Forschung Wissenschaft

A Matrix Method for Computing Process Networks Containing Multiple Nested Recycle Loops*

By MARCEL T. BRODMANN

Economation AG, CH-7002 Chur (Switzerland)

Summary

Many process networks contain nested recycle loops representing either flow of material or of information. The computation of such a process network typically involves iterative procedures, whereby each nested loop is often converged individually. Methods have been described for arriving at a calculation sequence which minimizes the number of recycle streams and/or variables to be computed iteratively. This paper presents a matrix method which makes use of the results of such network analysis as well as information obtained in solving each individual unit operation block to arrive efficiently at a solution of the complete process network.

The Problem of Computing Recycle Streams

Process Engineers have been facing the necessity to compute recycle streams ever since processing plants have been designed and built. However, the wide use of generalized heat and material balance programs (flowcharting programs) has greatly increased the size and complexity of the process networks which can be calculated. Typically, such a process flowsheet now contains multiple adjacent and nested recycle loops, as for instance the one shown schematically in Fig. 1. Much work has been done in recent years on the subject of analyzing complex process networks[1-4].

To solve for recycle streams in a flowsheet iterative methods have to be used. Fig. 2 shows the block diagrams of a simple process containing four unit processes $F_1 \ldots F_4$, a feed stream X_F, a product stream X_P, and five internal streams $X_1 \ldots X_5$. The streams connecting the boxes represent physical flow of material characterized by its total flow rate, composition, and thermodynamic state. Thus, a stream containing n components corresponds to a vector with $n+2$ elements.

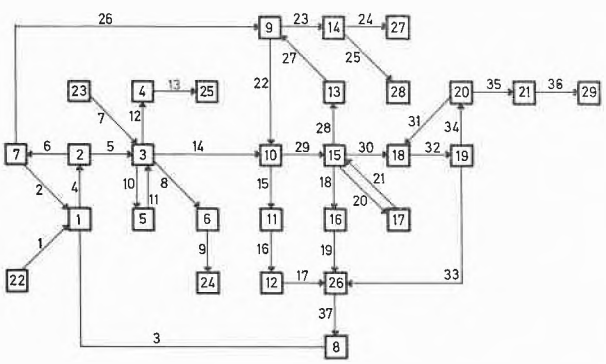


Figure 1. Block Diagram for Alkylation Plant

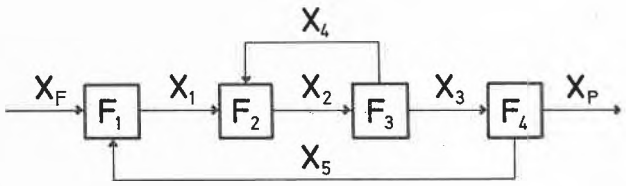


Figure 2. Simple Process Block Diagram

Obviously, a value for the unknown recycle stream X_5 has to be assumed before the equations of unit process F_1 can be solved. Similarly, an estimate of stream X_4 is required to solve for X_2. Once the unit processes F_3 and F_4 are computed, new values of X_4 and X_5 are available, and the calculations can be repeated until a final value of the recycle streams is obtained. Such an iterative process requires much computational effort, and there is always a risk that it will not converge.

Solution Strategies

The selection of a solution strategy to speed up and enhance convergence requires a number of decisions. First, a set of recycle streams has to be chosen. If the system of Fig. 2 is cut at X_2 instead of assuming values for X_4 and X_5, then only one unknown stream has to be estimated and iterated on, thus reducing computation required at each iteration. Several algorithms exist for

* Presented at the International Congress of the European Federation of Chemical Engineering (Use of Electronic Computers in Chemical Engineering) held Paris in April 1973.

[1] R. W. H. Sargent and A. W. Westerberg, *Trans. Instn. Chem. Engrs. 42* (1964) (179) 190–7.
[2] D. G. Steward, *J. SIAM B2* (1965) 345.
[3] F. L. Worley Jr. and R. L. Motard, AIChE *Chem. Engng. Computing 2* (1972) 4.
[4] Y. V. S. Jain and J. M. Eakman, AIChE *Chem. Engng. Computing 2* (1972) 40.

selecting the set of recycle streams that minimizes the number of torn streams or variables[1, 2, 5–7].

However, minimizing the number of recycle streams may not be the best solution strategy. Since the convergence characteristics of the system depend on the chosen set of recycle streams, it can well also increase the number of iterations required to obtain a converged solution. A discussion of the interrelationsship between tearing and convergence is given in [8, 9].

In cases where a recycle block contains multiple recycle streams, a decision has to be made whether each recycle loop should be iterated on individually, or the entire system converged globally. Thus, in Fig. 2 a value can be assumed for recycle stream X_5 and then the inner loop converged before a new value for X_5 is obtained. Alternately, values for X_4 and X_5 are assumed and updated at each iteration, until the total system is converged. Since problems may arise when attempting to solve loops individually[9], solving the system globally seems to be a good general approach.

A last decision in selecting a solution strategy concerns the convergence algorithm to be used. The algorithm most widely known is the successive substitution method, which derives its popularity from its simplicity and economy in storage space[10]. After assuming initial values for all recycle streams, the process network is solved for new values which are re-substituted until the difference between old and new values is less than a given tolerance.

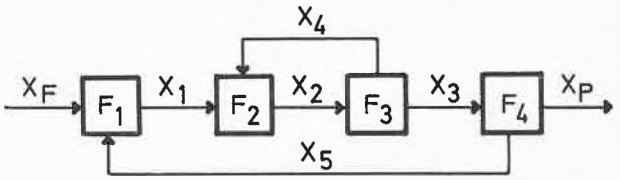
Convergence Acceleration

Simple as this method is, it can be made to converge with many systems. It has one major drawback, though: it is slow to converge, so slow in fact that it may become useless for very large systems. To overcome this difficulty, convergence accelerators are used. Most frequently, starting estimates for the next iteration are extrapolated from the results of previous iterations. This can be done easily for individual recycle streams. However, many iterations are required before the interactions among various recycle loops can be reflected, and thus the convergence speed is improved only gradually.

An alternate technique for improving convergence is to use sensitivity information generated while solving each individual unit operation. Together with information about the structure of the process network, these sensitivities can be used to estimate the interdependence of the recycle variables. A method based on this approach has been developed and implemented in a generalized material balance program and will be described in the remainder of this paper.

Mathematical Description

In mathematical terms, the process network shown in Fig. 2 is represented by the following system of equations:

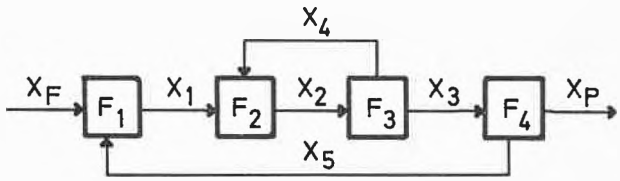


$$F_1\ (X_F,\ X_1,\ X_5) = 0$$
$$F_2\ (X_1,\ X_2,\ X_4) = 0$$
$$F_3\ (X_2,\ X_3,\ X_4) = 0$$
$$F_4\ (X_3,\ X_4,\ X_P) = 0$$

$$F_i(X_j) = 0 = F_i(X_j^*) + \sum_j \frac{\delta F_i}{\delta X_j}\,\Delta X_j,$$

$$i = 1 \ldots 4,$$
$$j = 1 \ldots 5,$$
$$\Delta X_j = X_j - X_j^*,$$

$\frac{\delta F_i}{\delta X_j}$ evaluated at X_j^*.



$$F_1 = X_1 - f_1\ (X_F,\ X_5)$$
$$F_2 = X_2 - f_2\ (X_1,\ X_4)$$
$$F_3 = \begin{vmatrix} X_3 - f_3\ (X_2) \\ X_4 - f_4\ (X_2) \end{vmatrix}$$
$$F_4 = X_5 - f_5\ (X_3)$$

$$\Delta X_i - \sum_{\substack{j \\ j \neq i}} \frac{\delta f_i}{\delta X_j}\,\Delta X_j - \Delta i = 0,$$

$$i,j = 1 \ldots 5$$
$$\Delta i = f_i(X_j^*) - X_i^*$$

Mathematical Description of Process Network

$$\begin{aligned} F_1 \quad & (X_F,\ X_1,\ X_5) = 0 \\ F_2 \quad & (X_1,\ X_2,\ X_4) = 0 \\ F_3 \quad & (X_2,\ X_3,\ X_4) = 0 \\ F_4 \quad & (X_3,\ X_4,\ X_P) = 0, \end{aligned} \tag{1}$$

where $F_1 \ldots F_4$ are the sets of (nonlinear) equations describing the unit operations of the flowsheet, and X_F, $X_1 \ldots X_5$, X_P the vectors characterizing the interconnecting streams.

Linearizing around an estimated solution vector X_j^* gives

$$F_i(X_j) = 0 = F_i(X_j^*) + \sum_j \frac{\delta F_i}{\delta X_j} \Delta X_j, \tag{2}$$

$$\begin{aligned} i &= 1 \ldots 4, \\ j &= 1 \ldots 5, \\ \Delta X_j &= X_j - X_j^*, \\ \frac{\delta F_i}{\delta X_j} &\text{ evaluated at } X_j^*. \end{aligned}$$

Since each unit operation is programmed with its product streams as output variables, equations (1) can also be written

$$\begin{aligned} F_1 &= X_1 - f_1 \quad (X_F,\ X_5) \\ F_2 &= X_2 - f_2 \quad (X_1,\ X_4) \\ F_3 &= \begin{vmatrix} X_3 - f_3 & (X_2) \\ X_4 - f_4 & (X_2) \end{vmatrix} \\ F_4 &= X_5 - f_5 \quad (X_3), \end{aligned} \tag{3}$$

thereby dividing the set of equations F_3 into two subsets for the two product streams.

The linearized system then becomes

$$\Delta X_i - \sum_{\substack{j \\ j \neq i}} \frac{\delta f_i}{\delta X_j} \Delta X_j - \Delta_i = 0, \tag{4}$$

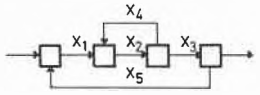
$$\begin{aligned} i, j &= 1 \ldots 5 \\ \Delta i &= f_i(X_j^*) - X_i^* \end{aligned}$$

For the system of Fig. 2, this can be written in matrix form as shown in Fig. 3.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & -\frac{\delta X_1}{\delta X_5} \\ -\frac{\delta X_2}{\delta X_1} & 1 & 0 & -\frac{\delta X_2}{\delta X_4} & 0 \\ 0 & -\frac{\delta X_3}{\delta X_2} & 1 & 0 & 0 \\ 0 & -\frac{\delta X_4}{\delta X_2} & 0 & 1 & 0 \\ 0 & 0 & -\frac{\delta X_5}{\delta X_3} & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \Delta X_1 \\ \Delta X_2 \\ \Delta X_3 \\ \Delta X_4 \\ \Delta X_5 \end{bmatrix} = \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \Delta_3 \\ \Delta_4 \\ \Delta_5 \end{bmatrix} \tag{5}$$

Figure 3. Matrix Representation of Linearized System Equations

The structure of the JACOBI matrix in Fig. 3 can be read directly off the stream adjacency matrix of the process network (Fig. 4).



TO \ FROM	X_1	X_2	X_3	X_4	X_5
X_1	0	0	0	0	1
X_2	1	0	0	1	0
X_3	0	1	0	0	0
X_4	0	1	0	0	0
X_5	0	0	1	0	0

Figure 4. Stream Adjacency Matrix

Each non-zero entry in the stream adjacency matrix corresponds to a non-zero submatrix in the system Jacobian, containing the partial derivate or sensitivity of the product stream X_i with respect to the feed stream X_j. The sensitivities can be generated by the individual unit operation programs and stored for further use by the convergence routine. This requires only marginal effort when a sensitivity based method is used to solve the unit operation equations.

Reducing the System Jacobian

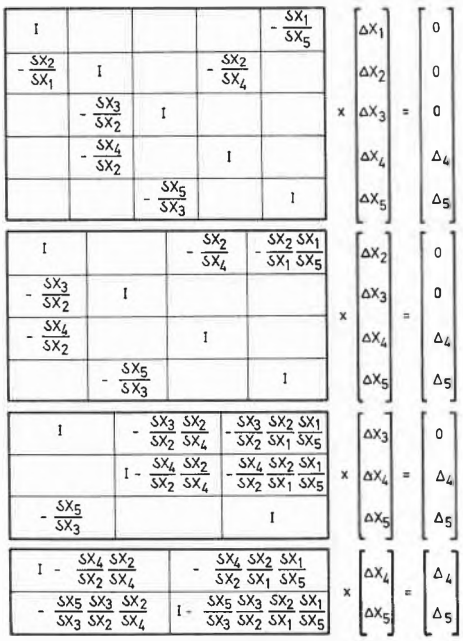


Figure 5. Reduction of System Jacobian

[5] D. I. RUBIN, *Chem. Engng. Progr.* Symposium, Ser. No. 37, *58* (1962) 54.
[6] W. LEE and D. F. RUDD, AICHE *J. 12* (1966) (6) 1184.
[7] J. H. CHRISTENSEN and D. F. RUDD, AICHE *J. 15* (1) (1969) 94.
[8] F. C. EDIE and A. W. WESTERBERG, *Chem. Engng. J. 2* (1971) (1) 17.
[9] F. C. EDIE and A. W. WESTERBERG, AICHE *Chem. Engng. Computing 1* (1972) 35.
[10] P. E. BARKER and A. J. LEATHER, *I. Chem. Engng. Symposium*, Series No. 35, *2* (1972) 27.

The system Jacobian is used to generate new guesses of the recycle variables during iteration. To do this, advantage is taken of the fact that in equation (5) all Δ_i's are zero except those belonging to recycle streams. Rows with zero right-hand side and their corresponding columns can be removed from the matrix through substitution in the following way:

> Any non-zero, non-diagonal entry *ij* of row *i* is premultiplied by all non-zero, non-diagonal entries *ki* of column *i*, and the resulting product matrices are subtracted from the submatrix in field *kj*.

This procedure is illustrated in Fig. 5 for the matrix equation (5), assuming that X_4 and X_5 have been selected as recycle streams. Once the system Jacobian has been reduced, it can be inverted to generate new guesses for the iteration scheme.

Implementation of Method

To implement the matrix convergence method described above, the tasks assigned to the System Executive and to the Unit Operations programs had to be coordinated with those performed by two specially written modules, the Network Analyzer and the Convergence Routine. Fig. 6 shows how the various tasks have been divided among these programs.

The Systems Executive has two main functions: it coordinates and directs the execution of the other modules, and it stores and retrieves bulk data, primarily results from the network analysis phase and sensitivities generated by the Unit Operations.

Network Analysis

The Network Analyzer is at the core of the matrix convergence system. Its first task is to identify recycle loops, to form recycle blocks (maximal cyclical subnets), i.e., groups of unit operations which have to be solved simultaneously by iteration, and to establish the precedence order of unit operations and recycle blocks. The tracing method described by STEWARD[2] is employed to this end. However, a stream adjacency matrix is used instead of a node adjacency matrix. This allows parallel streams to be handled easily while requiring only slightly more storage space. As an additional advantage, the structure of the system Jacobian can be read directly off the matrix. The stream adjacency matrix is stored in the form of linked row lists and thus occupies two storage positions for each non-zero element.

A second task of the Nework Analyzer is to select the recycle streams within each recycle block. The program offers two options—to minimize the number of tear

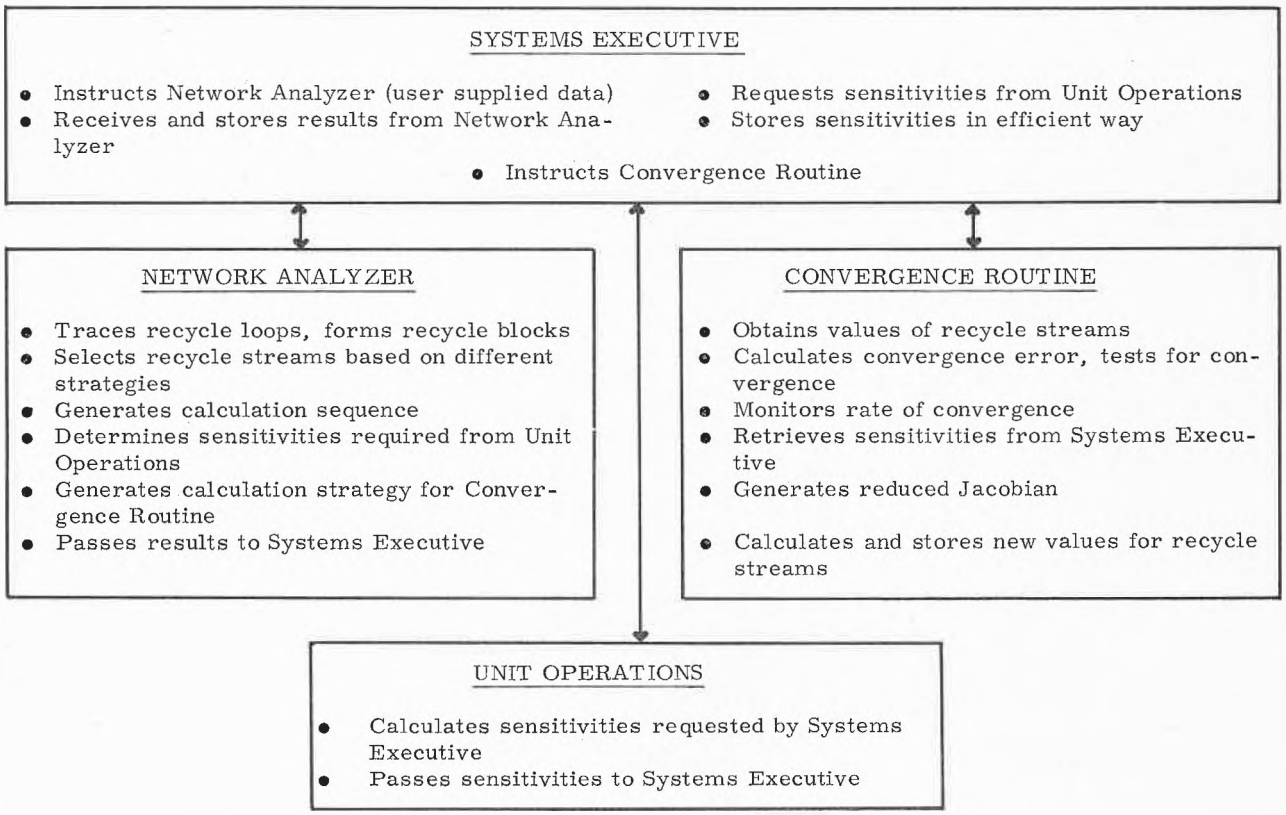
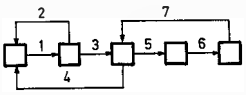
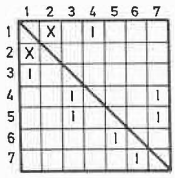
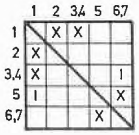


Figure 6. Distribution of Tasks among System Modules







Finding Subcycles of Recycle Block

points, or to choose the recycle streams in a way that no Unit Operation with all unknown feed streams appears in the initial calculation sequence. In addition, the user can specify what streams he wants to have torn.

The algorithm which minimizes the number of break points works in two steps. First, chains of unit operations are collapsed by removing streams until a matrix is obtained that consists only of subcycles of length two. The method and subroutine used are the same as those described for reducing the system Jacobian. From the resulting matrix which contains symmetrical entries around the main diagonal, the minimal tear solution is found in a second step. Once the recycle streams have been selected, the calculation sequence within the recycle block can be generated using STEWARD's algorithm[2].

Finding the sensitivities which have to be calculated by the Unit Operations requires no additional effort, since sensitivities are already identified by the row entries of the stream adjacency matrix. Similarly, the calculation sequence which the Convergence Routine has to follow to reduce the system Jacobian is implicitly contained in the adjacency matrix. However, lists of intermediate calculation results and explicit calculation sequences are prepared to help organize data storage and to speed up convergence calculations during iteration.

Unit Operations

The Unit Operations have been organized to operate in two phases, a calculation phase and a derivative-generation phase. The derivative-generation is skipped whenever the successive substitution mode has been specified, or when the appropriate row lists of the stream adjacency matrix are empty.

Derivative information obtained while solving the unit operation equations is used whenever possible for computing the requested sensitivities. In some Unit Operation blocks the system equations have been differentiated analytically and coded as a separate module. Others, especially smaller routines, obtain sensitivities by perturbation.

Convergence Routine

A special start-up phase is necessary when using the matrix convergence method, especially in connection with the minimum recycle stream option. The algorithm active under this option frequently selects recycle streams for which no initial estimate is available, resulting in zero input Unit Operations. Obviously, no derivatives can be computed when the feed is unknown. In addition, the starting value might be too far off to obtain convergence. To avoid this difficulty, successive substitution is always used initially for at least two iterations.

Evaluation of Results

The matrix convergence method has been tested on a number of process flowsheets, and results have been compared with successive substitution. Though no quantitative evaluation is given at this time, some general observations can be made.

The method worked best in cases which did not converge—or only very slowly—with successive substitution. In general, a converged solution was obtained in less than eight iterations. Most of these cases were highly interdependent networks. Furthermore, the method seems to work better with multiple recycle streams rather than with a minimum tear solution.

In some cases, the number of iterations could be reduced when compared with successive substitution, though overall computation time remained about the same. Where a fast solution can be obtained with successive substitution, the matrix method is clearly inferior.

Attempts have been made to further enhance the method by reducing computer time and storage requirements. One means to achieve this is to use only the diagonal elements in the sensitivity matrices. Improvements can also be made by alternating between the successive substitution and the matrix methods.

A final word of caution is in order. The method is sensitive to coding errors in generating the sensitivities. Thus, the sensitivity modules have to be tested carefully before experiments can be run to evaluate convergence properties. However, this is the case in most programming work.