Improving chemical process design with supercomputers

Alan B. Coon, Union Carbide Technical Center, South Charleston, West Virginia
Christopher H. Goheen and Mark A. Stadtherr, University of Illinois, Urbana, Illinois

Although the equation-based (EB) methodology is the most attractive approach to chemical process design and simulation on supercomputers, it often results in a computational bottleneck. Successful implementation of the EB approach requires a linear solver that makes efficient use of parallel and vector architectures. This efficiency is critically dependent on the reordering phase of the equation-solving step of the simulation when direct sparse solvers are used. Therefore, the reordering algorithm is a crucial factor in exploiting the supercomputer's resources. A new reordering algorithm was teamed with a CRAY-2 computer system at the National Center for Supercomputing Applications to produce significant improvements in the linear equation solving step for an EB flowsheeting code.

Chemical process design methods

Traditionally, the most popular approach to chemical process flowsheeting has been the sequential-modular problem formulation. In this approach, the problem flowsheet equations are grouped according to the process units with which they are associated; they are then solved together (possibly along with a set of design specification equations) as several independent groups of equations. The order of solution of these groups is determined primarily by the flow of material through the actual chemical process. However, the equations representing the process topology with respect to these units contain variables that are needed as input to certain modular solution routines, and the values of some of these variables, often referred to as "tear stream variables," need to be estimated initially to begin the computations. Once the sequence of unit modular computations has been completed, the new values of the tear stream variables are compared to the previous values, and the sequence is repeated if some previously determined convergence criteria are not met. Most of the current commercial process design packages are sequential modular based codes.

Computational experience with this approach on vector processors, however, has not been encouraging. Typically, performance improvements in sequential modular approaches on supercomputers are attributable to scalar speed differences alone, with insignificant performance gains by vectorization. While parts of these codes have been rewritten for vectorization, this effort does not produce a significant overall impact for a wide variety of chemical plant simulations and designs because of the inherently sequential nature of the sequential-modular approach. (The minority of sequential-modular problems that vectorize well usually include one process module that is modeled by solving a very large set of simultaneous equations; solution

of this module dominates the overall computation time, and in this case the sequential modular formulation is actually more equation oriented than module oriented.) The potential for using this approach on parallel processing machines is also significantly limited. Successful parallelization of code, like successful vectorization, requires that there be some independence in the order of computation.

In the equation-based approach to chemical process flowsheeting, the problem equations describing both process units and topology are solved simultaneously, rather than dividing the problem into smaller unit modular subproblems to be solved sequentially. This approach allows design specifications and constraints to be incorporated easily in the problem. It also significantly increases computational efficiency and enhances flexibility in the types of problems that can be solved. In the general case, equation-based process flowsheeting requires the solution of large, sparse differential-algebraic equation (DAE) systems. These systems result from the algebraic equations, the partial differential equations (PDE) (the presence of which usually requires a discretization of the spatial domain), and ordinary differential equations (ODE) that describe the various unit processes, the process topology, and design specifications. The general DAE problem can be written as

$$F(y', y, t) = 0 \text{ for } t_0 \le t \le t_f \quad (1)$$
$$y(t_0) = y_0 \quad (2)$$
with $y \in \Re^n$, $t \in \Re$, and $F : D \subset \Re^n \times \Re^n \times \Re \rightarrow \Re^n$

Here F, the nonlinear flowsheet equations, and y, the time dependent variable set including both differential and algebraic variables, are assumed to be sufficiently differentiable. Equations 1 and 2 represent an implicit ODE system if the Jacobian ($\partial F/y'$) is nonsingular for $t_0 \le t \le t_f$. Otherwise, ($\partial F/y'$) is singular for some $t_0 \le t \le t_f$, in which case the reduction of the DAE to an ODE requires some subset of Equation 1 be differentiated. In either of these two cases, and provided that a set of consistent initial conditions can be established, the system in Equations 1 and 2 is reduced to the form

$$f(y) = 0 \quad (3)$$
with $y \in \Re^n$, and $f : S \subset \Re^n \rightarrow \Re^n$

and the nonlinear system of equations represented by Equation 3 remains to be solved at various time steps. This set of equations is linearized, and the resulting sparse Jacobian matrix (more often, an approximation of this matrix) is solved as part of the nonlinear solution method. The resulting linear equation set can be written

$$Ax = b \quad (4)$$
where $A = (\partial f/\partial y)|_{y = y_k}$, or some appropriate approximation
$x = y_{k+1} - y_k$
$b = -f(y_k)$

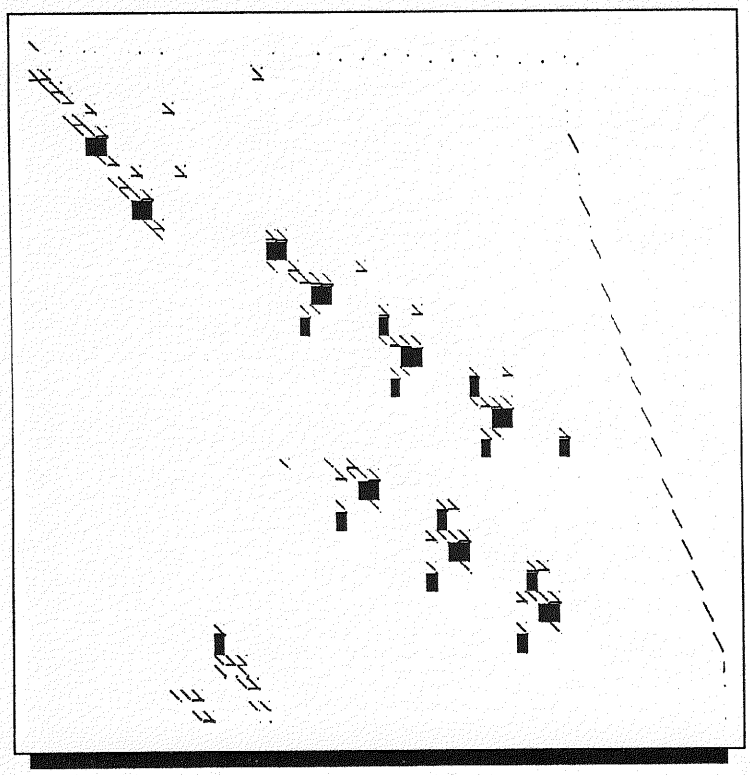
The solution of this large, sparse system of equations is the single most computationally intensive step in the overall flowsheet solution process. Equation 4 usually is solved several times for each nonlinear equation solution, with the coefficients in matrix A updated at each iteration, and the process continuing until some convergence criterion is satisfied. The nonlinear equation-solving step, in turn, must be executed for each time step in the simulation. In some problems, the linear solution routine may require nearly 90 percent of the overall flowsheeting computation time.

Direct sparse solvers

Most general-purpose direct sparse linear solution routines use some type of indirect addressing, which can degrade vector performance drastically. In these instances, the use of hardware gather/scatter in vector machines can be quite effective. Another approach is to use variable bandwidth solvers, thereby eliminating the need for indirect addressing at the expense of executing vector operations that may include a great deal of zero-operand arithmetic. The unnecessary operations occur because these methods treat all coefficients between the bandwidth boundaries as nonzeros. While this approach enhances vectorization, popular measures of such enhancement, such as MFLOPS rating, can be misleading because the overall solution will be quite inefficient if the bandwidth is not kept reasonably small.

The frontal method has been a popular means of exploiting sparse matrix structure on vector processors, because the difficulties inherent in indirect addressing on vector processors can be circumvented in much the same way as with variable bandwidth solvers. However, the frontal method and its variants tend to work well only for unsymmetric matrices that can be reordered to a nearly structurally symmetric form. One approach to improving performance of the frontal method on EB matrices that cannot be reordered easily to a nearly structurally symmetric form is to reorder them to a block structure that is nearly structurally symmetric. The reordering algorithm considered here was originally developed for a coarse-grained parallel sparse solver that can exploit the natural block structure of EB flowsheeting matrices.[1] This generalized block tridiagonal reordering produces a symmetric block structure. This structure is achieved through a recursive two-way partitioning of the vertices in the underlying bipartite graph of the sparse matrix. For a good frontal method ordering, the frontal matrix size must be kept small. This can be achieved by minimizing the number of graph edges that are cut by the partition.

Graph partitioning algorithms have several applications, most notably VLSI placement and routing and computer program paging, and they are used in various divide-and-conquer strategies. As a result, many algorithms for finding graph partitions that minimize the size of the cutset (number of edges cut) have been proposed over the past two decades. However, the problem of partitioning the vertex set into two sets of the same order with the minimum number of nets cut is NP-complete, and consequently these algorithms attempt to select a good approximation of such a partition using various heuristics. EB flowsheeting matrices present a further complication, however, for these matrices are usually highly nonsymmetric and their sparsity is not well approximated by that of their symmetric parts. Therefore, a bipartite graph model must be used to include the asymmetry of the matrix. In spite of this additional complexity, a linear time-partitioning algorithm for unsymmetric EB flowsheeting matrices has been developed which drastically improves



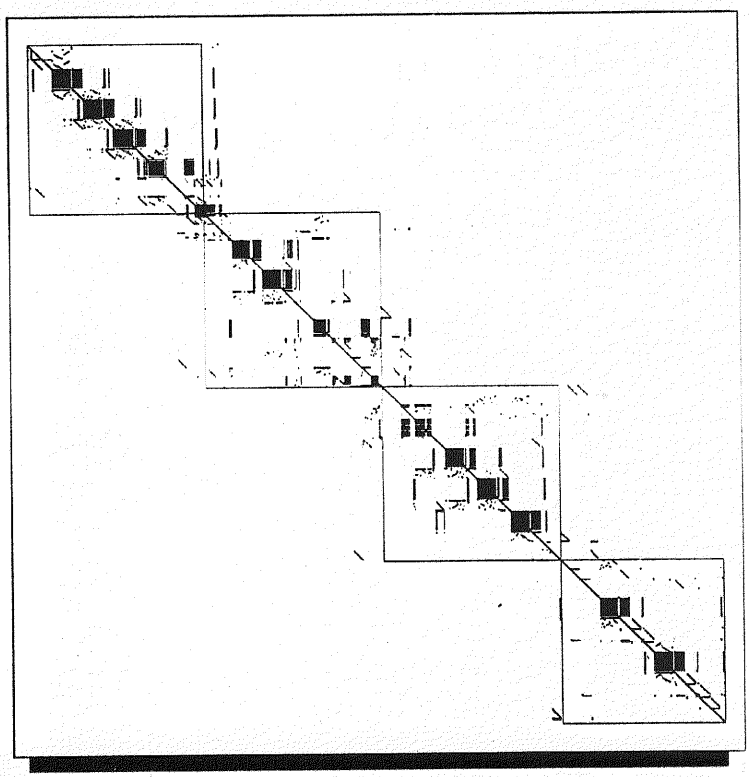


Figure 1. (above). Occurence matrix for Problem 1 before new reordering.

Figure 2. (above right). Occurence matrix for Problem 1 after new reordering.

a CRAY-2 system implementation of an EB flowsheeting code.[2]

Test problems

Two flowsheeting problems were used to test the effectiveness of two new variants of the reordering algorithm, along with the original reordering.[3] The two new orderings differ in the amount of imbalance allowed in partitioning the underlying graph of the matrix. SEQUEL-II, a prototype EB flowsheeting code developed at the University of Illinois and implemented on the CRAY-2 system at the National Center for Supercomputing Applications, was used to generate sets of equations for these problems.[4,5] The first flowsheeting test problem was a light hydrocarbon recovery process with 20 chemical components. The SEQUEL-II formulation required 28 units, 48 streams, 1477 equations, and 18,592 nonzero coefficients. The second flowsheeting problem was a simulation of a natural gas processing plant with 20 components. The SEQUEL-II formulation for this process consisted of 21 units, 39 streams, 1235 equations, and 16,868 nonzero coefficients. Both problems were quite sparse, with a nonzero density of approximately 1 percent. For the numerical experiments on these test problems, two versions of the frontal method that differ in the rank of the update to the frontal matrix were used.[3] Both the rank-one and the rank-four update routines were coded in Cray Assembly Language for increased efficiency.

Results

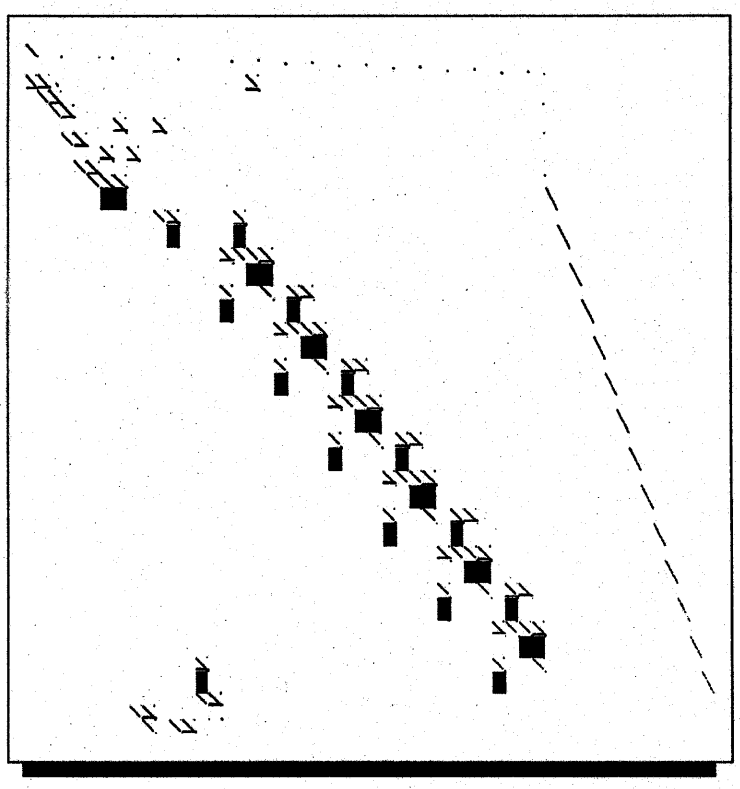
Comparisons on the CRAY-2 system using SEQUEL-II show substantial improvement with the new partitioning scheme over the original reordering

Table 1. Maximum frontal matrix sizes and total linear solution times with rank-one update.

			Original reordering (reverse P4)		New (balanced) reordering		New (unbalanced) reordering	
Pblm	Order	Nonzeros	Maximum frontal size	Total linear time (sec)	Maximum frontal size	Total linear time (sec)	Maximum frontal size	Total linear time (sec)
1	1477	18,592	60 x 942	6.49	111 x 235	4.38	111 x 236	4.48
2	1235	16,868	239 x 887	25.48	130 x 217	5.66	109 x 177	5.44

Table 2. Maximum frontal matrix sizes and total linear solution times with rank-four update.

			Original reordering (reverse P4)		New (balanced) reordering		New (unbalanced) reordering	
Pblm	Order	Nonzeros	Maximum frontal size	Total linear time (sec)	Maximum frontal size	Total linear time (sec)	Maximum frontal size	Total linear time (sec)
1	1477	18,592	60 x 944	4.68	114 x 235	3.31	111 x 236	3.61
2	1235	16,868	241 x 888	15.37	132 x 219	4.68	109 x 177	3.89



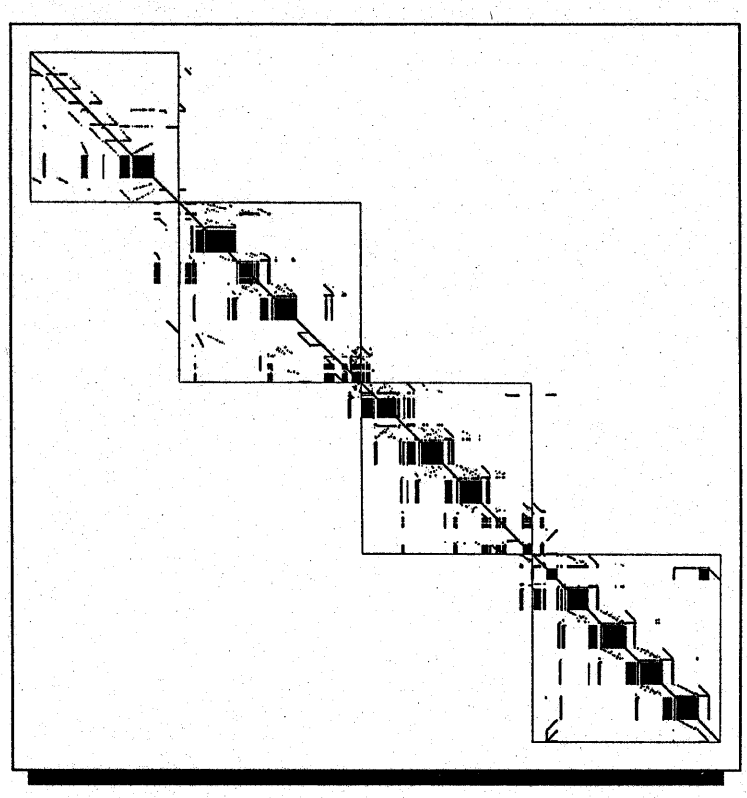


Figure 3. (above left). Occurence matrix for Problem 2 before new reordering.

Figure 4. (above). Occurence matrix for Problem 2 after new reordering.

used with the frontal method in SEQUEL-II. The frontal matrix sizes given by the new reordering were significantly smaller than those given by the original. Table 1 indicates that balanced partitioning is slightly better than the unbalanced partitioning for Problem 1, and the reverse for Problem 2. Table 2 shows the improvement in performance gained by switching to a rank-four update resulting from a more efficient use of the single path to memory on the CRAY-2 system. Figures 1-4 are the occurence matrices for both problems, before and after reordering.

Conclusions

Initial experience with the new generalized block tridiagonal orderings indicates that substantial improvements in the linear solution stages of equation-based process simulation are possible even for methods that already vectorize well on supercomputers. In the case of the frontal method, the frontal matrix sizes and amount of floating-point arithmetic involving zero-valued operands were strongly dependent on the method used to reorder the matrix. Based on this finding, the equation-based simulator was improved by developing new algorithms for the reordering, a stage that represents an insignificant fraction of the overall solution time.

Equation-based chemical process flowsheeting codes are the appropriate choice for vector and parallel machines, because the potential for concurrency in the traditional sequential modular method is inherently limited. However, efficient use of such advanced computer hardware also depends on a suitable choice of algorithms for the equation-based methodology. This study demonstrates that the appropriate combination of methodology and algorithms can fully exploit the power of a supercomputer such as the CRAY-2 system.

About the authors

Alan B. Coon is a senior chemical engineer at Union Carbide Corporation in the Applied Mathematics and Processs Simulation Group. He received his B.S.Ch.E. degree from the University of Texas at Austin, and his M.S. and Ph.D. degrees in chemical engineering from the University of Illinois at Urbana-Champaign.

Christopher H. Goheen received his B.Ch.E. degree at Georgia Tech and his M.S.Ch.E. degree at the University of Illinois. He is completing Ph.D. studies in chemical engineering at the University of Illinois. He will join the Technical Computing Section of Mobil Research and Development Corporation in Princeton, New Jersey, in the fall of 1991.

Mark A. Stadtherr is a professor at the University of Illinois at Urbana-Champaign in the Department of Chemical Engineering. He also works with the National Center for Supercomputing Applications. He received his B.Ch.E. degree from the University of Minnesota and his Ph.D. degree from the University of Wisconsin.

References

1. Coon, A. B. and M. A. Stadtherr, "Parallel Sparse Matrix Solvers for Equation-based Process Flowsheeting," *Institution of Chemistry Engineering Symposium Series*, Vol. 114, pp. 165-176, 1989.
2. Coon, A. B., "Orderings and Direct Methods for Coarse Granular Parallel Solutions in Equation-based Flowsheeting," Ph.D. Thesis, University of Illinois, Urbana, Illinois, 1989.
3. Zitney, S. E., "Frontal Algorithms for Equation-based Chemical Process Flowsheeting on Vector and Parallel Computers," Ph.D. Thesis, University of Illinois, Urbana, Illinois, 1989.
4. Stadtherr, M. A. and C. M. Hilton, "Development of a New Equation-based Process Flowsheeting System: Numerical Studies," *Selected Topics on Computer-Aided Process Design and Analysis*, R. S. H. Mah and G. V. Reklaitis, Eds., AIChE Symposium Series, Vol. 78, pp. 12-28, 1982.
5. Zitney, S. E. and M. A. Stadtherr, "Computational Experiments in Equation-based Chemical Process Flowsheeting," *Computers and Chemical Engineering*, Vol. 12, pp. 1171-1186, 1988.