of the good features of both the SeqM approach and the EB approach. Since design specification equations can be handled directly on the flowsheet level, there is no need for costly "control" loops to converge the design specifications as required in the SeqM approach. And although there is still some nesting of iteration loops, the problem of slow convergence can be greatly reduced provided an efficient and reliable method is used to solve the flowsheet-level equations. Since the approximate Jacobian generated on the module level is smaller than that required by the EB approach, storage requirements are less, though, depending on the details of the approach used, sparse matrix methods may still be needed.

It is perhaps more useful, however, to regard the SimM approach as a special case of the EB approach [2], or vice versa, since as mentioned above, the EB approach and the SimM approach can be thought of as two extremes of the same basic idea. In each case, one in effect solves the model equations simultaneously with the connection and specification equations. Generally in the fully EB case, each individual model equation is linearized on one level of the computation (the "equation level"), and then on the next level (flowsheet level) all the equations are solved simultaneously and a correction step determined. In the SimM case, modules or procedures for solving the model equations are linearized or otherwise approximated on one level (module level), and these approximate models then solved simultaneously with the other equations on the next level (flowsheet level) to get a correction step. Thus the only basic difference between the two extremes is whether the approximations used on the flowsheet level are determined from the individual model equations or from procedures solving sets of model equations.

RECENT DEVELOPMENTS

In this section, we discuss recent developments in four areas related to EB flowsheeting. Developments in other areas have been discussed in more detail by Perkins [2].

Industrial Experience

In comparison to the SeqM approach, the EB approach offers much greater speed and flexibility in making specifications, especially when dealing with complex processes and controlled simulations. These potential advantages are generally recognized today, and there is increasing industrial interest in this approach, as discussed in more detail below. However, a number of problems have also been cited that have to date hindered the widespread industrial adoption of

the EB approach.

Perhaps the most serious problem from a fundamental standpoint is that the EB approach has acquired a poor reputation for reliability. This is due in part to the difficulty in providing a good initial guess for all the variables, and also to the need to use a general-purpose equation solver, as opposed to the special-purpose methods that can be applied within the computational modules of the SeqM approach to handle unusually difficult problems.

Some other problems of a less fundamental nature are also cited in connection with the EB approach. For instance, the very flexibility of this approach makes it easier for the user to make inconsistent specifications and makes error checking harder. These and other cited problems that can be categorized as a lack of friendliness to the user are in general not fundamental in nature and should disappear as better "software engineering" is applied to the development of EB packages. A common misconception regarding the EB approach is that the programs implementing it are not modularly organized and thus will be hard to maintain or to make additions to. This however is not generally true. The typical EB program comprises a number of modules, including a library of process unit modules in much the same fashion as the SeqM approach. The key difference is that in the EB approach the unit modules are used to generate equations while in the SeqM approach the modules are actually special-purpose equation solvers. This also means that the addition of new process unit models is in fact quite easy, since one needs only generate the model equations, not worry about how they are to be solved.

It should be noted that at least part of the reluctance of industry to adopt this approach is that they have a considerable investment in SeqM software, and are unwilling to take the risk or expense of trying something completely different. This situation seems to be changing somewhat however.

For instance, the EB program TISFLO-II was developed at Dutch State Mines, and has been used successfully in industrial applications, including simulation of an ammonia plant [8], optimization of utility generation and supply to a chemical complex [4], and olefin plant optimization [9].

Exxon [10] has recently conducted an evaluation of the general-purpose EB package SPEEDUP. They found SPEEDUP to be flexible and easy to use, but that problems of the sort discussed above were still not completely resolved. The most fundamental problems found were that convergence was not always reliable unless a fairly good initial guess was supplied, and that the program was inefficient. The latter is particularly disappointing; however it should be pointed out that the inefficiency seems to be due primarily to overhead in the input translator, and not to the basic computational strategies used.

Finally, it should be noted that some special-purpose EB programs are in industrial use, primarily for utility system calculation [e.g., 11] and systems of multistage separation units [e.g., 12]. It appears however that further research in EB flowsheeting is needed before truly general-purpose EB systems achieve widespread industrial use.

Inclusion of Procedures

As discussed above, the EB approach and the SimM approach can be thought of as two extremes of essentially the same basic idea. It is easy to imagine a sort of generalized non-SeqM approach in which some model equations would be used individually and others handled using procedures. In fact a whole range of such possibilities exist between the EB and SimM ends of the scale. Such an approach offers more speed and flexibility than the SimM approach alone, and since the equations handled in procedures could be solved using special-purpose algorithms that permit the user to tune individual solution algorithms and handle problem-specific constraints, it also offers improvements in reliablity in comparison to the EB approach alone.

An important trend in EB programs today is to provide this capability for including procedures. Macchietto [13] and Field et al. [6] have recently discussed several aspects of solving the mixed systems of equations and procedures that arise when procedures are included. It seems likely that as work proceeds on adding procedures to EB programs, and perhaps adding more EB features to SimM programs, the two approaches will gradually move closer and perhaps at some point meet. The question of where along the scale this meeting should be, if indeed there should be a meeting at all, is unresolved.

Nonlinear Equation Solving

The central computational problem in EB flowsheeting is the solution of a very large nonlinear equation system. While initial work in EB flowsheeting typically involved solution by tearing to reduce the number of variables iterated on, the recent trend is toward solution by simultaneous linearization. In this case all the equations are linearized and all the variables iterated on simultaneously, using Newton-Raphson or some related technique. We discuss here two current issues, reliability and Jacobian evaluation.

If Newton-Raphson is used, the need at each iteration to evaluate the Jacobian, either analytically or by finite difference, may be very time consuming. In fact, in their evaluation of SPEEDUP, Gupta et al. [10] found that finite-difference Newton-Raphson was always quite slow, and thus they used this method very little. So there is considerable motivation for the use of sparse quasi-Newton methods in which an approximation to the Jacobian is updated at each iteration. One such method is Schubert's method; however several authors have found this to be unreliable. Gupta et al. found that a quasi-Newton method proposed by Paloschi [14] gave better results than Schubert's method. However, this may be due partly to the fact that the implementation of Paloschi's method in SPEEDUP uses partitioning to reduce the size of the equation systems to be solved, while the implementation of Schubert's method does not. In general the quasi-Newton methods appear to require many more iterations than Newton-Raphson, so there is essentially a trade-off between number of iterations and time per iteration.

In order to try to combine the good features of both quasi-Newton methods and Newton-Raphson, Lucia and Macchietto [15], Lucia and Westman [16], Miller and Lucia [17], Lucia et al. [18], Westman et al. [19], Macchietto [13], and Field et al. [6] suggest the use of hybrid methods in which easily evaluated partial derivatives, or terms in partial derivatives, are computed analytically, while those derivatives or terms that are more difficult to evaluate are approximated by a quasi-Newton update. It is still

not clear what mix of exact and approximated derivative information provides the best combination of overall speed and reliability. Such a mix is likely to vary, perhaps significantly, from problem to problem.

The other issue of interest here is reliability. One approach to improving reliability is to provide better automatic initialization schemes. These are largely empirical in nature, so a variety of such schemes are possible. One possibility, suggested most recently by Gupta et al. [10] is to do a few SeqM iterations to initialize the variables. This may be particularly attractive if the trend toward including more procedures in EB systems continues to develop.

Another approach is to try to improve the global convergence properties of the nonlinear equation solver used. For example, Chen and Stadtherr [20] have shown recently how the use of Powell's dogleg [21] technique for computing a correction step can be used to improve the reliability of Schubert's update. A class of methods with generally excellent global convergence properties is the class of so-called homotopy (or continuation) methods. The use of such methods in EB problems has been discussed recently by Wayburn and Seader [22], Seader et al. [23], and Wayburn and Seader [24]. One way of looking at such methods is to think of them as generating a sequence of initial guesses until eventually a good enough one is produced. In this sense the homotopy methods are similar in principle to the evolutionary approach for problem initialization suggested by Locke [5]. The efficiency of the homotopy approach essentially depends on how long the sequence of initial guesses needs to be. Hlavacek [25] has reported that in his experience a relatively small such sequence is needed in some chemical engineering applications. There are apparently no published comparisons of homotopy to other initialization procedures in EB flowsheeting.

Sparse Matrix Methods

The EB approach requires the periodic solution of a large, sparse, linear equation system ultimately involving perhaps tens of thousands of equations. The large storage required to implement this approach was once considered a major problem. However with the continuing improvements in computer hardware, and the development of better sparse matrix techniques this problem has ameliorated considerably, though further improvements along these lines are still desirable.

Most EB programs make use of some general-purpose sparse matrix method, such as Harwell's MA28. These routines take little advantage of the natural block-stream structure of the flowsheeting problem, and in fact tend to destroy what desirable structure might have originally been present in the matrix. Stadtherr and Wood [26,27] have recently described sparse matrix methods that do take advantage of the inherent structure in flowsheeting matrices, and maintain that desirable structure throughout the solution of the sparse matrix. Comparisons indicate that these sparse matrix methods significantly outperform the usual general-purpose approach. It appears however that some significant changes in the sparse matrix techniques currently used will be needed if one is to take best advantage of the latest computing technology, as discussed in the next section.

IMPACT OF NEW COMPUTING TECHNOLOGY

A new challenge and opportunity for those working in the area of process simulation, design, and optimization is the increasing availability of supercomputers. These machines differ architecturally from today's "conventional" large mainframe computers, and have the potential to provide very large increases in computational speed relative to the conventional machine. However the amount by which speed can be increased depends on the problem to be solved, on how that problem is formulated, and on what strategies are used to solve the problem. If its architecture is not well exploited, a doubling of speed is all that might be expected; if fully exploited, speed may be increased by a factor of twenty or more. This level of performance comes at a price upwards of $10 million, however.

While recent years have seen an explosion in microcomputer technology, with constant improvements in speed and memory, this has generally not been seen on the other end of the computer size spectrum. Machines ranging from superminis (e.g., the VAX 11/780), through fast mainframes (e.g., the CDC CYBER series, or array processors combined with a VAX), to supercomputers (e.g., the Cray-1) have been on a relatively constant plateau of speed and memory size for several years, despite advances in chip technology. Indications are that this is soon to change, at least with respect to supercomputing. Supercomputers two orders of magnitude faster than current supercomputers seem likely by the end of the decade. Perhaps of even more impact is that the technology now exists to mass-produce supercomputers comparable to the current Cray-1 for under $1 million each. Nobel laureate Kenneth Wilson of Cornell University [28] expects that competition will cause prices for these mass-produced machines to eventually drop to not much more than the prices of the widespread superminis. An explosion in supercomputing technology seems to be on the horizon.

The ready availability of relatively cheap supercomputing power will have a major impact in the areas of process flowsheeting and process control, provided that practitioners are able to take advantage of the different architecture of these machines. From the standpoint of steady-state process flowsheeting and optimization, the use of supercomputing will provide truly interactive design capabilities, most likely involving powerful local workstations, and sophisticated graphics interfaces for input and output. The engineer will be able to make a design change in a complex process and will almost immediately be able to see the simulated result. In terms of actual wall-clock time, such simulations today too often require several minutes even on a relatively fast conventional machine, and perhaps several hours on an engineer's personal micro. Supercomputing will thus mean tremendous increases in the productivity of the design engineer. The supercomputer will also have a major impact on the ability to do realistic dynamic process simulations and to design control systems. While such off-line simulations can be used to improve control system design and plant operability, realistic on-line simulations using supercomputers may provide a capability for on-line optimization of complete plant operations.

It is important to emphasize that computational strategies that work well on the conventional machines may not be the best on a supercomputer. For instance, when Duerre and Bumb [29] implemented the ASPEN flowsheeting system on a Cray-1, they found that it ran only two to three times faster than on an IBM 370, which indicates that little advantage was taken of the

supercomputer architecture. Of the different approaches to process flowsheeting it is not clear which is likely to be best suited to use on supercomputers. We discuss here one issue involved in implementing the EB approach on supercomputers.

As noted above, the solution of a large sparse linear equation system is a key step in EB flowsheeting, and we consider here how this step should be carried out on supercomputers. Research dealing with the direct solution of sparse linear equation systems on supercomputers has dealt both with general systems of equations having no regular structure, and with highly structured systems such as those with a tridiagonal, banded, or block-diagonal form. Process flowsheeting matrices do not have such a highly structured form, and in general require the use of more general-purpose linear equation solvers, though as Stadtherr and Wood [26,27] have pointed out, there is some problem structure that can be exploited. General-purpose routines however are not easily "vectorizable", i.e., they are not able to take much advantage of the vector processing capability of the supercomputer. This is clearly seen in some comparisons conducted by Duff and Reid [30]. A general-purpose full matrix routine, compiled and executed on a Cray-1, was found to be roughly 20 to 30 times faster than the same routine compiled and executed on an IBM 3033. On the other hand, when the same comparison was made using a general-purpose sparse matrix routine, it was found to be only about two times faster on the Cray-1, which appears to be due simply to the fact that the scalar speed of the Cray-1 is roughly twice that of the IBM 3033. Clearly very little vectorization of the sparse matrix routine was possible, largely, it appears, because of the amount of indirect addressing in sparse matrix codes. Quoting from [30], "This is very disappointing since there is no easy fix which can give us better vectorization. We are therefore forced to rethink our sparse matrix algorithms with the Cray architecture in mind." Similarly if we are to make the most effective use of supercomputers in equation-based flowsheeting, we must rethink the sparse matrix strategies used.

A variety of ideas [e.g., 31-34] have been suggested for more effectively using supercomputers in general-purpose sparse linear equation solving. In the context of equation-based flowsheeting and optimization, there are three such ideas that appear to be of some promise, namely the use of a frontal approach [e.g., 35], the use of block-oriented methods [e.g., 36], and use of methods involving a search for contiguous nonzero elements [e.g., 37]. In each case the basic idea is to treat parts of the sparse matrix as full to increase the amount of vectorization. This will mean the storage of some zeros, and thus some wasted storage. It also means some wasted operations due to multiplications by zero. The goal is to use the structure of the process flowsheeting problem to keep the percentage of wasted storage and wasted operations relatively low. Stadtherr and Vegeais [38] have recently discussed some of the advantages and disadvantages in this regard of the three methods mentioned above.

Another area in which new computing technology will have an important impact is in use of powerful personal computers or local workstations. One can envision such workstations being used both for communicating with supercomputers on which rigorous simulations might be run, or for local use in running simulations using relatively simple models. Such local use should be particularly useful in the preliminary screening of process alternatives, or in taking a close look at individual units or groups of units in a larger process. Ideally for use in screening, the local program should have an optimization capability so that process alternatives can be compared on a roughly optimal basis. Again it is not clear which of the approaches to process flowsheeting is best suited for this type of computing technology. Some SeqM software has already been converted for use on PC's, and this approach might be best for those machines with rather limited central memory (RAM). On the other hand, the SeqM approach is not well suited to optimization, so one might expect a SimM or EB approach, or combination thereof, to also have some advantages, especially for machines equipped with plenty of RAM.

REFERENCES

1. M. Shacham, S. Macchietto, L. F. Stutzman & P. Babcock, Equation oriented approach to process flowsheeting. Comput. Chem. Eng. 6, 79 (1982).

2. J. D. Perkins, Equation-oriented flowsheeting. In Proceedings of the Second International Conference on Foundations of Computer-Aided Process Design (eds. A. W. Westerberg and H. H. Chien), CACHE (1984).

3. J. D. Perkins and R. W. H, Sargent, SPEEDUP: A computer program for steady-state and dynamic simulation of chemical processes. In Selected Topics on Computer-Aided Process Design and Analysis (eds. R. S. H. Mah and G. V. Reklaitis), AIChE Symposium Series (1982).

4. M. G. G. Van Meulebrouk, A. G. Swenker & J. A. de Leeuw den Bouter, Using TISFLO-II for the optimization of utility generation and supply for a chemical complex. IChemE Symp. Ser. 74, 7 (1982).

5. M. H. Locke, A CAD tool which accomodates an evolutionary strategy in engineering design calculations. Ph. D. Thesis, Carnegie-Mellon University (1981).

6. A. J. Field, R. P. Maddams & W. Morton, The incorporation of procedures in an equation-oriented flowsheeting environment. IChemE Symp. Ser. 92, 341 (1985).

7. M. A. Stadtherr & C. M. Hilton, Development of a new equation-based process flowsheeting systems: Numerical studies. In Selected Topics on Computer-Aided Process Design and Analysis (eds. R. S. H. Mah and G. V. Reklaitis), AIChE Symposium Series (1982).

8. P. Cronin, S. Barendregt, R. Srinivasan & M. Van Meulebrouk, Optimization of an ammonia plant using an equation based package. Pres. at AIChE Spring National Meeting, Houston (1985).

9. J. A. de Leeuw den Bouter, M. Van Meulebrouk, A. G. Swenker & S. Barendregt, Olefin plant optimisation using SPYRO and TISFLO-II. Pres. at AIChE Spring National Meeting, Houston (1983).

10. P. K. Gupta, R. C. Lavoie & R. R. Radcliffe, An industry evaluation of SPEEDUP. Pres. at AIChE Annual Meeting, San Francisco (1984).

11. E. Gordon, M. H. Hashemi, R. D. Dodge & J. LaRosa, A versatile steam balance program. CEP, 74(7), 51 (1978).

12. B. Hegner and H. Schoenmakers, CHEMASIM--Experience with BASF's Simultaneous Process Simulator. IChemE Symp. Ser. 92, 365 (1985).

13. S. Macchietto, Solution techniques for processes described by mixed sets of equations and procedures. IChemE Symp. Ser. 92, 377 (1985).

14. J. R. Paloschi, The numerical solution of nonlinear equations representing chemical processes. Ph. D. thesis, University of London (1982).

15. A. Lucia and S. Macchietto, New approach to approximation of quantities involving physical properties derivatives in equation-oriented process design. AIChE J., 29, 705 (1983).

16. A. Lucia and K. R. Westman, Low cost solution to multistage, multicomponent separation problems by a hybrid fixed point algorithm. In Proceedings of the Second International Conference on Foundations of Conputer-Aided Process Design (eds. A. W. Westerberg and H. H. Chien), CACHE (1984).

17. D. C. Miller and A. Lucia, The behavior of a hybrid fixed-point method in a chemcial process design environment. AIChE J., 31, 329 (1985).

18. A. Lucia, D. C. Miller & A. Kumar, Thermodynamically consistent quasi-Newton formulae. Pres. at AIChE Annual Meeting, San Francisco (1984).

19. K. R. Westman, A. Lucia, and D. C. Miller, Flash and distillation calculations by a Newton-like method. Comput. Chem, Eng., 8, 219 (1984).

20. H. S. Chen and M. A. Stadtherr, On solving large sparse nonlinear equation systems. Comput. Chem. Eng., 8, 1 (1984).

21. H. S. Chen & M. A. Stadtherr, A modification of Powell's dogleg method for solving systems of nonlinear equations. Comput. Chem. Eng. 5, 143 (1981).

22. T. L. Wayburn and J. D. Seader, Solutions of systems of interlinked distillation columns by differential homotopy-continuation method. In Proceedings of the Second International Conference on Foundations of Conputer-Aided Process Design (eds. A. W. Westerberg and H. H. Chien), CACHE (1984).

23. J. D. Seader, R. Chavez & T. L. Wayburn, Multiple solutions to systems of interlinked distillation columns by differential homotopy continuation. Pres. at AIChE Annual Meeting, San Francisco (1984).

24. T. L. Wayburn and J. D. Seader, Degree theory and homotopy--Tools for computer-aided process design. Pres. at AIChE Annual Meeting, San Francisco (1984).

25 V. H. Hlavacek, Invited discussion: Nonsequential modular flowsheeting. In Proceedings of the Second International Conference on Foundations of Conputer-Aided Process Design (eds. A. W. Westerberg and H. H. Chien), CACHE (1984).

26. M. A. Stadtherr and E. S. Wood, Sparse matrix methods for equation-based chemical process flowsheeting--I. Reordering phase, Comput. Chem. Eng., 8, 9 (1984).

27. M. A. Stadtherr and E. S. Wood, Sparse matrix methods for equation-based chemical process flowsheeting--II. Numerical phase, Comput. Chem. Eng., 8, 19 (1984).

28. K. Wilson, Supercomputing: Cooperation with industry, SIAM News, 18(5), 6 (1984).

29. K. H. Duerre and A. C. Bumb, Implementing ASPEN on the Cray computer, Los Alamos Report LA-UR-81-3528, Los Alamos National Laboratory, Los Alamos, New Mexico (1981).

30. I. S. Duff and J. K. Reid, Experience of sparse matrix codes on the Cray-1, Comput. Phys. Comm., 26, 293 (1982).

31. W. P. Peterson, Vector Fortran for numerical problems on Cray-1, Comm. ACM, 26, 1008 (1983).

32. D. A. Calahan, Performance of linear algebra codes on the Cray-1, SPE J., 21, 558 (1981).

33. D. A. Calahan, Direct solution of linear equations on the Cray-1, Cray Channels, 3(2), 2 (1981).

34. D. Heller, A survey of parallel algorithms in numerical linear algebra, SIAM Review, 20, 740 (1978).

35. I. S. Duff and J. K. Reid, The multifrontal solution of unsymmetric sets of linear equations, AERE Harwell Report CSS 133 (1983).

36. D. A. Calahan, A block-oriented equation solver for the Cray-1, SEL Report No. 136, University of Michigan, Ann Arbor (1980).

37. D. A. Calahan and W. G. Ames, Vector processors: Models and applications, IEEE Trans. on Circuits and Systems, CAS-26, 715 (1979).

38. M. A. Stadtherr and J. A. Vegeais, Process flowsheeting on supercomputers. IChemE Symp. Ser. 92, 67 (1985).