Third International Symposium on
Process Systems Engineering
Sydney, 28 Aug - 2 Sept 1988

A Frontal Algorithm for Equation-Based Chemical Process Flowsheeting on Vector Computers

S.E. ZITNEY
Graduate Student, Chemical Engineering, University of Illinois
M.A. STADTHERR
Associate Professor, Chemical Engineering, University of Illinois

1 INTRODUCTION

The periodic solution of a large sparse system of linear equations is a key step in the equation-based (EB) approach to chemical process flowsheeting. The sparse linear matrices that arise in EB problems do not have highly regular structures such as tridiagonal or banded forms, nor do they have desirable numerical properties such as diagonal dominance or positive definiteness. For this reason, most EB programs make use of general-purpose sparse matrix methods.

Unfortunately, general-purpose sparse matrix solvers appear to be incompatible with most vector computers because of indirect addressing in inner loops. On vector computers, it is desirable to address data in contiguously or regularly indexed vectors. This may be done in a sparse matrix algorithm by treating parts of the sparse matrix as full.

The frontal approach alternates between assembly and elimination so that operations are confined to a relatively small submatrix (frontal matrix) that can be regarded as full. The front advances down the diagonal of the sparse matrix as the solution proceeds. Operating on full matrices in this way eliminates indirect addressing, and so facilitates vectorization on advanced computer architectures.

Based on Gaussian elimination, the frontal approach originated as a banded matrix solver for finite element applications (Irons, 1970; Hood, 1976). Each element is assembled into the frontal matrix in an order dictated by element numbering. Equations and variables enter the front as their elements are processed. A variable can be selected as a pivot only after its row and column is fully summed; that is, whenever all the elements containing the variable are in the frontal matrix. After an elimination step, the pivot row is temporarily stored out of core for the purpose of back-substitution. This interleaved assembly-elimination process continues until all the elements are assembled and the elimination is completed. Finally, the previously eliminated rows are retrieved from auxiliary storage in reverse order so that the solution can be found by back-substitution.

Most of the recent research on the frontal approach has concentrated on solving large sparse matrices on sequential computers with limited central memory. Thompson and Shimazaki (1980) developed a frontal procedure which uses a compact skyline storage scheme to reduce data transfer between disk and core. Beer and Haas (1982) developed a frontal solver which allows for partitioning of the frontal matrix and improves disk I/O. Idá and Lord (1984) devised a frontal routine to accommodate medium to large-sized problems on minicomputers. Gloos et al. (1986) used controlled block paging and special disk file management for computers with physical memories not large enough to store the frontal matrix.

The potential for using the frontal approach on vector computers was apparently first recognized by Duff (1979). Duff (1984) and Duff and Reid (1982) have since implemented a frontal code (MA32) on the Cray-1 at Harwell. The MA32 routine is designed for the out-of-core solution of sparse unsymmetric matrices, and allows for input by equation as an option. However, the code is generally structured for input by finite element.

More recently, Dave and Duff (1987) modified the MA32 routine to take better advantage of the Cray-2 architecture. Since chaining cannot be used on the Cray-2, they implemented a two-pivot version of the frontal approach which overlaps both floating-point pipes with the load/store pipe. The two-pivot kernel is coded in assembler, and combines the use of local memory and vector registers.

As mentioned above, the frontal approach was developed to accommodate large matrices on computers with small central memories. This approach makes use of auxiliary storage (slow-access) and is I/O intensive. An alternative approach is to make use of the large central memories (rapid-access) available on today's supercomputers. A frontal solver that functions entirely in central memory requires considerably less solution time than a frontal code that uses auxiliary storage. However, when working with very large problems, it may still be necessary to use auxiliary storage even on systems such as the Cray-2 with its very large central memory of 256 Megawords (Dave and Duff, 1987).

The idea of applying the frontal approach to process flowsheeting matrices was first proposed in the papers by Stadtherr and Vegeais (1985) and Vegeais and Stadtherr (1988). However, they did not implement a complete frontal code or provide a complete algorithm for their approach. The frontal approach as implemented here in SEQUEL-II (Zitney and Stadtherr, 1988) is used to solve several EB flowsheeting problems on the Cray X-MP/48 supercomputer. We also make comparisons to a general-purpose sparse matrix solver, namely LUISOL, a very reliable and efficient routine described elsewhere (Chen and Stadtherr, 1984; Stadtherr and Wood, 1984b).

2 DESCRIPTION OF FRONTAL ALGORITHM FOR EB FLOWSHEETING PROBLEMS

The frontal approach for finite element problems was detailed in the original papers of Irons (1970) and Hood (1976). We present here an adaptation of this approach to the more general process flowsheeting matrix. We first briefly outline our algorithm, and then demonstrate it by means of a simple example. Finally, we discuss the most important aspects of its implementation.

Prefront process

(1) Reorder the process flowsheeting matrix using the P4 algorithm (Hellerman and Rarick, 1972).
(2) Reverse the row (equation) order.
(3) Perform a pass through the matrix to determine the last occurrence of each variable. This information is stored, so that the fully-summed variables can be detected in Step (6).
(4) Initialize the frontal matrix.

Frontal process

(5) Assemble a flowsheet equation into the frontal matrix. If all equations have been assembled, go to Step (12); otherwise continue to Step (6).
(6) Determine whether any variables are fully summed by using the information stored in Step (3). If there are fully-summed variables, go to Step (7); otherwise return to Step (5).
(7) Search for pivot row by partial pivoting in the fully-summed column.
(8) Normalize pivot row.
(9) Eliminate and delete pivot row and column.
(10) Store pivot row in central memory.
(11) Return to Step (6).
(12) Compute the solution vector by back-substitution.

It should be emphasized that the frontal matrix is assembled by equation, rather than by finite element. The assembled equations (rows) are always fully summed since variables enter the frontal matrix the first time they are encountered in an equation. An elimination step is performed whenever any variable (column) becomes fully summed; that is, whenever the equation containing the last occurrence of the variable is reached.

As an example, consider the following 6x6 matrix:

	1	2	3	4	5	6
1	X	X				X
2		X	X	X		
3			X	X		
4				X	X	X
5	X			X		X
6		X	X	X	X	X

In the prefront process, the matrix is ordered by taking an a priori reordering method intended to reduce fill-in in general sparse matrix solvers and reversing the order of the rows (Stadtherr and Vegeais, 1985; Vegeais and Stadtherr, 1988). The a priori reordering method used here is the partitioned preassigned pivot procedure, P4, devised by Hellerman and Rarick (1972). A detailed description of the method and a discussion of its computational efficiency are given by Stadtherr and Wood (1984a). The P4 method generates a lower triangular matrix with spike columns (columns 4 and 6, below). For our example problem, the P4 ordering that results is:

	3	2	5	6	1	4
3	X					X
2	X	X				X
4			X	X		X
6	X	X	X	X		X
1		X		X	X	
5				X	X	X

Reversing the order of the rows produces an upper triangular matrix with spike columns. For convenience, we renumber the rows and columns in ascending order:

	1	2	3	4	5	6
1	X	X	X			
2		X	X		X	
3	X		X	X	X	X
4	X		X	X		
5	X				X	X
6	X					X

The ordering of the flowsheet equations is critical since it determines the maximum size of the frontal matrix during elimination. The reordering of flowsheeting matrices into a desirable form for the frontal approach will be discussed in more detail below.

The prefront routine then makes a pass through the matrix to determine when each variable appears for the last time. This information is stored, so that during the subsequent assembly-elimination process the completion of a column can be detected. In our example, variables 1-6 make their last appearance in equations 6,2,4,4,5, and 6, respectively.

The assembly-elimination process starts by assembling the first equation into the frontal matrix. In order to do this, variables 1,2, and 3 must be entered into the front. The initial frontal matrix looks like:

	1	2	3
1	X	X	X

Note here that the assembled equation (row) is fully summed, but that none of the variables (columns) are fully summed because each is contained in at least one future equation. Thus we proceed to the second equation with variables 2, 3, and 5. This equation introduces a new active variable (variable 5), and it must enter the frontal matrix. The set of active variables becomes (1,2,3,5), and the frontal matrix now looks like:

	1	2	3	5
1	X	X	X	
2		X	X	X

The second variable is now fully summed. The frontal solver determines this by using the array of last occurrences stored during the prefront process. A nonzero element in the second column is now chosen as a pivot. After eliminating the other nonzero elements in the column, the pivot row and column are removed. The deleted pivot row is stored for use in back-substitution. If the second equation is chosen as the pivot row the following frontal matrix is obtained:

	1	3	5
1	X	X	F

The F here represents a zero element that has become a nonzero element during elimination (a fill-in). Note here that the first equation could also have been chosen as the pivot row. Numerical pivoting considerations will be discussed in more detail below. Now the third row is entered into the frontal matrix. Since the third equation contains the new active variables 4 and 6, they must enter the frontal matrix. All six of the variables have now entered the frontal matrix, which looks like:

	1	3	5	4	6
1	X	X	F		
3	X	X	X	X	X

None of the variables are fully summed. Thus we proceed to the fourth equation after whose assembly the frontal matrix reaches its maximum size (3x5):

	1	3	5	4	6
1	X	X	F		
3	X	X	X	X	X
4	X	X		X	

It should be noted here that there is a maximum front size for any ordering of the equations, and the front dimensions must be large enough to accommodate this maximum. Variables 3 and 4 are now both fully summed. Using equation 1 to eliminate variable 3 leaves the frontal matrix:

	1	5	4	6
3	X	X	X	X
4	X	F	X	

If equation 3 is then used to eliminate variable 4, we have:

	1	5	6
4	X	F	F

The fifth equation is now entered, and the frontal matrix is:

	1	5	6
4	X	F	F
5	X	X	X

Variable 5 is now fully summed and may be eliminated. Pivoting on the fifth equation and entering the final equation into the frontal matrix results in:

	1	6
4	X	F
6	X	X

which is now solved by simple Gaussian elimination. After the assembly-elimination process is completed, the pivot rows are retrieved from central memory in reverse order, and the solution vector is computed by back-substitution.

3 IMPLEMENTATION DETAILS

3.1 Numerical Pivoting

We discuss here the problem of pivoting for numerical stability during the elmination phase of the frontal approach. A typical frontal code for finite element problems makes use of a scheme found in most general-purpose sparse matrix solvers, namely threshold pivoting. In this case each pivot a_{1k} has to meet the following threshold tolerance:

$$|a_{1k}| > t \max_i |a_{ik}|$$

where t is a preset fraction in the range $0<t<1$. The pivots must be chosen from amongst the fully-summed rows and columns, but the maximum is taken over all rows (fully and partly summed) in the frontal matrix. If the pivotal search does not find an entry that satisfies the threshold criterion, the fully-assembled variables remain in the front. This has the effect of increasing the size of the frontal matrix and the number of arithmetic operations.

In our frontal algorithm for flowsheeting matrices, the assembly proceeds equation by equation so that the rows of the frontal matrix are always fully summed. In this case, any nonzero element in a fully-summed column may be chosen as a pivot and it is not necessary to limit pivoting to threshold pivoting. Instead, partial pivoting (t=1) may be performed in the column with no penalty of additional storage or elimination operations.

3.2 Storage of Normalized Pivot Rows

After an elimination step in the frontal approach, the normalized pivot row must be temporarily stored for the purpose of back-substitution. The method of storage depends on the central memory size and disk capabilities of the computer, as well as on the size of the problem. In a typical frontal code developed for computers with limited central memory, the pivot row is transferred to a slow auxiliary storage media such as magnetic tape or disk, and its space in central memory is reused.

For the EB flowsheeting problems in this study, the use of auxiliary storage is unnecessary due to the large central memory of the Cray X-MP/48. The normalized pivot rows are stored in central memory, where they can be accessed rapidly. It should be emphasized that the test problems used here are not particularly large by flowsheeting standards, but they are comparable in size to test problems used by others. However, it may be necessary for larger and/or more complex problems to transfer the pivot rows in and out of central memory. In this case, the main memory of the Cray X-MP can be augmented with 128 million words of fast auxiliary SSD memory. The maximum potential data transfer rate for this solid-state storage device is 200 times faster than conventional disk storage (DD-49 disk drives).

3.3 Vector Processing Considerations

The frontal algorithm is attractive from the standpoint of vectorization because all elimination operations occur in the relatively dense frontal matrix. Thus, we can use full-matrix code without indirect addressing. The elimination step in our code looks like:

```
      DO 20 I=1,FE
         DO 10 J=1,FV
         FRONT(J,I)=FRONT(J,I)+PCOL(I)*PROW(J)
   10    CONTINUE
   20 CONTINUE
```

where the frontal matrix (FRONT) is updated by the outer product of the pivot column (PCOL) and the pivot row (PROW), and FE and FV are the number of equations and variables in the front, respectively. The innermost loop is a dense SAXPY and so can exploit chaining of the add and multiply functional units on the Cray X-MP. Since only the innermost loop is a candidate for vectorization, it is more efficient to make the inner loop have the largest range. With our frontal approach, the number of variables is always greater than or equal to the number of equations. Thus, we make the index of the inner loop correspond to the number of variables in the frontal matrix. Also, we process the two-dimensional frontal array in a column-wise manner in order to avoid memory bank conflicts, which can have a pronounced effect on computation time.

Although the amount of vectorization is increased by treating parts of the sparse matrix as full, some operations are done on zero entries. This disadvantage of the frontal approach is discussed in more detail below.

3.4 Reordering

The performance of our frontal algorithm depends considerably on the row ordering, which determines the size of the frontal matrix during elimination. For the banded matrices arising from finite element problems, the amount of storage needed for the front is only B*(2B - 1), where B is the bandwidth of the matrix. Flowsheeting matrices are not strictly banded, however.

We take advantage here of the fact that a desirable row-column ordering for the frontal approach is the opposite of the order for a general-purpose sparse solver, such as LU1SOL (Stadtherr and Vegeais, 1985; Vegeais and Stadtherr, 1988). With LU1SOL, it is desirable to process a matrix with small rows first and small columns last; that is, a lower triangular matrix. For the frontal approach it is desirable to treat a matrix with small columns first and small rows last, that is, an upper triangular matrix. Thus, we order flowsheeting matrices by taking an a priori reordering method (P4) intended to reduce fill-in in general sparse matrix solvers and reversing the order of the rows. The reverse-P4 row ordering results in an upper triangular matrix with spike columns. If the triangular portion was full, the amount of array storage needed for the front would be (MLSPK + 1)*N, where MLSPK is the maximum number of local spikes (this term is defined in Stadtherr and Wood (1984a) and is related to the proximity to the diagonal of the nonzeros in a spike). However, since the triangular portion of a flowsheeting matrix is relatively sparse, the storage needed for the frontal matrix will typically be less than (MLSPK + 1)*N. This can be seen in our example problem (MLSPK = 2 and N = 6), where the frontal matrix requires a 3 x 5 array, instead of the 3 x 6 array that would have been needed for the full upper triangular case.

4 SEQUEL-II OPTIONS

SEQUEL-II is a prototype flowsheeting system that provides a means for testing different computational strategies for the EB approach. Recent developments regarding the various numerical solution techniques available in SEQUEL-II can be found in Zitney and Stadtherr (1988). Throughout our study of the frontal approach the following options are used in SEQUEL-II:

1. All thermodynamic properties are calculated using the Peng-Robinson (1976) equation of state with all binary parameters set equal to zero.

2. The correction steps are generated by Powell's dogleg method, as modified by Chen and Stadtherr (1981).

3. The initial Jacobian evaluation and subsequent re-evaluations are performed using the sparse finite difference scheme of Curtis, Powell, and Reid (1974).

4. The sparse Jacobian matrices are updated using the least-relative-change update recently devised by Bogle and Perkins (1987).

5. The initial guesses for the process variables are generated using a simplified sequential-modular scheme (Zitney and Stadtherr, 1988).

6. The relative convergence tolerance defined by Chen and Stadtherr (1981) is 10^{-7}. The discretization parameter for Jacobian evaluation is $10^{-5}x_j$, where x_j is the variable being perturbed.

7. These studies were performed using the CDC Cyber 175 and one CPU of the Cray X-MP/48 at the University of Illinois. All runs using the Cyber 175 were done in single precision (60 bits) and under the Fortran Extended (OPT=1) compiler. All runs using the Cray X-MP were done in single precision (64 bits) and under the CFT 1.14 compiler.

5 FLOWSHEETING PROBLEMS

Four benchmark problems are used in this study. Problem 1 is a simple equilibrium flash separation process adapted from exercise 4 in the FLOWTRAN exercise book (Clark, 1977). Problem 2 is the well-known four-flash-unit system used by Cavett (1963). Problem 3 is the ammonia synthesis process studied by Stadtherr and Hilton (1982). Problem 4 is a light hydrocarbons recovery process adapted from exercise 25 in the FLOWTRAN exercise book (Clark, 1977).

Statistics of problem sizes are given in Table I. The table lists for each problem the number of variables, the number of nonzero elements in the occurrence matrix, the percentage density of the matrix, and the maximum number of equations and variables in the frontal matrix. The maximum front size is for reverse-P4 reordering.

TABLE I

FLOWSHEETING PROBLEM STATISTICS

Problem Number	Number of Variables	Number of Nonzeros	% Density	Maximum Front Size
1	80	338	5.3	7 x 42
2	288	3332	4.0	63 x 221
3	155	630	2.6	14 x 78
4	350	1634	1.3	21 x 172

6 NUMERICAL RESULTS

We compare here the performance of our frontal solver with that of a general sparse matrix solver (LU1SOL, Chen and Stadtherr (1984); Stadtherr and Wood (1984)) on the four EB flowsheeting problems listed above. Table II reports the numerical results: the time required for the solution of a single linear system, the total solution time for the flowsheeting problem, and the speedup of the frontal solver on the Cray over LU1SOL-Cray and over LU1SOL-Cyber. The total solution time consists of the linear time and the time spent in all other routines. Thus, a reduction in linear solution time usually results in the total speedup being less than the linear solution speedup.

The flowsheeting matrices were reordered with reverse-P4 for the frontal solver and with P4 for LU1SOL. All run times in Table II do not include the time spent in the P4 routine, but the time required to reverse the row order is included in the frontal solver run times. The Cray runs with LU1SOL are performed using vectorization and hardware gather/scatter.

TABLE II

PERFORMANCE COMPARISON OF FRONTAL SOLVER AND LU1SOL

	Frontal-Cray		LU1SOL-Cray		LU1SOL-Cyber	
Problem Number	Linear Time (sec)	Total Time (sec)	Linear Time (sec)	Total Time (sec)	Linear Time (sec)	Total Time (sec)
1	.002	.099	.006	.122	.030	.656
2	.051	2.486	.120	4.274	.542	24.437
3	.006	.658	.009	.762	.048	4.316
4	.021	.502	.022	.519	.114	3.042
	Speedup of Frontal-Cray over LU1SOL					
	Frontal-Cray		LU1SOL-Cray		LU1SOL-Cyber	
1	1	1	3.00	1.23	15.00	6.63
2	1	1	2.35	1.72	10.63	9.83
3	1	1	1.50	1.16	8.00	6.56
4	1	1	1.05	1.03	5.43	6.06

For problems 1 and 2, the frontal solver on the Cray is several times faster than LU1SOL on the Cray and is between one and two orders of magnitude faster than LU1SOL on the Cyber. The frontal solver did not perform very well on problem 4, however. It is important to point out here that there is an inverse relationship between the speedup of the frontal solver over LU1SOL and the percentage density of the occurrence matrix. This happens because many more operations are wasted on zero entries when the occurrence matrix is more sparse. Any zeros in the frontal matrix are stored explicitly, and such operations cannot be made conditional without adversely affecting the amount of vectorization. Thus, by treating parts of the sparse matrix as full submatrices, the vector operations are performed at a much faster rate, but some of these are wasted on unnecessary operations. We could try to exploit the sparsity of the frontal matrix, but we would have to be careful not to destroy the desirable characteristics (simplicity and vectorizability) of the innermost loops of the frontal solver. The success of the frontal approach depends largely on the front remaining small since this will tend to lower the number of overall arithmetic operations as well as reduce the number of wasted operations on zeros. Thus, it is very important to find a reordering scheme that keeps the size of the frontal matrix small. The reverse-P4 scheme used here gives a desirable ordering, but it is not an attempt to provide an optimum ordering. It should be emphasized that any number of matrix orderings can be used, and in practice an optimum ordering cannot be determined in a feasible amount of time. Recently, Vegeais and Stadtherr (1988) found that the BLOKS reordering scheme (Stadtherr and Wood, 1984a) performs very well in connection with the frontal approach. This reordering scheme attempts to identify and maintain the inherent block structure associated with process flowsheeting matrices.

7 CONCLUSIONS

A frontal algorithm for the solution of large sparse systems of linear equations arising in EB flowsheeting is presented. The frontal solver is designed specifically for input by equation and functions entirely in core.

Results on four flowsheeting problems indicate that the frontal solver on the Cray X-MP/48 outperforms the general sparse solver, LU1SOL, on the Cray and on the CDC Cyber 175. However, there appears to be an inverse relationship between the speedup of the frontal solver over LU1SOL and the percentage density of the occurrence matrix. A more detailed appraisal of the effects of sparsity and matrix structure on the performance of the frontal approach is still needed.

The reverse-P4 scheme used here gives a desirable ordering, but the performance of the frontal approach could be further enhanced by the development of a reordering scheme that is designed to produce an ordering tailored specifically to the frontal approach. Thus the frontal approach shows considerable promise for the solution of EB process flowsheeting matrices on vector computers.

8 ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation under Grant number DMC-8520663.

9 REFERENCES

BEER, G. and HAAS, W. (1982). A partitioned frontal solver for finite element analysis. Int. J. Num. Meth. Engng., Vol. 18, pp 1623-1654.

BOGLE, I.D.L. and PERKINS, J.D. (1987). Improvements to sparse quasi-Newton methods for algebraic systems. Proceedings of Chemical Engineering Fundamentals 87, XVIII Congress: The Use of Computers in Chemical Engineering, European Federation of Chemical Engineering, Giardini Naxos, Italy April 26-30, pp 47-51.

CAVETT, R.H. (1963). Application of numerical methods to the convergence of simulated processes involving recycle loops. Proc. Am. Petrol. Inst., Vol. 43, pp 57-75.

CHEN, H.S. and STADTHERR, M.A. (1981). A modification of Powell's dogleg method for solving systems of nonlinear equations. Comput. Chem. Engng., Vol. 5, No. 3, pp 143-150.

CHEN, H.S. and STADTHERR, M.A. (1984). On solving large sparse nonlinear equation systems. Comput. Chem. Engng., Vol. 8, No. 1, pp 1-7.

CLARK, J.P. (1977). Exercises in process simulation using FLOWTRAN. CACHE Corp., Cambridge, MA.

CURTIS, A.R., POWELL, M.J.D., and REID, J.K. (1974). On the estimation of sparse Jacobian matrices. J. Inst. Math. Appl., Vol. 13, pp 117-119.

DAVE, A.K. and DUFF, I.S. (1987). Sparse matrix calculations on the Cray-2. Parallel Comput., Vol. 5, pp 55-64.

DUFF, I.S. (1979). Recent developments in the solution of large sparse linear equations. Presented at IRIA Fourth International Symposium on Computing Methods in Applied Sciences and Engineering, December 10-14, Versailles.

DUFF, I.S. (1984). Design features of a frontal code for solving sparse unsymmetric linear systems out-of-core. SIAM J. Sci. Stat. Comput., Vol. 5, No. 2, pp 270-280.

DUFF, I.S. and REID, J.K. (1982). Experience of sparse matrix codes on the Cray-1. Computer Physics Communications, Vol. 26, pp 293-302.

GLOOS, D., HOEHN, W., and SPELLUCCI, P. (1986). Improvement of the runtime behaviour of a frontal solution code. Communications in Applied Numerical Methods, Vol. 2, pp 489-498.

HELLERMAN, E. and RARICK, D. (1972). The partitioned preassigned pivot procedure (P4). In Sparse matrices and their applications (Edited by D.J. Rose and R.A. Willoughby). New York, Plenum Press.

HOOD, P. (1976). Frontal solution program for unsymmetric matrices. Int. J. Num. Meth. Engng., Vol. 10, pp 379-399.

IDA, N. and LORD, W. (1984). Solution of linear equations for small computer systems. Int. J. Num. Meth. Engng., Vol. 20, pp 625-641.

IRONS, B.M. (1970). A frontal solution program for finite element analysis. Int. J. Num. Meth. Engng., Vol. 2, pp 5-32.

PENG, D.Y. and ROBINSON, D.B. (1976). A new two-constant equation of state. Ind. Eng. Chem. Fund., Vol. 15, No. 1, pp 59-64.

STADTHERR, M.A. and HILTON, C.M. (1982). Development of a new equation-based process flowsheeting system: numerical studies. In Selected Topics on Computer-Aided Process Design and Analysis (Edited by R.S.H. Mah and G.V. Reklaitis), AIChE Symposium Series, Vol. 78, No. 214, pp 12-28.

STADTHERR, M.A. and VEGEAIS, J.A. (1985). Process flowsheeting on supercomputers. IChemE Symp. Ser., Vol. 92, pp 67-77.

STADTHERR, M.A. and WOOD, E.S. (1984a). Sparse matrix methods for equation-based chemical process flowsheeting: I. Reordering phase. Comput. Chem. Engng., Vol. 8, No. 1, pp 9-18.

STADTHERR, M.A. and WOOD, E.S. (1984b). Sparse matrix methods for equation-based chemical process flowsheeting: II. Numerical phase. Comput. Chem. Engng., Vol. 8, No. 1, pp 19-33.

THOMPSON, E. and SHIMAZAKI, Y. (1980). A frontal procedure using skyline storage. Int. J. Num. Meth. Engng., Vol. 15, pp 889-910.

VEGEAIS, J.A. and STADTHERR, M.A. (1988). Vector processing strategies for sparse matrix problems in equation-based flowsheeting. Submitted for publication.

ZITNEY, S.E. and STADTHERR, M.A. (1988). Computational experiments in equation-based chemical process flowsheeting. Accepted for publication.