AN ABSTRACT OF THE THESIS OF

Carlos C. Sanz for the degree of Master of Science

in Chemical Engineering presented on March 15, 1985.

Title: Strategies for Process Flowsheeting

Abstract approved: Redacted for privacy

Two packages of subroutines were developed to perform material balances on chemical processes using the simultaneous modular approach and the equation-based approach. The performances of these packages were compared for five different processes under at least two conditions: one with no design specifications, and one with two or more design specifications.

The equations arising from chemical process simulation using the simultaneous modular approach and equation-based approach are nonlinear. Therefore, two subroutines were developed to solve systems of nonlinear equations using a modification of Powell's dogleg method as proposed by Chen and Stadtherr. One of the nonlinear solver subroutines uses sparse matrix techniques and updates the Jacobian through Schubert's formula. The other uses full matrix techniques and the Jacobian is updated through Broyden's formula. Both subroutines were tested with five problems and the results compared.

The results obtained with the two packages of subroutines and the nonlinear solver subroutines compared well with similar problems from the open literature.

Strategies for Process Flowsheeting

by

Carlos C. Sanz

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed March 15, 1985

Commencement June 1985

APPROVED:

Redacted for privacy

Professor of Chemical Engineering in charge of major

Redacted for privacy

Chairman of Chemical Engineering Department

Redacted for privacy

Dean of Graduate School

Date thesis is presented

Typed by Meredith Turton for Carlos Sanz

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to Dr. Aydin Konuk for his guidance and help throughout this work and, most importantly for his friendship.

For the financial support, I am indebted to the Chemical Engineering department (Teaching Assistantship, 1983-84) and to Itau S.A. Planejamento e Engenharia (Scholarship, 1982-85). I am also indebted to Milne Computer Center; without the grant they awarded to me I could not perform the extensive computer work required by this thesis.

I am thankful to all my professors at Oregon State University -- Dr. J. Frederick, Dr. L. Garrison, Dr. J. Knudsen, Dr. O. Levenspiel, Dr. R. Mrazek and Dr. C. Wicks -- for sharing part of their knowledge with me. In addition, I am thankful to Dr. K.L. Levien for the numerous suggestions, and the effort he put forth in getting this thesis in shape!

Finally, and most importantly, I am indebted to my parents, Cesar and Margarita, my parents in the U.S.A., Dorothy and Carlo Rocca, and (yes, I took my vitamins) to Angela, for their understanding, patience and love throughout the course of this study. Certainly the telephone company will miss me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

STRATEGIES FOR PROCESS FLOWSHEETING

INTRODUCTION

Computer programs to simulate chemical processes are widely used in industry for a number of purposes. Important uses include: the design of new plants, optimization of new and existing plants and, more recently, optimal control of processes and the training of plant personnel.

The use of computer programs in the chemical industry started in the mid-fifties with programs written to perform mass and energy balances around single units, such as distillation columns, evaporators, flash drums, etc. As computers became faster and capable of greater storage, the programs written for single units were integrated to simulate entire processes.

The beginning of the widespread use of computers to simulate chemical processes started in the early sixties. At the same time, the literature in the area started to flourish; examples of it are today's "classical" articles by Rosen (1962) and Cavett (1963). During the sixties and seventies, the chemical industry invested heavily in this field. Today, there are thousands of articles in the open literature, and some journals are specific to this field. However, 90 percent of the articles are concerned with one specific solution method: the sequential modular approach.

Recently, two other approaches have received considerable attention: the equation-based approach and the simultaneous modular approach.

In this work, two packages of subroutines were developed to perform simulations of processes through the equation-based approach and the simultaneous modular approach. Another package of subroutines, SIMFLOW (Kayihan, 1979), was slightly modified and used to simulate processes through the sequential modular approach.

The aim of these packages is to serve as a teaching aid to sophomore-level stoichiometry classes, thus, only material balances are considered. In addition, we intend to verify some of the results reported in the literature on the performance of various simulation methods.

Chapter I presents a review of the three methods; the equation-based approach, the simultaneous modular approach, and the sequential modular approach. Chapter II presents the basic methods to solve systems of nonlinear equations. Chapter III presents the structure of the libraries developed for the equation-based approach and for the simultaneous modular approach. In Chapter IV, five test problems are solved using all three approaches and results are compared. Chapter V presents the conclusions and the future work. In addition, there are appendices with listings of all programs (libraries), a short description of a modification of an algorithm used to solve nonlinear equations, and the equations describing material balances used in the sequential modular approach.

CHAPTER I

BACKGROUND

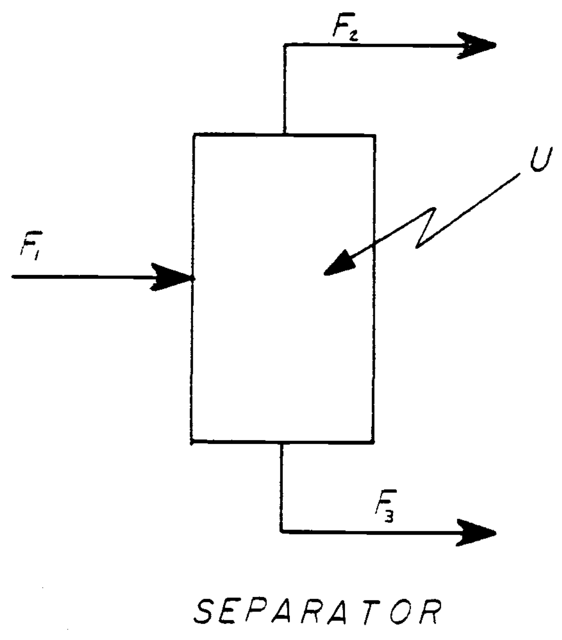
The Sequential Modular Approach

The sequential modular approach uses a library of unit operations modules (subroutines) which perform material and energy balances. The modules are written so that each calculates its output stream(s) variables, given its input stream(s) variables and any other parameters required (see figure I-1).

To simulate an entire process, or a portion of it, all modules describing the process are executed one after the other, following a specific order. Figure I-2 shows a simple flowsheet which will exemplify how the sequential modular approach functions.

In the example shown in figure I-2, F_i represents a vector of stream variables: total molar flow rate, mole fraction of each component, enthalpy, temperature and pressure. For each specific case studied, a subset of those variables will be used. In addition, each piece of equipment may require a set of equipment parameters. In the example of figure I-2, γ_r is the conversion of a reactant and k_J is the equilibrium constant for the $j^{\underline{th}}$ component in the flash drum. Assuming that γ_r, k_J, and F_1 are known, our task is to calculate F_2, F_3, and F_4.

Both modules, reactor and flash, are written so that the output stream variables are calculated, provided that the input stream and equipment parameters are known. In our example, one would first



F = STREAM VARIABLES

U = EQUIPMENT PARAMETERS

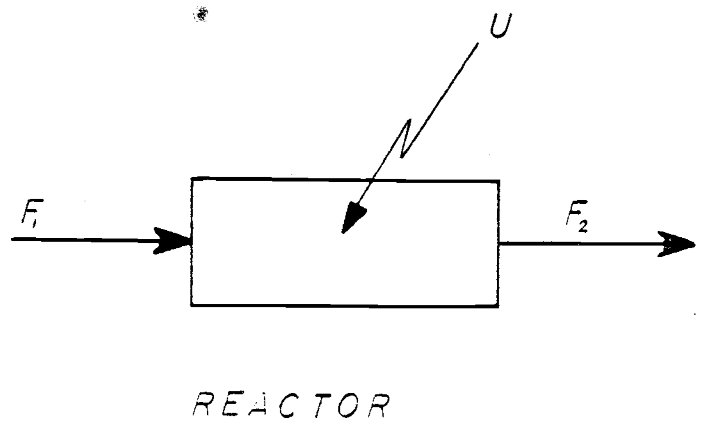
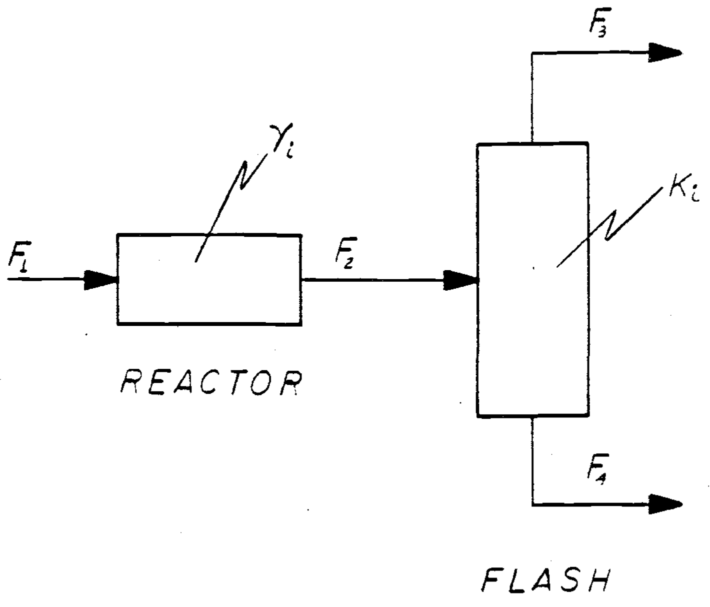


Figure I-1. Example of Simple Modules for the Sequential Modular Approach



γ_i = CONVERSION

K_i = EQUILIBRIUM CONSTANT

Figure I-2. Simple Flowsheet

calculate stream F_2 using the reactor module and then streams F_3 and F_4 using the flash module. It looks (and it is!) very simple because that is the natural order of calculations we would follow if we were to perform those calculations by hand. This characteristic of the sequential modular approach (following the natural order of calculations) is one of the main reasons for its popularity among process engineers. Given a process, it is easy to build a computer model of the entire process using 10-20 different modules.

What makes process simulation difficult are recycle streams. Figure I-3 shows a slight modification of the process from figure I-2. Now we have stream F_4 as a recycle, and a mixer is added.

As in the previous example, stream F_1, γ_I, and k_J are known. However, none of the streams entering a module are fully known. Although stream F_1 is specified, we cannot perform mixer calculations because stream F_4 is not known.

One way to solve the problem would be to "guess" all stream variables F_2, perform reactor, flash, and mixer calculations and compare the guessed and calculated values of F_2. If the guessed value agrees with the calculated value (within a certain tolerance), the solution has been found. Otherwise, a new guess would be given. This routine continues until a solution is found. It is clear that the method is awkward. It is feasible only if there are a few components and one recycle stream. With two or more recycle streams and 10-15 components, the method is impossible. In the previous example, stream F_2 is known as the tear stream. There are several

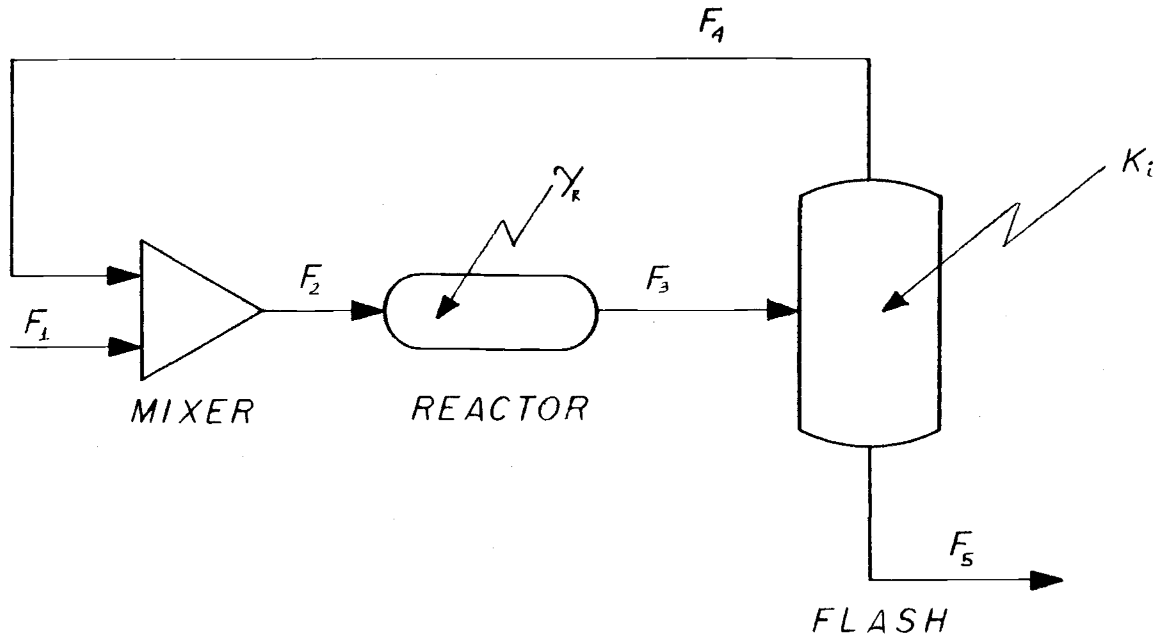


Figure I-3. Simple Flowsheet with a Recycle Stream

algorithms to choose the best tear stream; an excellent review may be found in Hlavacek (1977).

To overcome the problem of guessing new values, one has to remember that the process is described mathematically by a series of equations. In our example, we have:

$$F_3 = \Psi_1(F_2) \qquad \text{(I-1a)}$$

$$F_4 = \Psi_{2a}(F_3) \qquad \text{(I-1b)}$$

$$F_5 = \Psi_{2b}(F_3) \qquad \text{(I-1c)}$$

$$F_2 = \Psi_3(F_1,F_4) \qquad \text{(I-1d)}$$

Ψ_1, Ψ_2, and Ψ_3 are functions relating output streams with input streams and equipment parameters. Clearly Ψ_1, Ψ_2, and Ψ_3 are series of calculations performed by the reactor, flash and mixer modules. If we use equation I-1a to eliminate F_3 from equation I-1b, and equation I-1b to eliminate F_4 from equation I-1d, we get:

$$F_2 = \Psi_3(F_1,\Psi_{2a}(\Psi(F_2))) \qquad \text{(I-2a)}$$

or

$$F_2 = \Psi_4(F_1,F_2) \qquad \text{(I-2b)}$$

Equation I-2b is actually a set of nonlinear equations, called stream connection equations, which can be solved using various methods. Most commercial programs using the sequential modular approach use the method proposed by Wegstein (1958). Wegstein's method is very popular because, to use the method, the equation being solved must be in the form of X = F(X); the stream connection equations are naturally in this form. In addition, the method converges quickly to the solution, provided that the system of equations has only mild nonlinearities, and each equation of the system is dominated by one variable (Westerberg et al., 1979).

The drawback of Wegstein's method appears when the simulation is a "controlled" simulation (or, simulation with constraints). In a controlled simulation, we allow one or more variables to be "free" to meet some design specification. Let us suppose that we want the flow rate (W) in the vector of variables F_4 to be equal to some value C. To meet this specification, γ_r is a free variable. Now the system of equations becomes:

$$F_3 = \psi_1(F_2, \gamma_r) \tag{I-3a}$$

$$F_4 = \psi_{2a}(F_3) \tag{I-3b}$$

$$F_5 = \psi_{2b}(F_3) \tag{I-3c}$$

$$F_2 = \psi_3(F_1, F_4) \tag{I-3d}$$

$$W = \psi_4(F_2, \gamma_r) = C \tag{I-3e}$$

The system is reduced to:

$$F_2 = \psi_5(F_1, F_2, \gamma_r) \tag{I-4a}$$

or

$$F_2 = \psi_5(F_2, \gamma_r) \quad \text{for specified } F_1 \tag{I-4b}$$

and

$$W = \psi_4(F_2,\gamma_r) = C \qquad (I\text{-}4c)$$

Wegstein's method can be applied to equation I-4b, stream connection equations; however, equation I-4c, the design specification equation, is not in the form required by Wegstein's method, that is, $X = F(X)$.

To overcome this problem, two levels of iterations are used. One is internal, and the free variable γ_r is assumed known. The other is external, and γ_r is manipulated in order to satisfy equation I-4c. Typically, two different nonlinear solver subroutines are used. For example: equation I-4c will be solved using the secant method and equation I-4b using Wegstein's method. A block diagram of this strategy is shown in figure I-4.

The internal level of iterations solves <u>stream connection equations</u> using Wegstein's method where the unknowns are the tear variables. The external level solves <u>design specifications equations</u> (in the example, γ_r is the unknown) using the secant method.

Actually Wegstein's subroutine solves a series of different simulation problems without constraints; each problem has a specific value for γ_r updated by the external secant iterations. If Wegstein's method performs an average of M iterations per secant iteration to converge the tear stream, and the secant subroutine performs K iterations for the convergence of the design variable, the total number of flowsheet evaluations will be $L = M * K$.

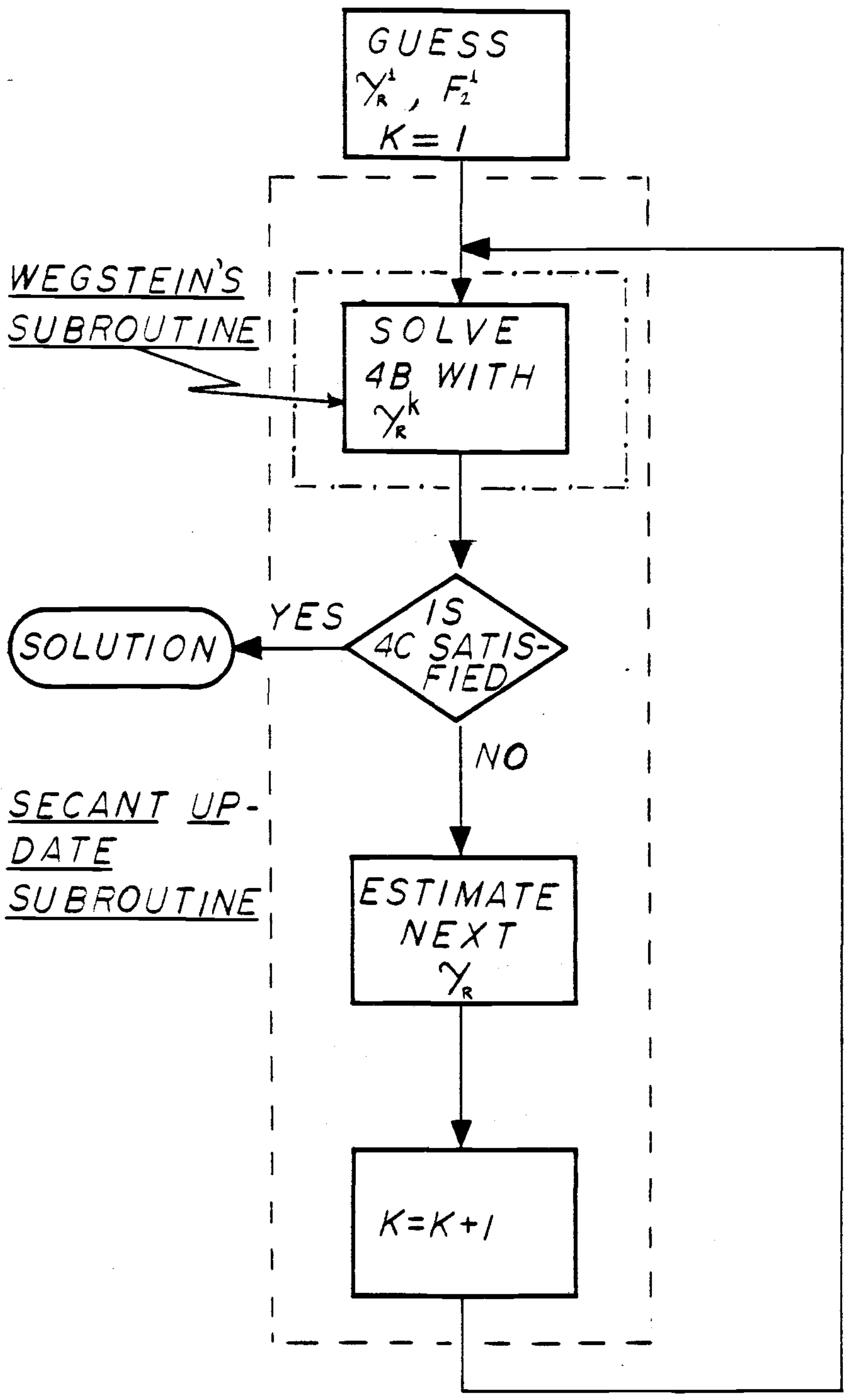


Figure I-4. Block Diagram of the Strategy Used to Simulate Process with Design Specifications

Depending on the complexity of the process and the initial guesses given, the total number of flowsheet evaluations may be prohibitively high.

Another approach used is to insert a control module. The control module behaves like a PID controller. The set point of the control module will be the design specification imposed on the system. After each iteration, the control module checks the error. Depending on the error's magnitude, the control module will change the manipulated variable (free variable) according to pseudo-PID parameters used. More details about the control module may be found in Westerberg et al., 1979.

Although the handling of design constraints is difficult and time-consuming, the sequential modular approach for process simulation is widely used. As Chen (1982) pointed out, the main reasons for the success of the sequential modular approach are:

1. The computer model of the process and the actual flowsheet are closely related, and it is easy for the process engineer to write the program used by the computer.
2. Each module is written to "stand alone" so it can be thoroughly tested, and the module can be very efficient and robust.
3. Special programs for the different types of equipment or nonstandard equipment can be easily incorporated into the package of subroutines.

4. It is very easy to implement; it does not require much computer storage (besides that required by each module).

A more complete overview of the sequential modular approach may be found in Westerberg et al. (1979) and Myers et al. (1976).

The Simultaneous Modular Approach

The simultaneous modular approach, as it will be defined in this work, does not differ much from the sequential modular approach. The modules used will be the same modules used in the sequential modular approach. However, the Newton-Raphson method (or a modification of it) will be used to solve the set of nonlinear equations.

In order to use the Newton-Raphson method, the set of nonlinear equations should be in the form

$$G(x) = 0 \tag{I-5}$$

For example, to solve equations I-4b and I-4c, one simply rearranges these equations to:

$$G_1(F_2,\gamma_r) = F_2 - \psi_5(F_2,\gamma_r) = 0 \tag{I-5a}$$

$$G_2(F_2,\gamma_r) = C - \psi_4(F_2,\gamma_r) = 0 \tag{I-5b}$$

Now both equations can be solved using the Newton-Raphson method, and only one nonlinear solver subroutine will be used. It is easy to see that we can add any constraint to the problem, provided that the constraint has a physical meaning. In short, stream connection equations (I-5a) and design specifications (I-5b) are solved simultaneously, instead of using two levels of calculations. It should be emphasized that the unknowns in equation I-5a (stream connection equations) are the tear variables. The non-tear stream variables do not appear as unknowns in equation I-5a.

Let μ be a vector of equipment parameters, C a vector of constraints, and T a vector of tear streams. With the simultaneous modular approach, any process simulation is arranged to be the solution of a set of nonlinear equations in the form:

$$G(T_i, \mu_k, C_k) = 0 \qquad \text{(I-6)}$$

This approach was successfully used by Perkins (1979) and Chen (1982). They report promising results in general simulations (controlled or not). Moreover, Chen also reports good results in optimization problems. The literature, at least the open literature, does not report any commercial program using the approach. Some authors mention that ASPEN (MIT and DOS) gives the option of using the approach. Unfortunately, thus far there are no articles in the literature about that feature.

It must be pointed out that the use of the Newton-Raphson method for process simulation is not new. Cavett (1963) presented a few examples using the approach; none of them were controlled simulations. Furthermore, there is some controversy about the use of the name "simultaneous modular." Perkins (1979) presents the method as sequential modular with a different method to improve convergence. Chen (1982) presents the same method as Perkins as the simultaneous modular approach. On the other hand, Westerberg et al. (1979) formulate the simultaneous modular approach in a different manner. They present the approach as it was first suggested by Rosen (1962). In that approach, each equipment module has two different versions. One is the rigorous model, and the other is a simple model which approximates the rigorous model. The simple model relates each output value approximately to linear combinations of all input values. For example, the flash unit of figure I-3 would be approximated by:

$$F_4 = a_{43}F_3 \tag{I-7a}$$

$$F_5 = a_{53}F_3 \tag{I-7b}$$

In a simulation, one would start guessing all a_{ij} and then solving a linear system of equations. Using the example of figure I-3, the linear system to be solved is:

$$\begin{aligned} F_2 - F_4 &= F_1 \\ F_3 - a_{32}F_2 &= 0 \\ F_4 - a_{43}F_3 &= 0 \\ F_5 - a_{53}F_3 &= 0 \end{aligned} \tag{I-8}$$

Assuming that F_1 (feed to the mixer) is known, the system of linear equations I-8 is easily solved. Using the rigorous model of each piece of equipment and the F_i variables just found, one recalculates all a_{ij}. If the recalculated values are essentially equal to the ones previously guessed, a solution has been found; if not essentially equal, all a_{ij} are updated again and the linear system I-8 is again solved. This procedure would continue until a solution is found.

This method did not meet with much success because the linear approximation is poor for some of the equipment. The linear model cannot predict, for example, the influence of an inlet stream temperature on the output streams of a flash drum. Some authors (Mahelec et al., 1979) used a strategy similar to that of Rosen, but used difference split-fraction models; they reported good results with this approach. It should be noted that if one increases the complexity of the "simple" modules, this approach will tend towards Newton's method. Newton's method linearizes all functions around a certain point in each iteration, but all interaction between the variables will be accounted for.

Also in Rosen's approach, each stream is a tear stream, whereas in the approach used in this work, we use blocks of equipment with the least possible number of tear streams. The fundamental differences between Rosen's approach and the approach used in this work are the number of streams torn and the model used to approximate the rigorous model (or blocks of equipment).

The Equation-Based Approach

In the equation-based approach, all equations describing the process are solved simultaneously. Let us consider again the simple flowsheet of figure I-3. We will assume that there are only 3 components and that the reaction in the reactor is: $A + 2B \rightarrow 3C$. Subscripts $j = 1, 2,$ and 3 will be used for components A, B, and C, respectively. In addition, W_i is the total flow rate of stream i (mol/hr), X_{ij} is the mole fraction of the $j^{\underline{th}}$ component in the $i^{\underline{th}}$ stream, γ_i is the conversion with respect to the $i^{\underline{th}}$ component, and k_i are the equilibrium constants in the flash drum. It should be noted that only mass balances will be considered. The system of equations is:

Mixer

$$W_2 = W_4 + W_1 \tag{I-8a}$$

$$X_{21} = (W_1X_{11} + W_4X_{41})/W_2 \tag{I-8b}$$

$$X_{22} = (W_1X_{12} + W_4X_{42})/W_2 \tag{I-8c}$$

$$X_{23} = 1.0 - X_{21} - X_{22} \tag{I-8d}$$

Reactor

Let a, b, and c be the stoichiometic coefficient of species A, B, and C; $r = W_2X_{21}\gamma_1/-a$

$$W_3 = W_2 + r(c - a - b) \tag{I-8e}$$

$$X_{31} = (W_2X_{21} + ar)/W_3 \tag{I-8f}$$

$$X_{32} = (W_2X_{22} + br)/W_3 \tag{I-8g}$$

$$X_{33} = 1 - X_{31} - X_{32} \tag{I-8h}$$

Flash

$$W_3X_{31} = W_4X_{41} + W_5X_{51} \tag{I-8i}$$

$$W_3X_{32} = W_4X_{42} + W_5X_{52} \tag{I-8j}$$

$$W_3X_{33} = W_4X_{43} + W_5X_{53} \tag{I-8k}$$

$$X_{41} = k_1X_{51} \tag{I-8l}$$

$$X_{42} = k_2X_{52} \tag{I-8m}$$

$$X_{43} = k_3X_{53} \tag{I-8n}$$

$$X_{41} + X_{42} + X_{43} = 1 \tag{I-8o}$$

$$X_{51} + X_{52} + X_{53} = 1 \tag{I-8p}$$

As there are 16 equations and 24 variables we have to specify 8 variables. The advantage of this approach is now clear. If we specify W_1, X_{11}, X_{12}, X_{13}, γ_1, k_1, k_2, and k_3, the problem is a simple simulation. If we specify X_{51}, X_{52}, W_5, X_{41}, X_{42}, W_1, X_{11}, and X_{12}, we have a design problem (or a controlled simulation). In the equation-based approach there are no distinctions between constrained or unconstrained problems. In addition, there is no need to search for a good tear stream as in the sequential modular or simultaneous modular. Also, it is well known that the sequential modular approach is very inefficient when there are more than 2 recycle streams and several constraints (Stadtherr and Wood, 1984). Again, the equation-based approach is immune to these problems.

The question that arises is, "Why isn't the equation-based approach widely used?" First, the execution time and the computer memory required can be prohibitive. A simple industrial simulation will have 5-10,000 equations. Some authors report simulations of complete chemical plants that would require the solution of 100,000 equations! Second, the industry has heavily invested in the development of the sequential modular approach. Why invest in a new method with so many challenges -- the solution of large systems of nonlinear equations -- when the sequential modular approach solves almost all industrial problems? Third, the sequential modular approach is user-friendly. It is easy for the process engineer to construct the sequence of modules that decribes a chemical process with the sequential modular approach. In addition, in the sequential modular approach, when the simulation fails valuable information can be obtained; in the equation-based approach, however, when a failure occurs, almost no useful information can be obtained.

Typically, a Newton-based approach is used to solve such large systems. Figure I-5 shows the structure of the Jacobian of the system of equations I-8a through I-8p; it is clear that the Jacobian is sparse. To speed up execution time, sparse matrix techniques must be used, not only to solve the linear system at each iteration, but throughout the nonlinear solver subroutine. The use of sparse matrix techniques will decrease both execution time and memory required by the program. Also, the linear system solver must use

VARIABLES

EQUATIONS	W_2	X_{21}	X_{22}	X_{23}	W_3	X_{31}	X_{32}	X_{33}	W_4	X_{41}	X_{42}	X_{43}	W_5	X_{51}	X_{52}	X_{53}
1	●								●							
2	●	●							●	●						
3	●		●						●							
4		●	●	●												
5	●	●			●											
6	●	●			●	●										
7	●		●		●		●									
8						●	●	●								
9					●	●			●	●			●	●		
10					●		●		●		●		●		●	
11					●			●	●			●	●			●
12										●				●		
13											●				●	
14												●				●
15										●	●	●				
16														●	●	●

Figure I-5. Jacobian of the System of Equations I-8a through I-8p

some reordering technique to minimize the creation of nonzero elements during Gaussian elimination.

Another point that should be considered is the evaluation of the Jacobian at every iteration. For systems of nonlinear equations, the Jacobian is usually evaluated numerically by forward differences; this procedure may be time-consuming. One way to reduce the problem is to evaluate the Jacobian once and then, instead of calculating the Jacobian numerically every iteration, one update the Jacobian using the information obtained from an increment given to each variable and the resultant variation in each function. Broyden (1965), Schubert (1970), and Broyden (1971) presented methods to update the Jacobian. The first is more suited to full matrix cases, and the second and third are more suited for sparse matrices. The literature is abundant in articles reviewing and comparing these three methods: Gallun and Holland (1980), Crowe (1984), Mah and Lin (1980), and Lucia (1982).

In addition to the use of sparse matrix techniques and the update of the Jacobian through Broyden's or Schubert's method, good initial values should be given to all variables. For small simulations, this will not be a problem, but when the simulation has 10,000 variables, some automatic procedure to initialize all variables must be incorporated. Good initial guesses are also required because Newton-based methods do not converge to a solution if the initial values of the variables are too far from the solution.

Unfortunately, there is no such method that guarantees a solution. A good improvement was suggested by Powell (1970); his method will be discussed in more detail in later chapters. Powell's method has received much attention lately, and the reviews are generally favorable. See, for example, Chen and Stadtherr (1981).

Today, at least five programs using the equation-based approach in various stages of development exist: SPEEDUP (Imperial College, U.K.), ASCEND II (Carnegie-Mellon), QUASILIN (University of Cambridge), FLOWSIM (University of Connecticut/Control Data), and SEQUEL (University of Illinois).

Recently, SPEEDUP went through a detailed evaluation in order to determine whether equation-based systems could be used effectively for process simulations. The evalution was carried out by Gupta and coworkers from Exxon Corporation and Prime Computer, Inc. In short, Gupta et al. compared the performance of SPEEDUP with the sequential modular program, COPE, used by Exxon. They report that SPEEDUP is not yet commercially competitive with the sequential modular approach. The evaluation was carried out using the following test problems:

1. Cavett problem (four flash drum problem)
2. Heat exchanger network
3. Five-stage absorber
4. Three tower separation

5. Steam system
6. Compressor network

The most relevant points in the evaluation were:

1. SPEEDUP proves that equation-oriented approach can solve large problems handled by a commercial sequential modular approach.
2. SPEEDUP demonstrates advantages of the equation-based architecture over the sequential modular approach.
3. The equation-based approach is well-suited for problems having a large number of design specifications (constraints) and it provides a flexible environment for solving new problems that cannot easily be solved with existing tools.
4. There is potential for efficient implementation of an optimization capability.
5. In some cases, it does not converge, unless good initial guesses are given.
6. It is relatively inefficient, especially for smaller problems.
7. It is difficult for users to determine the cause of divergence, when a simulation fails.

More details may be found in Gupta et al. (1984). Overall, the results obtained by Gupta et al. are exciting, to say the least. A comprehensive evaluation of the equation-based approach was performed, and the key problems were detected. What makes it more important is that the evaluation was performed to see the commercial potential of an equation-based approach, by a typical user of process simulators (Exxon). The results obtained with SPEEDUP show that some of the challenges encountered in the equation-based approach were solved, and we can forecast the commercial use of the approach in the near future.

CHAPTER II

SOLVING SETS OF NONLINEAR EQUATIONS

As shown in the last chapter, every process simulation with recycle streams and/or design specifications requires the solution of a system of nonlinear equations. Most commercial sequential modular programs use Wegstein's method to solve the set of nonlinear equations. Some programs have the option of using a "blend" of Wegstein's method with the successive substitution method. Programs using the equation-based approach or the simultaneous modular approach generally use the Newton-Raphson method, or a modification of it. Before we present a modification of the Newton-Raphson method used in this work, we will briefly review the three methods mentioned above.

Successive Substitution Method

The use of the successive substitution method is quite simple. In a process simulation using the sequential modular approach, the set of nonlinear equations has the form:

$$X = F(X) \tag{II-1}$$

where capital letters represent vectors or matrices. Starting with a set of initial guesses $X^o = (x_1^o, X_2^o, \ldots, x_n^o)^T$, all functions f_1^1, f_2^1, ..., f_n^1 are evaluated at x_i^o; i = 1, n. If a convergence criteria is reached, typically

$$| x_i^k - f_i^{k+1} | / | f_i^{k+1} | \leq \varepsilon$$

where ε is the desired accuracy, the solution has been found. Otherwise, x_i^0 replaced by f_i^1 and another iteration is performed. The recurrence relation is:

$$X^{k+1} = F^k(X^k) \qquad \text{(II-2)}$$

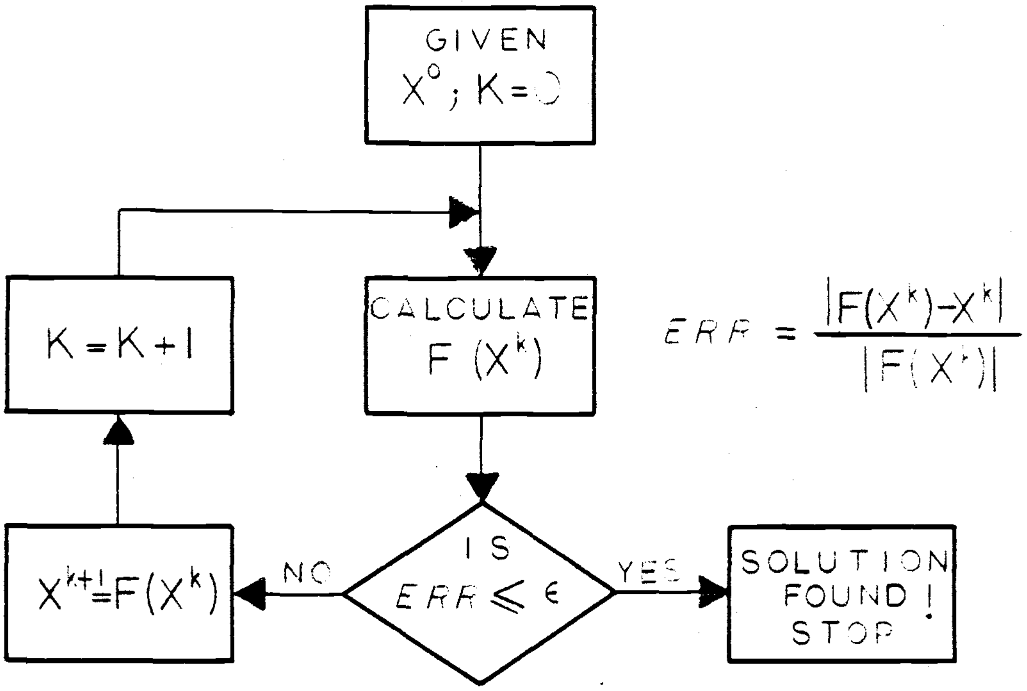
where the superscript k denotes the iteration number. Figure II-1 shows a flowchart of the method and a graphical representation of a one-dimensional case.

The method of successive substitution is quite effective when each function f_i is dominated by one variable x_i. Convergence, however, is sometimes very slow. The major advantage is its simplicity; its disadvantages are that if more than one variable strongly influence one or more of the functions, the method may not converge or convergence may be very, very slow.

<u>Wegstein's Method</u>

Wegstein's method is a modification of Aitken's Δ^2 method (Cavett, 1963). Wegstein's method, as the successive substitution method, is easy to implement.

Assuming that each function f_i, from the set of equations II-1, is a function of x_i alone, Wegstein's method extrapolates the new value of x_i along a straight line through two previous consecutive points. To initiate the method, two points are required to perform



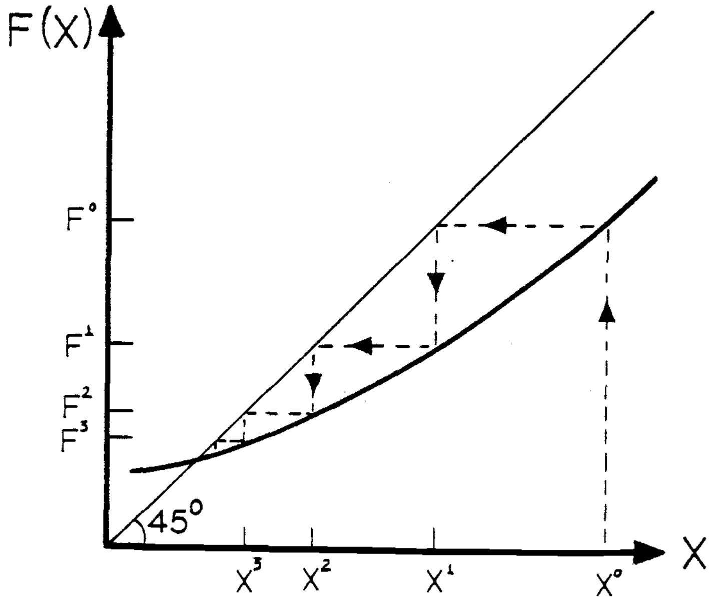


Figure II-1. Successive Substitution Method: Block Diagram and Graphical Representation

the first iteration. For a given set of initial values, $X^o = [\, x_1^o, x_2^o, \ldots, x_n^o \,]^T$ the easiest way to obtain the second set of points is by performing one successive substitution iteration. The recurrence relation for Wegstein's method is:

$$x_i^{k+1} = q_i^k*(f_i(X^k)) + (1 - q_i^k) * (x_i^k) \qquad \text{(II-3)}$$

where

$$q_i^k = 1./(1 - s_i^k) \qquad \text{(II-4)}$$

$$s_i^k = (f_i(X^k) - f_i(X^{k-1})/(x_i^k - x_i^{k-1}) \qquad \text{(II-5)}$$

$$i = 1, 2, \ldots, N$$

and

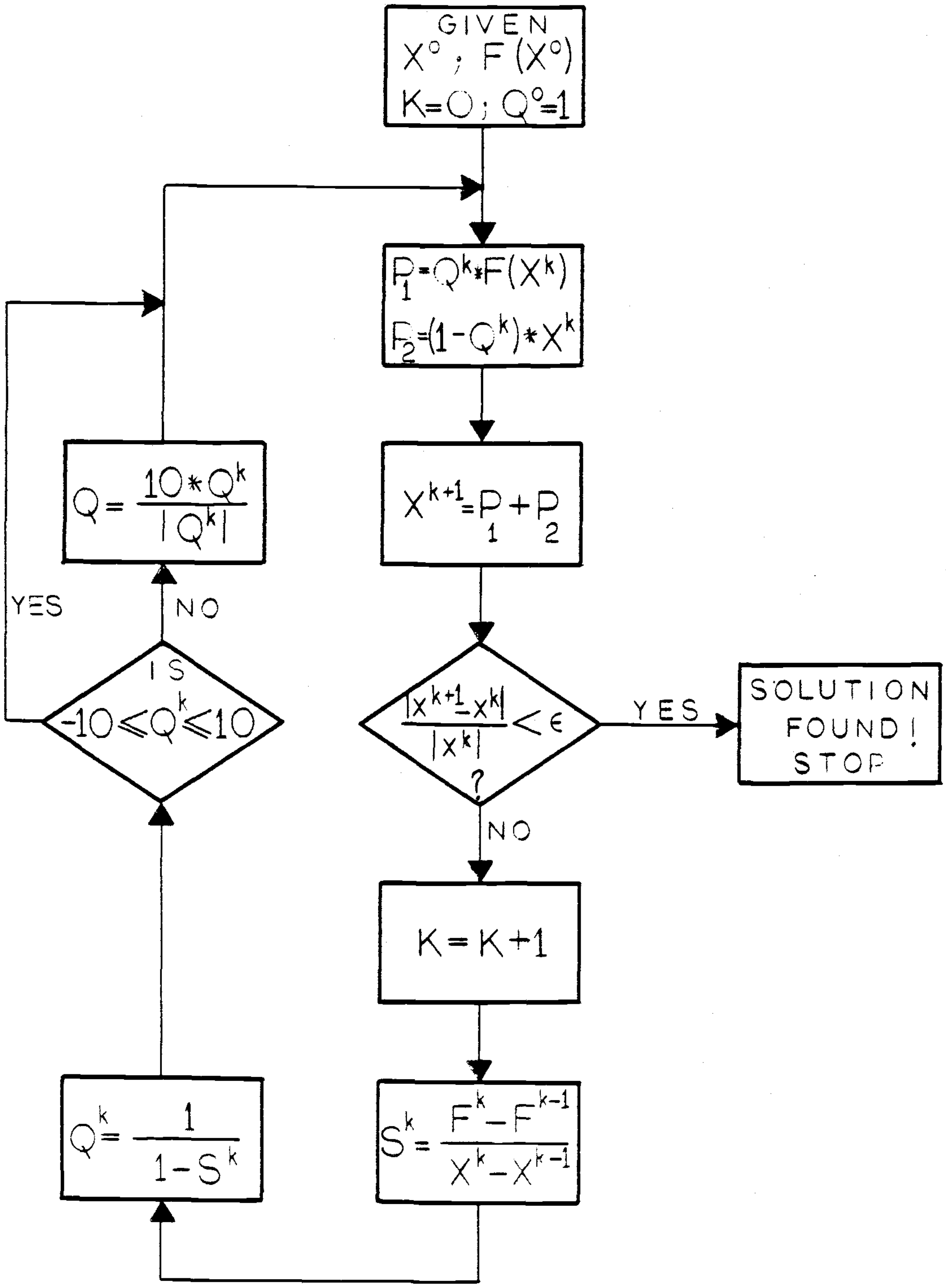
$$X_i^k = [\, x_1^k, x_2^k, \ldots, x_N^k \,]^T$$

$$F_i^k = [\, f_1(X^k), f_2(X^k), \ldots, f_N(X^k) \,]^T$$

Note that setting q_i equal to 1 is equivalent to performing one successive substitution iteration.

Figure II-2 shows a block diagram of the method and figure II-3 shows a graphical representation for a one-dimensional case.

Wegstein's method may converge quickly if each variable strongly influences a particular and unique function; otherwise the method will be very slow or it will not converge. In some cases, Wegstein's method may be very inefficient if the slope of the line



GIVEN X^0 ; $F(X^0)$ $K=0$; $Q^0=1$
$P_1 = Q^k * F(X^k)$
$P_2 = (1-Q^k) * X^k$
$X^{k+1} = P_1 + P_2$
$\frac{|X^{k+1} - X^k|}{|X^k|} < \epsilon$?
YES
SOLUTION FOUND! STOP
NO
$K = K+1$
$S^k = \frac{F^k - F^{k-1}}{X^k - X^{k-1}}$
$Q^k = \frac{1}{1-S^k}$
IS $-10 \leqslant Q^k \leqslant 10$
YES
NO
$Q = \frac{10 * Q^k}{|Q^k|}$

Figure II-2. The Wegstein Method: Block Diagram

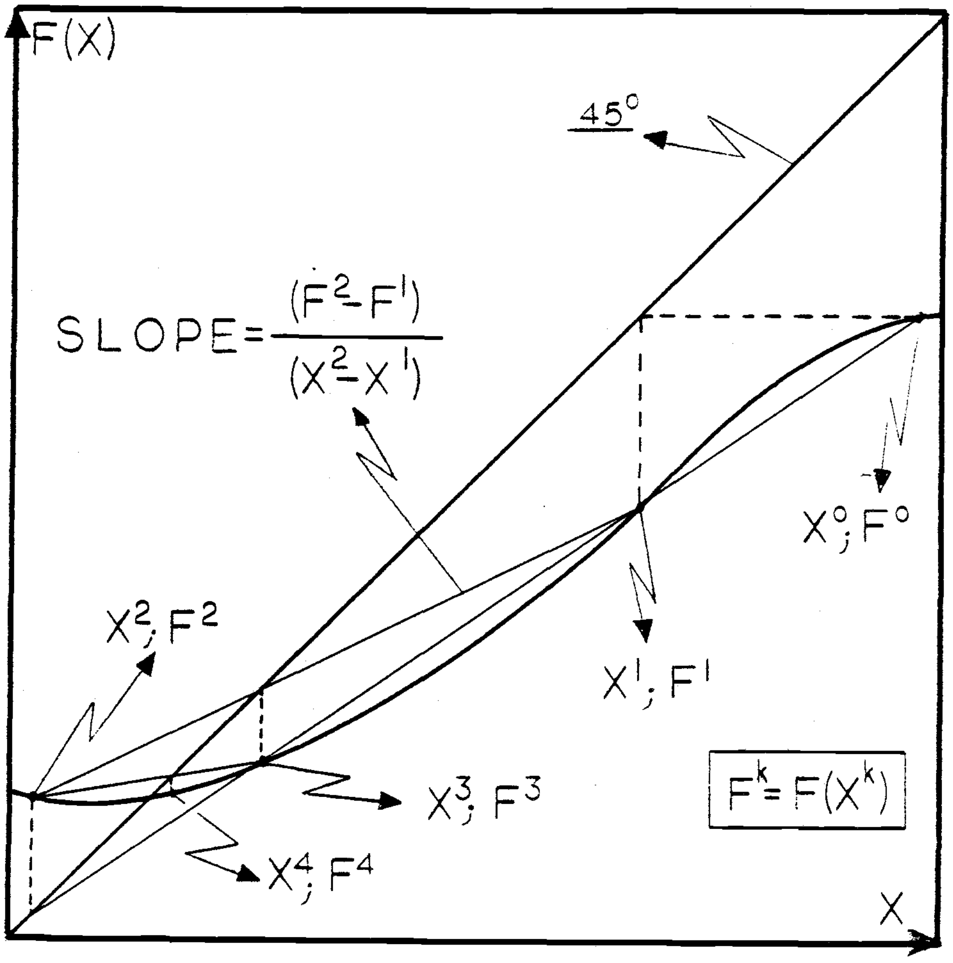


Figure II-3. The Wegstein Method: Graphical Representation

between two consecutive points approaches one. In other words, if S_i^k from equation II-5 approaches one then q_i^k approaches $\pm \infty$; in practice q_i^k is constrained, typical constraint values on q_i^k are $[-10, +10]$; $[-5, +5]$.

Some algorithms have the option of inserting one or more successive substitution iterations in between one or more of Wegstein's iterations. This "blend" of methods is quite effective in some cases.

A problem widely used to study the performance of nonlinear solvers is "Cavett's four flash drum." This hypothetical problem was idealized by Cavett (1963). We studied the performance of Wegstein's method with 0, 2, 4 and 8 successive substitutions in between each Wegstein iteration. The results are shown in figure II-4a and II-4b. For purposes of comparison, we also present the results of the successive substitution method.

From the figures, it is easy to see that both extremes, i.e., Wegstein's alone and successive substitution alone, are quite slow to converge. Moreover, while Wegstein's method has an erratic behavior, the successive substitution is quite constant in reducing the error and in converging. The effect of introducing some successive substitution iterations is extremely beneficial for this problem. The best performance was obtained with 4 successive iterations in between each Wegstein iteration.

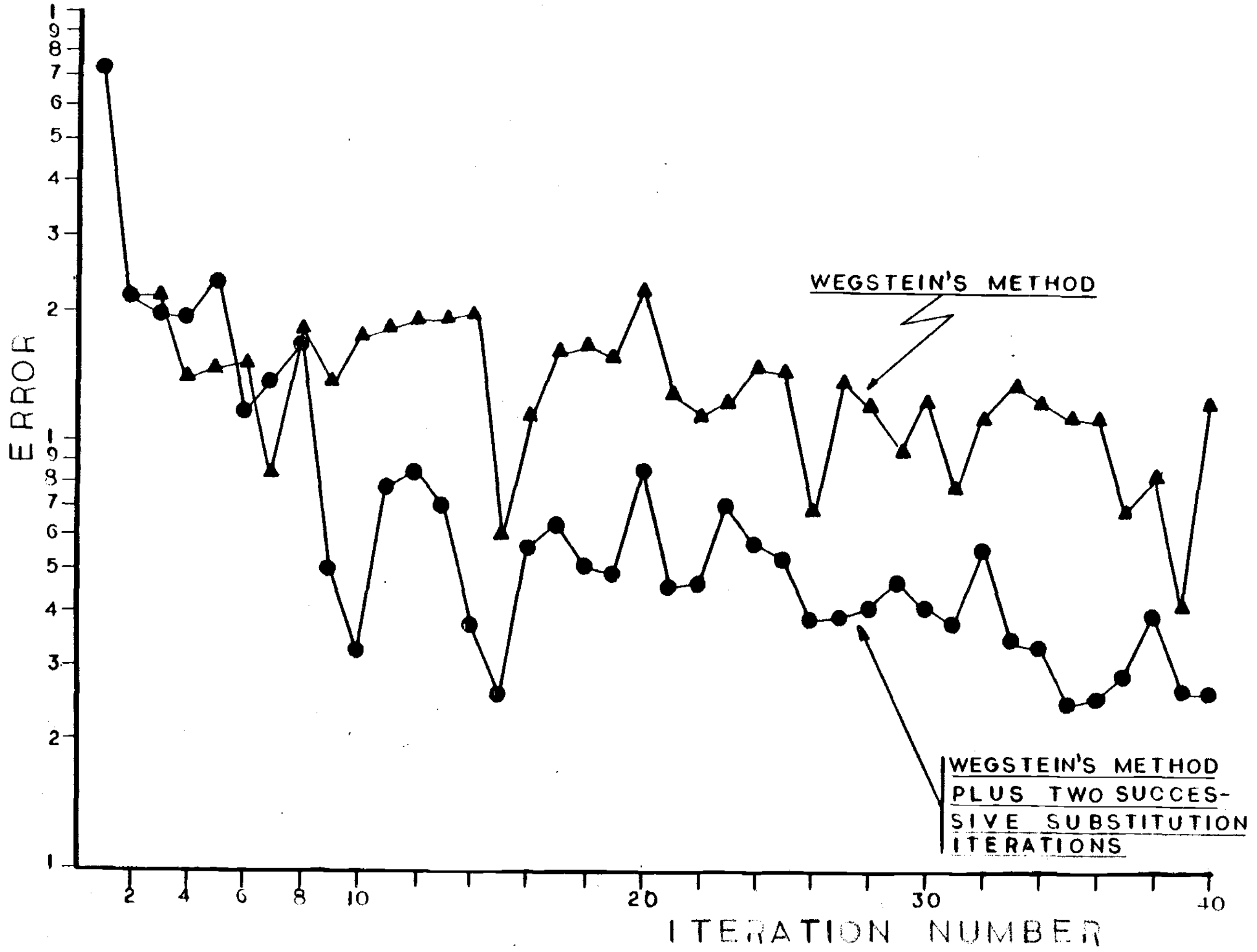


Figure II-4a. Results for Wegstein's Method with Cavett's Problem

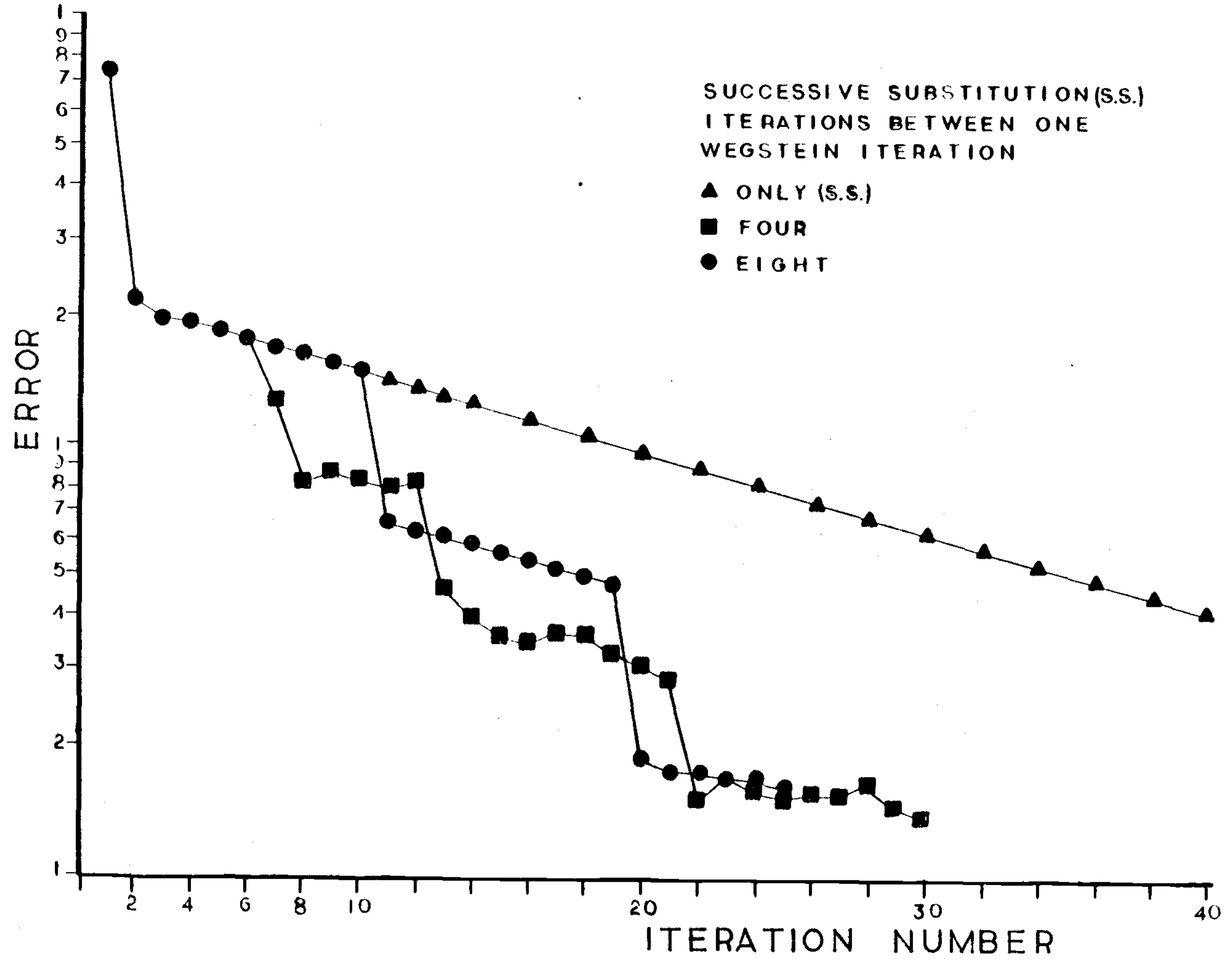


Figure II-4b. Results for Wegstein's Method with Cavett's Problem

Newton-Based Methods

The original method is known as the Newton-Raphson (N-R) method, and it is an iterative procedure based upon a Taylor series expansion terminated after the first derivative.

For a given set of N nonlinear equations,

$$F(X) = 0 \qquad \text{(II-6)}$$

it is desired to find the vector of unknowns $X^* = [x_1, x_2, \ldots, x_n]^T$ which simultaneously satisfies the equation set II-6. The Newton-Raphson method consists of the repeated use of the equation,

$$J^k P^k = -F(X^k) \qquad \text{(II-7)}$$

where the superscript k indicates iteration number, J^k is an N x N matrix of partial derivatives (the Jacobian matrix)

$$J^k = \begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{vmatrix}$$

P^k is an N-dimensional vector of correction steps,

$$P^k = X^{k+1} - X^k = [p_1, p_2, \ldots, p_n]^T$$

and $F(X^k)$ is an N-dimensional vector of function values evaluated at X^k,

$$F(X^k) = [f_1(X^k), f_2(X^k), \ldots, f_n(X^k)]^T$$

Equation II-7 is solved for P^k, and the predicted solution vector for the next iteration is given by:

$$X^{k+1} = X^k + P^k \qquad \text{(II-8)}$$

Convergence to a solution set is achieved when

$$\frac{|X^{k+1} - X^k|}{|X^{k+1}|} \leq \varepsilon$$

where ε is the desired accuracy of solution. Alternatively, we could use as the convergence criteria

$$\| F(X^k) \| \leq \varepsilon$$

where $\| \cdot \|$ denotes the Euclidian norm of $F(X^k)$

$$\| F(X^k) \| = [\sum_{i=1}^{N} f_i(X^k)^2]^{\frac{1}{2}}$$

A block diagram of the method is shown in figure II-5. It should be noted that at each iteration the Jacobian matrix must be

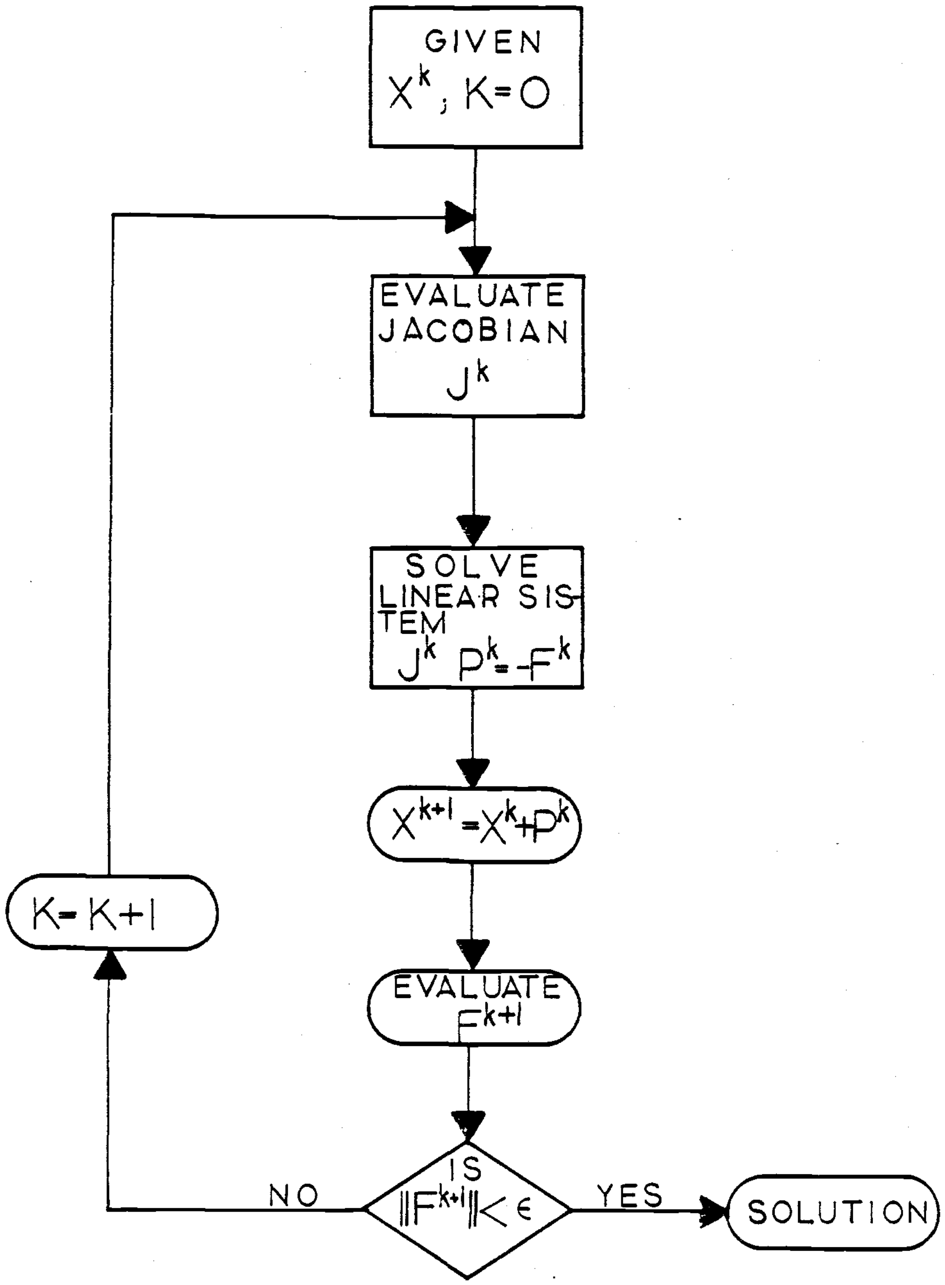


Figure II-5. The Newton-Raphson Method: Block Diagram

supplied. Also, the linear system II-7 is solved at every iteration.

When the functions are not analytical, and/or the Jacobian is difficult to obtain, we may calculate the Jacobian numerically, by forward differences,

$$J_{ij} = \left[\frac{\partial f_i}{\partial x_j} \right]_{X^k} \simeq \frac{f_i\,(X^k + h) - f_i(X^k)}{|h|} \qquad \text{(II-9)}$$

$i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, N$

h is an N-dimensional vector defined as follows,

$h_k = 0$ for $k \neq j$

$h_j =$ small positive number.

The calculation of the Jacobian by forward differences requires the evaluation of all N functions N + 1 times.

The functions that occur in chemical process simulations are, in general, "expensive" to evaluate. In the simultaneous modular approach, a series of equipment modules will be evaluated in each iteration to obtain function values. In addition, some modules, for example a flash drum, perform iterative solutions internally. Besides the computational effort, care must be taken to avoid "round-off" errors. For instance, if the convergence criteria used in the module is ε (say, 10^{-4}), perturbing a variable by less than ε will probably result in a poor evalution of the Jacobian matrix.

An alternative to Newton-Raphson's method is the use of Broyden's method (Broyden 1965).

Broyden's method is a modification of the Newton-Raphson method. The first iteration is essentially the same as in Newton-Raphson; in the second and subsequent iterations the Jacobian is not evaluated at X^k, but it is updated using Broyden's secant updated formula,

$$J^{k+1} = J^k + (Y^k - J^k P^k)(P^k)^T / (P^k)^T P^k \qquad \text{(II-10)}$$

where

$$Y^k = F(X^k + P^k) - F(X^k)$$

It is readily seen that no additional evaluations of functions are required, thus Broyden's algorithm is an attractive alternative to solving sets of nonlinear equations that occur in chemical process simulations. Actually, the use of the method in flowsheeting problems was suggested as early as 1966 (Rosen, 1966), and several other authors have either used the method or modifications of it in chemical process simulations (for example: Perkins (1979) and Mahalec et al. (1979)).

In general, Broyden's method will converge more slowly to a solution when compared with Newton-Raphson's. Dennis and Schnabel (1983), reported that at the solution, Broyden's update formula will generate a Jacobian with a relative error of approximately 1.1 percent when compared to the true Jacobian. Furthermore, the Jacobian as updated by Broyden's formula will have "fill-ins," that is, some elements of the true Jacobian that are equal to zero will assume values other than zero when updated by Broyden's formula. Moreover, Broyden's method, similar to Newton-Raphson's, may not converge to a solution if the initial estimates of the solution are far from the actual solution.

An alternative to Broyden's update formula is Schubert's update formula. The method was first proposed by Schubert (1970), and later by Broyden (1971). Schubert's formula does not update elements which are known constants. The formula is:

$$j_i^{k+1} = j_i^k + (y_i^k - j_i^k T_i^k)(T_i^k)^T / [(T_i^k)^T(T_i^k)] \qquad \text{(II-12)}$$

$$i = 1, 2, \ldots, N$$

where

$$Y_i^k = f_i(X^k + P^k) - f_i(X^k)$$

and

j_i^k = row vector containing the elements of the ith row of the Jacobian matrix J

f_i = ith element of $F(X^k)$

T_i^k = a column vector derived from P^k by setting to zero each element of P_i^k that corresponds to a known constant of j_i^k

In practice, T_i^k is set to zero only when an element of the Jacobian is zero. Any information about elements which are known constants but not equal to zero is disregarded.

When the Jacobian is sparse, Schubert's update formula is more attractive, since it does not create fill-ins and thus maintains the sparsity of the matrix. However, some authors (Mah and Lin (1980); Perkins and Sargent (1982)), reported unreliable results using the Schubert update formula in connection with the Newton-Raphson convergence algorithm.

As mentioned earlier, the Newton-Raphson method or nonlinear solver algorithms based on the N-R method (Broyden's or Schubert's methods) may diverge if the initial estimate of the solution is far from the actual solution. Because the N-R method is based on a local linearization of all functions by first order Taylor series expansions, the linearized functions are a good representation of the nonlinear functions when close to the solution, thus convergence is fast. Therefore, N-R based methods have good <u>local</u> convergence properties.

<u>A priori</u> we do not know the solution of a problem so there is no way to know how far from the solution the initial guessed values are. Although for chemical process simulation we can provide a good estimate of the solution-using very simple models or experience - the number of failures of N-R based methods may still be rather high.

In recent years N-R based methods have been used in conjunction with global methods for unconstrained optimization. The basic strategy of the global methods for unconstrained optimization is to solve the following problem:

$$\min R(X) = F(X)^T F(X) \tag{II-13}$$

Note that $R(X) = \| F(X) \|^2$. An iterative procedure would be to find correction steps P^k which, at each iteration, minimize the auxiliary function R(X) until a minimum is reached. If at that minimum R(X) = 0, the solution has been found. One of the drawbacks of the method is clear: R(X) may converge to a local minimum where the minimization problem is satisfied, but it is not a solution of the system of nonlinear equations. On the other hand, it can be proven that at a local minimum that is not the solution required,

$$J^T(X)\, F(X) = 0$$

Since $R(X) \neq 0$, thus, $F(X) \neq 0$, J(X) must be singular and since N-R based methods require the inverse of the Jacobian, they would also fail! Another point worth mentioning is that minimization methods have slow convergence properties, but, if no local nonzero minimum is reached, they will eventually converge to the solution of the equations, when it exists.

The Steepest Descent direction algorithm and Levenberg-Marquardt algorithm (Broyden, 1970), are two popular methods for the solution of systems of nonlinear equations based on the minimization of an objective function R(X); they both have good global convergence properties.

In practice, minimization algorithms are used to drive the initial estimate vector X^0 closer to the actual solution because these algorithms have good global convergence properties; then, the algorithm switches to N-R based methods, which have good local convergence properties.

A very popular algorithm with the properties mentioned in the last paragraph is due to Powell (1970). He proposed an hybrid algorithm based on the Levenberg-Marquardt method which showed excellent results. Later, Chen and Stadtherr (1981) proposed some modifications which improved Powell's method. In Appendix B there is a short review of Powell's algorithm and Chen and Stadtherr's modifications of Powell's method.

Implementation of MPDLM1 and MPDLM2

As part of this work, two subroutines were developed to solve systems of nonlinear equations. They both use Chen and Stadtherr's modification of Powell's method.

The need to have two versions of the same algorithm arises from the different characteristics of the Jacobian generated when solving sets of nonlinear equations using either the simultaneous modular approach or the equation-based approach. The first has a full Jacobian, only a few elements are equal to zero; the latter has a sparse Jacobian, only a few elements are different from zero.

To illustrate the characteristics of each approach, we will show the Jacobian generated by each in a typical simulation of an ammonia plant (figure II-6). In later chapters this problem will be presented in more detail. For now it is important to know that there are five components in each stream, five equipment modules (1 reactor, 1 mixer, 1 splitter and 2 flash drums), and the conversion of a reactant is given by the chemical equilibrium constant.

In the simultaneous modular approach there are 6 nonlinear equations being solved simultaneously: 5 stream connection equations and one constraint equation. The equations are:

$$f_i(X,\gamma_r) = x_i - y_i(X,\gamma_r) \qquad i = 1,5$$

$$f_6(X,\gamma_r) = K - Z(X,\gamma_r)$$

where

X	=	$[x_1,x_2,x_3,x_4,x_5]^T$
x_i	=	molar flow rate of the $i^{\underline{th}}$ component in the torn stream
y_i	=	molar flow rate of the $i^{\underline{th}}$ component in the torn stream obtained after each pass in the flowsheet. A pass in the flowsheet is defined as the sequential evaluation of all equipment modules. The initial and final point of the sequential evaluation is the torn stream(s)
γ_r	=	conversion of reactant r

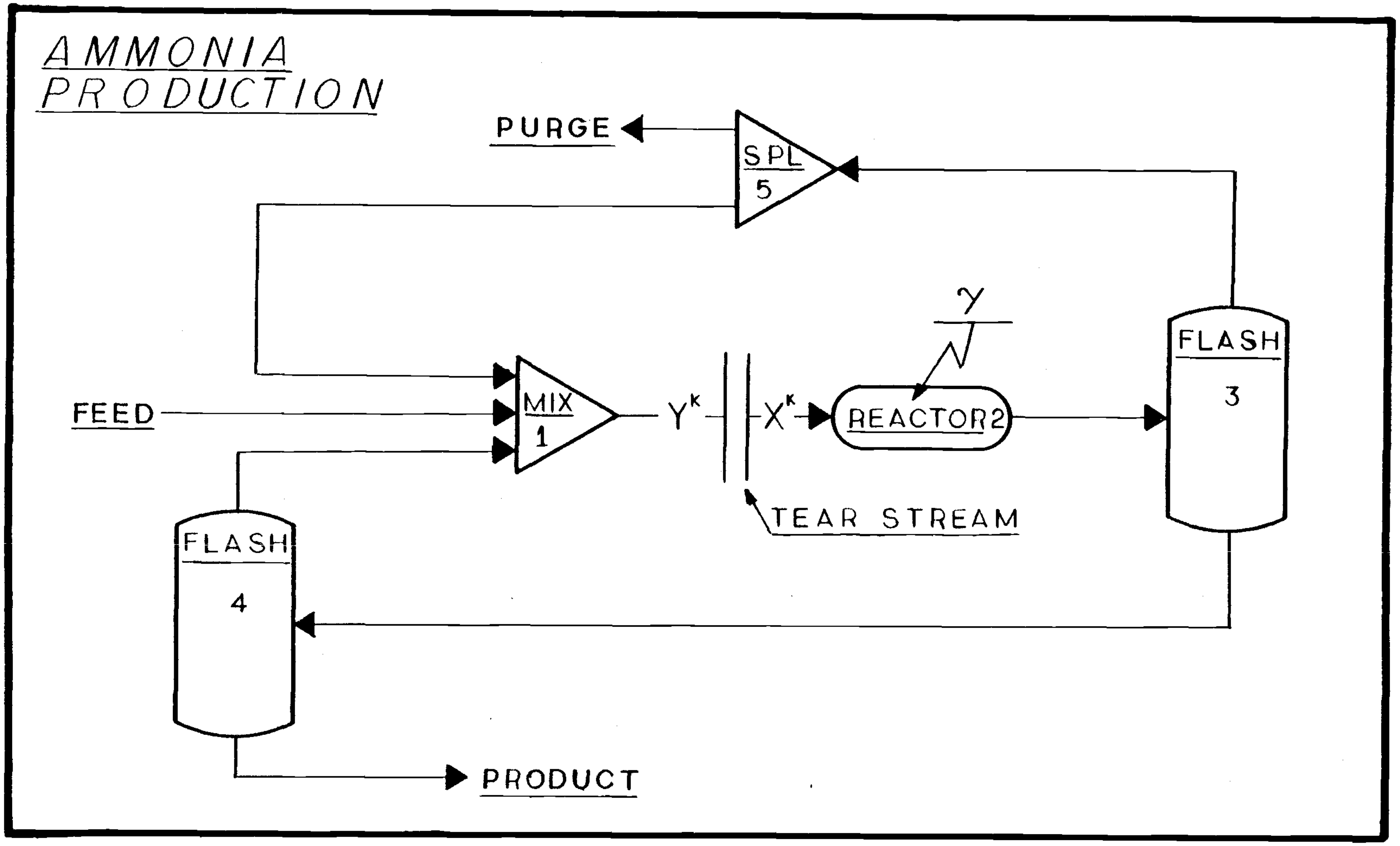


Figure II-6. Ammonia Plant Block Diagram

K = chemical equilibrium constant = 0.35

Z = a function of γ and x_i, $i = 1,5$

The Jacobian generated by the simulation of the NH_3 plant using the simultaneous modular approach is presented in figure II-7. It is clear that the Jacobian is nearly full.

To solve the same problem using the equation-based approach requires the solution of 50 simultaneous equations. The equations represent material balances around each piece of equipment. The Jacobian generated by the equation-based approach (figure II-8) had 182 nonzero elements. The full Jacobian has 50 x 50 = 2500 elements, so only 7.3 percent of the elements will be used throughout the nonlinear solver subroutines.

Knowing the characteristics of each approach, it was decided that Broyden's update formula would be more attractive for the simultaneous modular approach. For the equation based approach, Schubert's update formula was chosen. In addition, that subroutine uses sparse matrix techniques which enables it to store and operate on the nonzero elements only.

Subroutine MPDLM1 is used with the simultaneous modular approach (Broyden's update formula) and MPDLM2 with the equation-based approach (Schubert's update formula).

Following the suggestion of several authors (Powell, 1970; Chen and Stadtherr, 1981; Dennis and Schnabel, 1983), the Jacobian and function values are scaled. A scaling matrix DF is calculated so that the resulting scaled Jacobian (DF)(J) has the largest elements

VARIABLES

FUNCTIONS	X_1	X_2	X_3	X_4	X_5	γ
1	●					●
2	●	●	●			●
3	●	●	●	●	●	●
4	●	●	●	●	●	●
5	●	●	●	●	●	●
6	●	●	●	●	●	●

Figure II-7. Typical Jacobian Generated When Solving Nonlinear Equations Arising from the Simultaneous Modular Approach

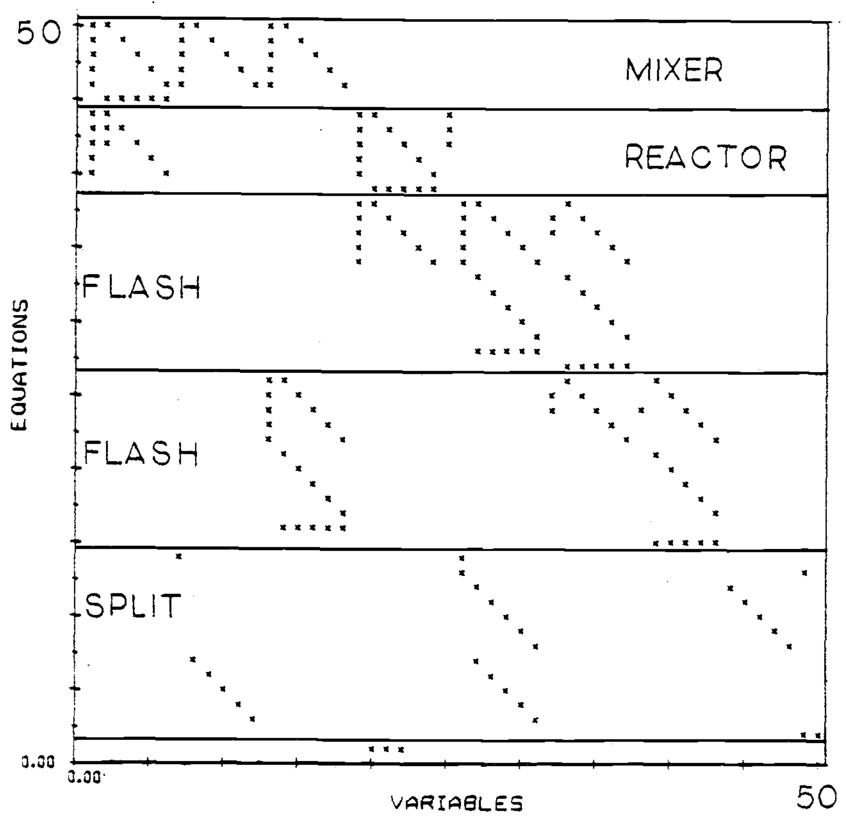


Figure II-8. Typical Jacobian Generated When Solving Nonlinear Equations Arising from the Equation-Based Approach

in each row equal to ±1. The scaling of the Jacobian and functions was suggested to improve the accuracy and convergence of nonlinear solvers.

Sometimes the rate of convergence begins to slow considerably. This may happen because the Jacobian update by Schubert's or Broyden's formula is not a good approximation of the true Jacobian. In this case the Jacobian is re-evaluated by forward differences. Early tests with both subroutines showed that the Jacobian updated by secant formulas "degrades" after 15-20 iterations if the initial estimate of the solution is not good. To decide when to re-evaluate the Jacobian, we used the same procedure suggested by Chen and Stadtherr (1981). It consists of the following steps:

1. After each Jacobian evaluation by forward differences, set IFLAG = 0.
2. After each iteration k, if

   $\| F(X^k + P^k) \|^2 \geq 0.999 \| F(X^k) \|^2$

   set

   IFLAG = IFLAG + 1

   Otherwise, IFLAG = IFLAG - 1

   If IFLAG < 0, set IFLAG = 0
3. Evaluate

   $R_1 = \| F(X^k + P^k) \|^2 / \| F(X^{k-4}) \|^2$

   $R_2 = \| F(X^{k-4}) \| / \| F(X^{k-9}) \|$

The Jacobian is re-evaluated if:

a. Since last Jacobian evaluation

$\| F(X^k + P^k) \|$ has been reduced by a factor of 2

<u>and</u>

b. IFLAG > 3 or $R_1 > R_2$

The parameters IFLAG, R_1, and R_2 can be interpreted as a "measure" of the progress towards the solution. R_1 "traces" the progress of the last iterations and R_2 the progress of the 5 iterations before those. If IFLAG is greater than 3, or R_1 is greater than R_2, we are ensuring that the converge is quite slow before the Jacobian is re-evaluated. The first condition is used to ensure that we do not re-evaluate the Jacobian too often.

The parameter IFLAG is also used to check when the algorithm reached a local minimum, or when the convergence is too slow. If IFLAG is greater than MAX (10, N+4), where N is the number of equations, the subroutine stops and an error message is issued.

Besides the update formula, another difference between MPDLM1 and MPDLM2 is the method used to solve the linear system. MPDLM1 uses an LU decomposition of the Jacobian matrix. The LU factorization,

$$J = LU$$

where L is a lower triangular matrix and U an upper triangular matrix, is performed every time the Jacobian is calculated by forward differences. In subsequent iterations the L and U factors are directly updated using Broyden's formula and Bennett's algorithm (Bennett, 1965).

The solution procedure for the set of linear equations is

$$J^k P^k = -F(X^k)$$

or

$$L^k U^k P^k = -F(X^k)$$

Define $U^k P^k = Y$ and solve

$$L^k Y = -F(X^k)$$

for Y.

Then solve

$$U^k P^k = Y$$

for P^k.

For MPDLM2 we used subroutine SPAMAT (Rodrigues, 1979). This subroutine solves the linear system by a Gaussian elimination, and it uses sparse matrix techniques to store elements and perform operations. The sparse matrix technique consists in storing only nonzero elements in an N x M matrix B. The row positions of each nonzero element are stored in another N x M integer matrix IC, and the number of nonzero elements in each row is stored in an N-dimensional vector IZ. For example, assume that a Jacobian matrix 4 x 4 has the following configuration:

$$J = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

Using the sparse technique just described we would have:

$$B(4x2) = \begin{bmatrix} 1 & 2 \\ 3 & 0 \\ 4 & 1 \\ 5 & 0 \end{bmatrix} \qquad IC(4x2) = \begin{bmatrix} 1 & 3 \\ 2 & 0 \\ 3 & 4 \\ 5 & 0 \end{bmatrix} \qquad IZ(4) = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

The drawback of this fomulation is that we must know beforehand the maximum number of elements in one single row in order to set the second dimension of matrices B and IC.

The small size example shown above is for clarity. There will be an economy in storage requirements if,

$$2(m * n) + n \leq n^2$$

or

$$(2m + 1) \leq n$$

where m is the maximum number of nonzero elements in one single row. For example, assume that the maximum number of elements in a single row of a 200 x 200 matrix is 15. The storage requirements would be 200 * 200 = 40,000 for the full matrix. Using the sparse matrix technique just described, only 2 * 200 * 15 + 200 = 6200 storage positions would be required, or 15.5 percent of the full matrix storage requirements.

This is not the best sparse matrix technique, but, for the purpose of this work, it performed quite well. An excellent review of sparse matrix techniques associated with the solution of linear systems may be found in Duff (1977).

Thus far the most important features of MPDLM1 and MPDLM2 have been discussed. Figure II-9 shows a block diagram of the algorithm. It does not include all details, however, Appendix C contains a printout of the source code for both subroutines.

Performance of MPDLM1 and MPDLM2

Both subroutines were compared with subroutine ZSPOW from IMSL. We solved 5 problems, four of them from the open literature and often used to compare the performance of nonlinear solvers. Some results from Chen and Stadtherr (1981) and Powell (1970) are available, so they are reproduced for comparison purposes. The fifth problem is the simulation of a system of 3 counter-current evaporators for Kraft black-liquor.

Problem 1 - Brown's Almost Linear Function (PBALF)

$$F_i = x_i - 11 + \sum_{j=1}^{10} x_i = 0 \qquad i = 1,9$$

$$F_{10} = 1 - \prod_{j=1}^{10} x_j = 0$$

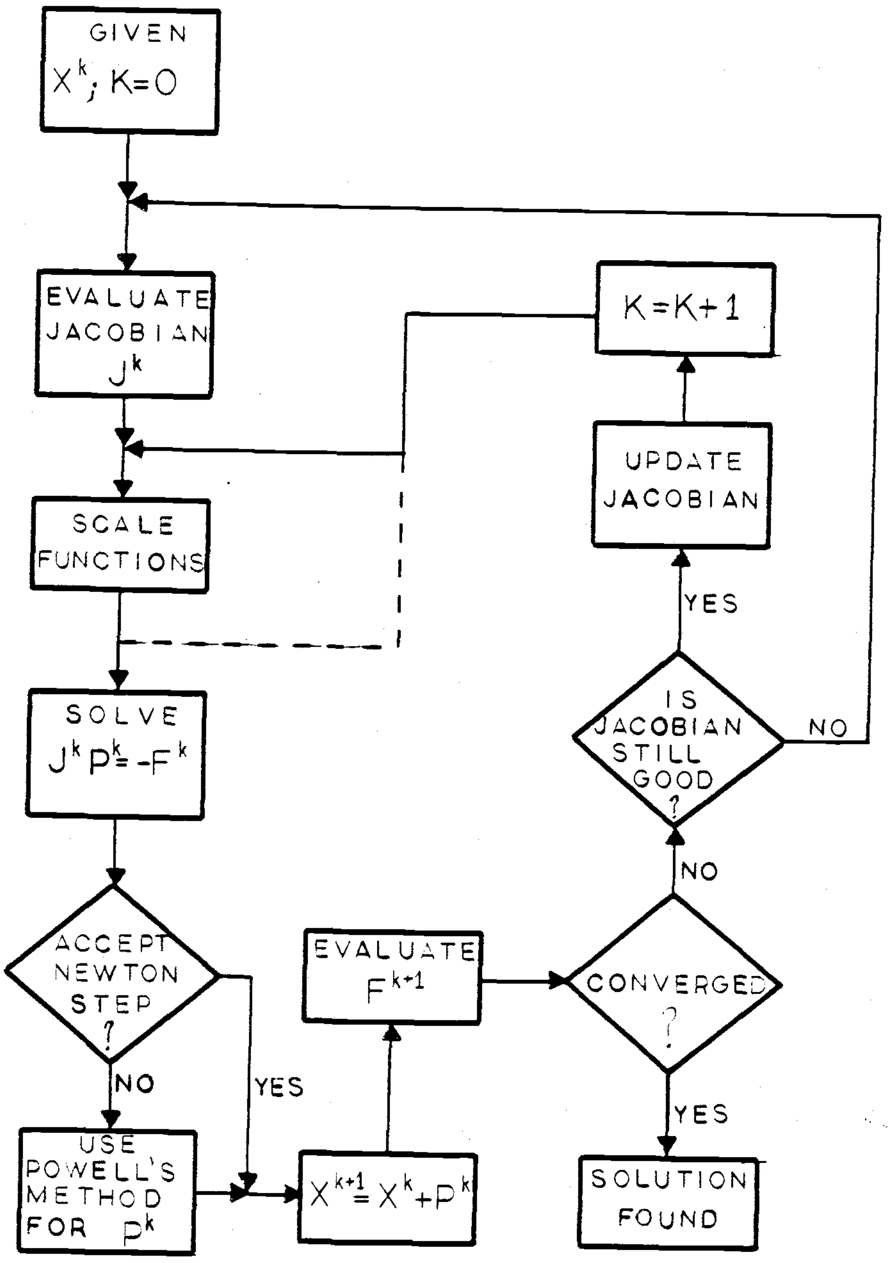


Figure II-9. Block Diagram of the Modified Newton-Raphson Method Used in This Work

initial point: $x_i = 0.5;\ i = 1,10$

solution: $x_i = 1;\ i = 1,10$

Problem 2 - Powell's Badly Scaled Function (PPBSF)

$$F_1 = 10{,}000 * x_1 * x_2 - 1 = 0$$

$$F_2 = \text{EXP}(-x_1) + \text{EXP}(-x_2) - 1.0001 = 0$$

initial point: $x_1 = 0;\ x_2 = 1$

solution: $x_1 = .1098 * 10^{-4};\ x_2 = 9.106147$

Problem 3 - Rosenbrock's "Banana" Function (PRBF)

$$F_1 = 10.0 * (x_2 - x_1 * x_1) = 0$$

$$F_2 = 1.0 - x_1 = 0$$

initial point: $x_1 = -1.2;\ x_2 = 1.0$

solution: $x_1 = 1.0;\ x_2 = 1.0$

Problem 4 - Powell's Singular Function (TET)

$$F_1 = x_1 + 10 * x_2 = 0$$

$$F_2 = \sqrt{5} * (x_3 - x_4) = 0$$

$$F_3 = (x_2 - 2 * x_3)^2 = 0$$

$$F_4 = \sqrt{10} * (x_1 - x_4)^2 = 0$$

initial point: $X = [3,-1,0,1]^T$

solution: $X = [0,0,0,0]^T$

Problem 5 - Kraft Black-Liquor Evaporator (KBLE)

The flow diagram is shown in figure II-10, initial values and solution are shown in table II-2.

The number of function evaluations, that is, the number of times the subroutine containing the nonlinear system was called, including evaluations used to obtain the Jacobian, are summarized in table II-1.

Table II-1. Performance of Nonlinear Solvers (Number of Function Evaluations)

	Problem				
Subroutine	PBALF	PPBSF	PRBF	TET	KBLE[2/]
MPDLM1	26	49	7	26	--
MPDLM2	37	26	26	91	11.619
ZSPOW	31	22	181	110[1/]	18.651
Chen Stadtherr	26	40	7	47	--
POWELL	--	223	28	--	--

1/ failed to solve

2/ only execution time available (CPU-sec)

In general, the results obtained with MPDLM1, are superior, in terms of iterations required, when compared with the results of ZSPOW. In addition they are similar to those obtained by Chen and Stadtherr, as expected since the methods are the same. However,

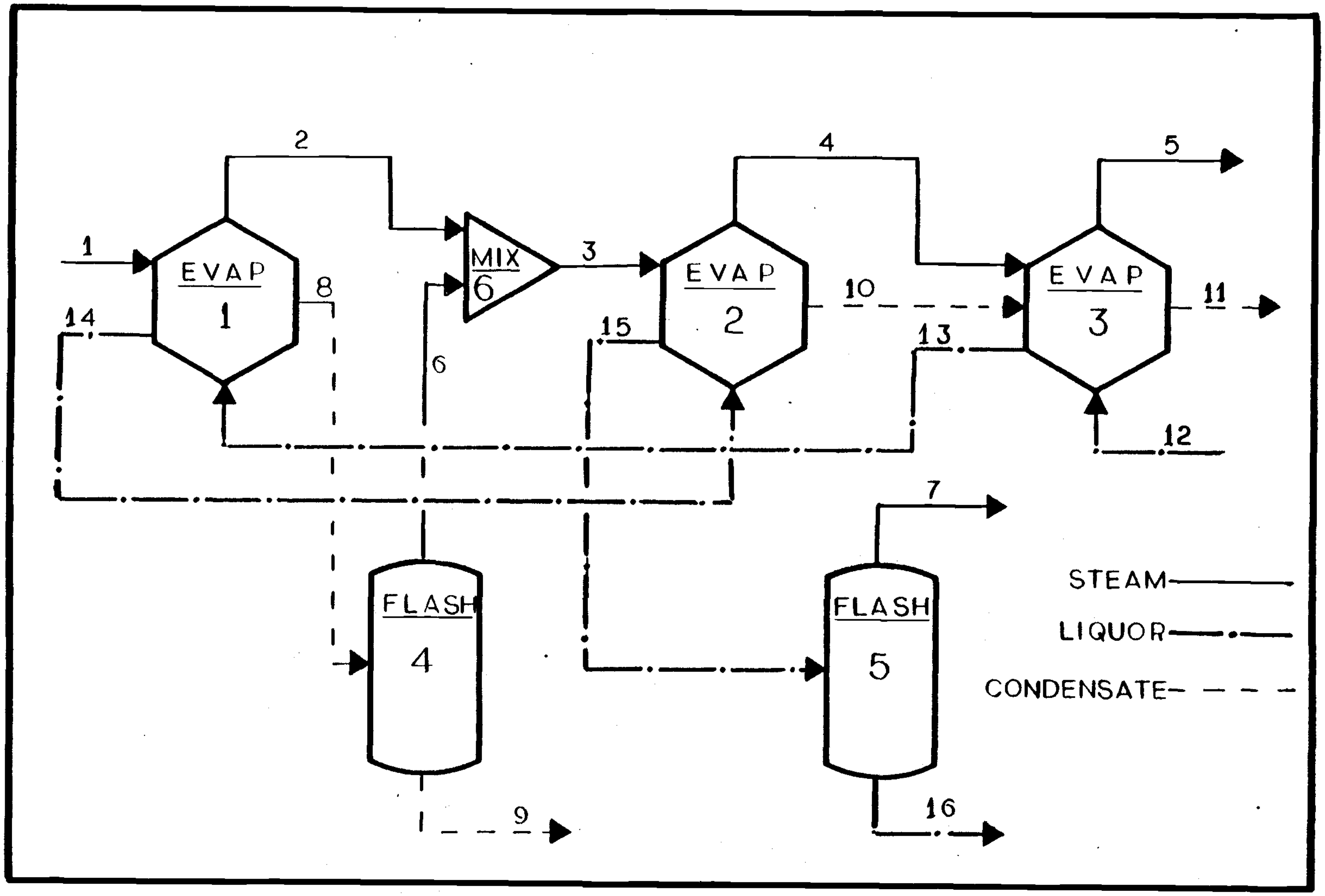


Figure II-10. Block Diagram of the 3 Effect Evaporator Problem

Table II-2. Initial Values and Solution of the 3 Effect Evaporator Problem

```
--------------------------------------------------------------------------------------

   NUMBER OF VARIABLES=54

--------------------------------------------------------------------------------------

    THE SYSTEM TO BE CALCULATED,WITH INITIAL GUESSES AND
      KNOWN VARIABLES WILL BE PRINTED NOW

--------------------------------------------------------------------------------------

STREAM #  FLOW  RATE        PRESSURE   TEMPERATURE   SOLID CONTENT        ENTHALPY
           (LB/H)            (PSIA)     ( DEG F)       (LB/LB)            (BTU/LB)

    1    100000.0000        50.0000     282.0000        .0000            1000.0000
    2    100000.0000        20.0000     200.0000        .0000            1000.0000
    3    100000.0000        20.0000     200.0000        .0000            1000.0000
    4    100000.0000        10.0000     180.0000        .0000            1000.0000
    5    100000.0000         5.0000     140.0000        .0000            1000.0000
    6     10000.0000        20.0000     200.0000        .0000            1000.0000
    7      5000.0000         5.0000     215.0000        .0000            1000.0000
    8    100000.0000          .0000     281.0020        .0000             200.0000
    9     90000.0000          .0000     200.0000        .0000             200.0000
   10    100000.0000          .0000     227.9559        .0000             200.0000
   11    200000.0000          .0000     193.2139        .0000             200.0000
   12    400000.0000          .0000     180.0000        .1500             143.4109
   13    300000.0000          .0000     180.0000        .2000             140.1515
   14    200000.0000          .0000     180.0000        .3000             134.0580
   15    100000.0000          .0000     180.0000        .4000             128.4722
   16     94000.0000          .0000     180.0000        .5000             123.3333

--------------------------------------------------------------------------------------

   IF YOU WANT TO STOP THE PROGRAM AND CHANGE INITIAL
   GUESSES ENTER 100 (AS ? APPEARS)
? 2

--------------------------------------------------------------------------------------

   THE FOLLOWING VALUES OF Q AND UA WILL BE USED

--------------------------------------------------------------------------------------
            BTU/H•F                 BTU/Lb
   UA(1)=  3000000.0000          Q(1)=100000000.0000
   UA(2)=  4000000.0000          Q(2)=100000000.0000
   UA(3)=  4000000.0000          Q(3)=100000000.0000
--------------------------------------------------------------------------------------

   THE CALCULATED VALUES FOR THE SYSTEM OF EVAPORATORS    ARE:

STREAM #  FLOW  RATE        PRESSURE   TEMPERATURE   SOLID CONTENT        ENTHALPY
           (LB/H)            (PSIA)     ( DEG F)       (LB/LB)            (BTU/LB)

    1    117422.7768        50.0000     281.0020        .0000            1174.0713
    2     92268.1028        24.4540     244.7940        .0000            1163.1579
    3     97461.0763        24.4540     244.4758        .0000            1162.9958
    4    101511.8165         9.7773     215.5471        .0000            1154.1407
    5    111597.5336         5.0000     165.7888        .0000            1132.7387
    6      5192.9735        24.4540     238.8440        .0000            1160.1163
    7      2142.8393         5.0000     187.1811        .0000            1142.7860
    8    117422.7768          .0000     281.0020        .0000             249.0020
    9    112229.8032          .0000     238.8440        .0000             206.8440
   10     97461.0763          .0000     238.8440        .0000             206.8440
   11    198972.8929          .0000     192.1519        .0000             160.1519
   12    400000.0000          .0000     180.0000        .1500             143.4109
   13    288402.4664          .0000     165.7888        .2080             126.2325
   14    196134.3636          .0000     244.7940        .3059             192.2539
   15     94622.5470          .0000     215.5471        .6341             145.1691
   16     92479.7077          .0000     187.1811        .6488             122.0534

--------------------------------------------------------------------------------------

   UA(1)=  3000000.0000          Q(1)=108624204.8010
   UA(2)=  4000000.0000          Q(2)= 93187588.3397
   UA(3)=  4000000.0000          Q(3)=105452264.5423

--------------------------------------------------------------------------------------
```

they are not identical and two possibilities exist for that disagreement; one is that a user given parameter, the "distance" between the initial guess and the actual solution, was not the same. That parameter plays an important role in the number of iterations required. The other possibility is an error in our coding of Chen and Stadtherr's algorithm.

The results of Powell with the problems PRBF and PPBSF, when compared with MPDLM1, confirm the improvement of Powell's algorithm by Chen and Stadtherr.

MPDLM2, using Schubert's update formula and sparse matrix techniques, had inferior performance when compared to MPDLM1. The reason for that is probably the simplification introduced in Schubert's update formula, i.e., treating only zeros as constants, instead of all known constants in the Jacobian. On the other hand, MPDLM2 solved problem TET, in which ZSPOW failed. In addition, MPDLM2 solved the evaporator problem in 60 percent of the execution time required by ZSPOW, which demonstrates the benefit of using sparse matrix techniques.

CHAPTER III

IMPLEMENTATION OF SIMO AND EQSS

In this chapter we will describe the implementation of SIMO (Simultaneous Modular Library) and EQSS (Equation Solving Simulation Library). A listing of the source code for both libraries is shown in Appendix C. It should be noted that both libraries use the same nomenclature for the flowsheet variables:

FLOW(I)	=	Molar flow rate of the I$\underline{\text{th}}$ stream (mol/unit of time)
COMP(I,L)	=	Mol fraction of L$\underline{\text{th}}$ component in the I$\underline{\text{th}}$ stream
EQP(I,J)	=	J$\underline{\text{th}}$ equipment parameter of the I$\underline{\text{th}}$ module

Because of this, the modules from SIMO can be used to initialize varibles in EQSS, and the same input/output subroutines can be used for both approaches.

SIMO Library

This library is used to perform chemical process simulations using either the sequential modular approach or the simultaneous modular approach. There are 26 subroutines available, which can be organized in three major categories:

1. Nonlinear Equations Solvers
2. Equipment Modules

3. Support Subroutines

Subroutines To Solve Nonlinear Equations

There are seven nonlinear solver subroutines; five are one-dimensional (one equation in one unknown) and two are multi-dimensional (N equations in N unknowns). Each subroutine uses a different method of solution, so the user has the option of using the best method to solve specific problems. In addition to the seven nonlinear solvers, there is one subroutine to perform an LU (L lower triangular, U upper triangular) factorization of a general N x N matrix.

We will now proceed with a brief description of each subroutine. The nomenclature of several parameters common to some of the subroutines are:

NT	-	Total number of iterations allowed.
EPS	-	Desired accuracy (typical values 10^{-3}, 10^{-4}).
X	-	An N-vector of initial guesses which are specified as input by the user; on output X carries the best estimate of the solutions.
F	-	An N-vector of function values (Input/Output).
N	-	Number of equations.
K	-	User provided parameter to control the printing of iterations results; every $k^{\underline{th}}$ iteration the results are printed.

<u>SUBROUTINE NEWTON</u> (X, NT, EPS, SFNC, K) - Solves one nonlinear equation, F(X) = 0, in one unknown using Newton's method. The user must provide SUBROUTINE SFNC (X,F,FD), which calculates function values and derivative values at X (Kayihan, 1979).

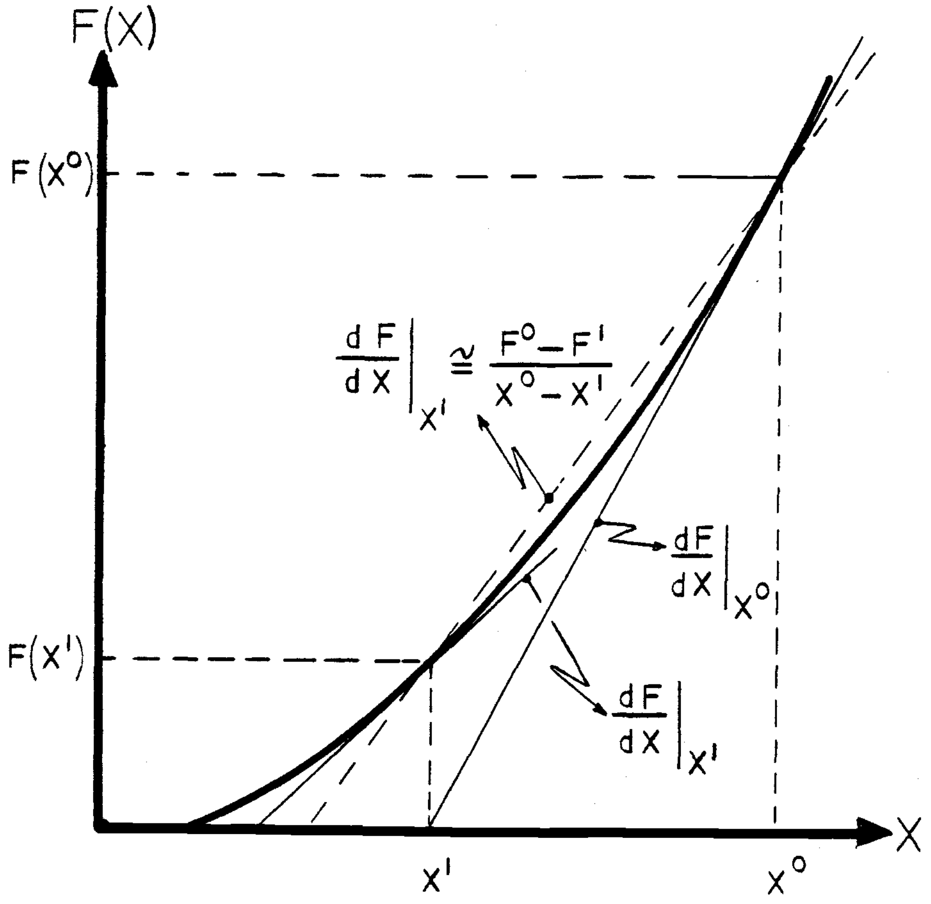
<u>SUBROUTINE INTHLV</u> (XL, XR, X, NT, FNC, K) - Solves one nonlinear equation, F(X) = 0, in one unknown using the interval-halving (half-interval) method. The user must specify left-hand (XL) and right-hand (XR) bounds on the root. In addition the user must provide FUNCTION FCN(X), which calculates function values at X (Kayihan, 1979).

<u>SUBROUTINE SUCSUB</u> (X, NT, EPS, FNC, K) - Solves one nonlinear equation, X = F(X), in one unknown using the successive substitution method. The user should provide FUNCTION FNC(X) which calculates the value of F(X) at X (Kayihan, 1979).

<u>SUBROUTINE WEGSTN</u> (X, NT, EPS, FNC, K) - Solves one nonlinear equation, X = F(X), in one unknown using Wegstein's method. The user provides FUNCTION FNC(X) which calculates the value of F(X) at X (Kayihan, 1979).

<u>SUBROUTINE WEGSMD</u> (N, X, NT, EPS, SUB, K) - Solves "N" nonlinear equations, X = F(X), in "N" unknowns using Wegstein's method. The user must provide SUBROUTINE SUB(N, F, X) which calculates the value of F(X) at X (Kayihan, 1979).

<u>SUBROUTINE SECNEW</u> (X, NT, EPS, SUB) - Solves one nonlinear equation, F(X) = 0, in one unknown using the secant method. The first iteration is a Newton iteration, the following are secant approximations. Figure III-1 has a graphical representation of the



$F(X)$
$F(X^0)$
$F(X^1)$
$\left.\frac{dF}{dX}\right|_{X^1} \cong \frac{F^0 - F^1}{X^0 - X^1}$
$\left.\frac{dF}{dX}\right|_{X^0}$
$\left.\frac{dF}{dX}\right|_{X^1}$
X^1
X^0
X

Figure III-1. Graphical Representation of the Method Used in SECNEW to Solve Nonlinear Equations

method. The user must provide SUBROUTINE SUB (X,F) which calculates function values of F(X) at X.

<u>SUBROUTINE MPDLM</u> (FCN, X, F, N, B, NT, IDGT, MS) - Solves N nonlinear equations, F(X) = 0, in N unknowns using Chen and Stadtherr modification of Powell dogleg method (see Chapter II and Appendix B). FCN is a user-written subroutine to calculate the values of the functions at X. B is a matrix with dimensions N x 32. IDGT is the number of digits of accuracy required. MS is a user-specified parameter defined as follows:

MS = 0; no scaling of Jacobian and functions

MS = 2; auto-scaling

MS = 3; auto-scaling and the Jacobian of the system of equations is given externally through a user given subroutine JACOBI (X, F, N, B).

<u>SUBROUTINE FACLU</u> (A, IP, N) - Factorization of the N x N matrix A into a product of a lower triangular matrix (L) and an upper triangular matrix (U). L has unit diagonal elements which <u>are not</u> stored. <u>A</u> is stored columnwise, in a vector of dimension N^2. IP, on output, contains a permutation vector of A (from "IBM - Programmer's Manual," 1968).

Equipment Modules

There was no need to develop new equipment modules for the simultaneous modular approach because the source code of SIMFLOWS (Kayihan, 1979) was available. A few modifications were introduced into SIMFLOWS in order to simulate more complex problems. The number of components allowed was increased from 7 to 19. The number of equipment modules and streams allowed was increased from 20 to 30. In addition, the separator module was modified to allow three output streams instead of two output streams as it was originally developed.

The subroutines and their graphical representation can be found in figure III-2. The details of the modules are shown in Appendix A.

Support Subroutines

Support subroutines are used for input, output and to prepare the system of nonlinear equations for solution. For either approach, sequential or simultaneous modular, the user must provide one subroutine with the equipment modules describing the process in its sequential order of calculation. When using the simultaneous modular approach the subroutine must be named FSIS. The support subroutines are:

SUBROUTINE READ (NST, NEQ) - A subroutine to read "NST" stream variables and "NEQ" equipment parameters. The user provides in a data file all known stream variables, equipment parameters, as well as the initial guess(es) of the torn stream(s). A sample of a data file is presented in figure III-3 (Kayihan, 1979).

SUBROUTINE CHECKS (N1) - A subroutine to check the consistency of stream variables of stream N1. If a flow rate is lower or equal to zero, or the mole fractions do not add up to 1, an error message is issued and the program stops (Kayihan, 1979).

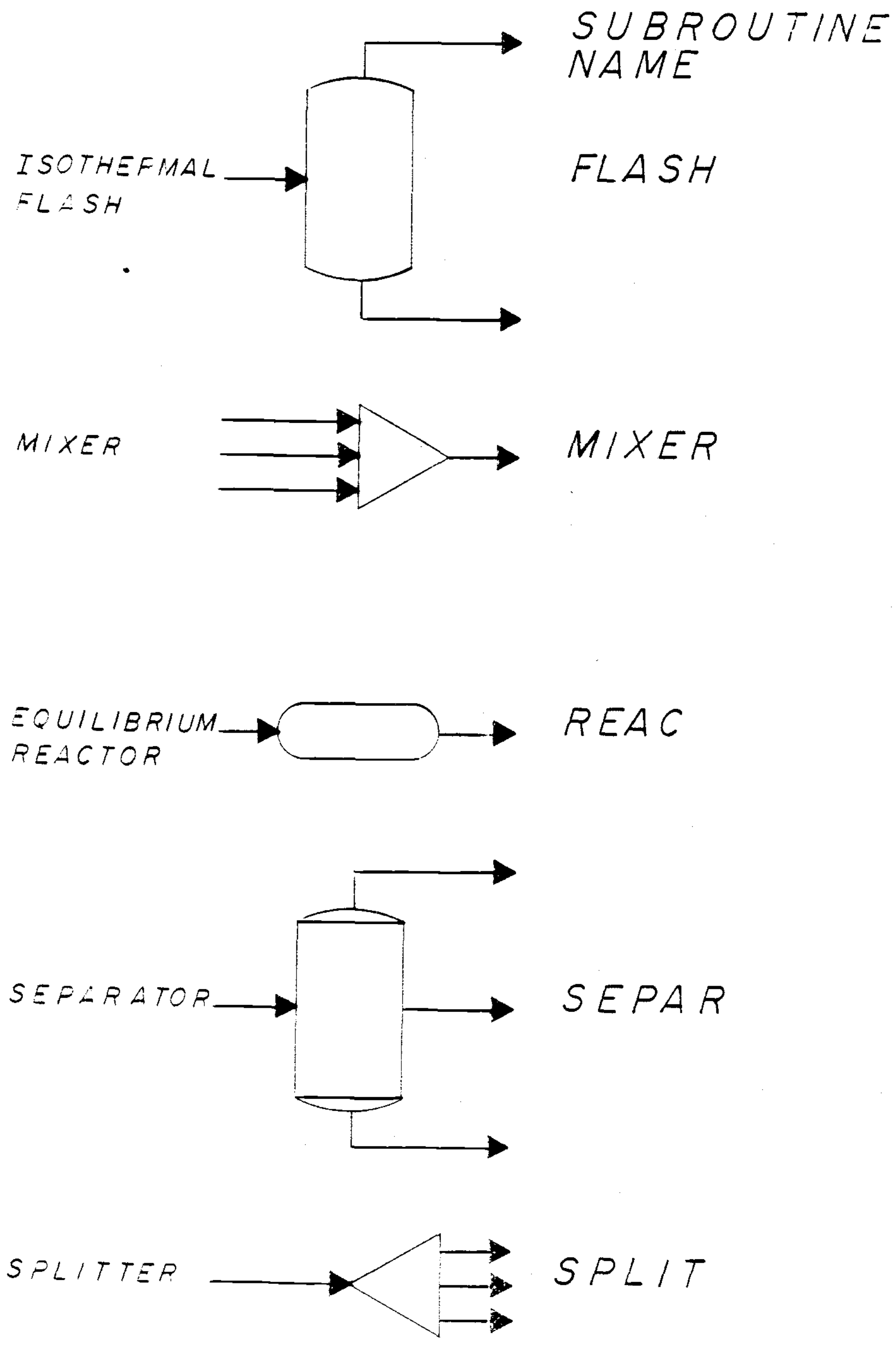


Figure III-2. Graphical Representation of the Subroutines Used with the Simultaneous Modular Approach

```
C
C
 ETHANAL PRODUCTION
1212345678123456712345671234567123456712345671234567123456712345671234567123456712345
 8        ETHOL  ETHAL  H2O    EE     EEE    H2      G      ORG
 1 263.77 .2616         7384
 2 39 1847  0266 03830  90590  00000  01830  .00000  00000  01090
 3 1000.   20000 00000  70000  00010  .00000 00000   00000  09900
 4
 5
 6   388   .4779  1685   .1476  1701  .0005   0044   .0088   023
 7
 8
 9
10
11
12
13 6000.   .0108  0806   9080   0000   0000   0006   0000   0000
14
15
16
17
18
19
20
21
22
23
24
25
26
12345678123456781234567812345678123456781234567812345678123456781234567812345678123456
 1  SEPAR
1        0.      0 05467 0.      0.      0       0       0
0        0       0 94251 1       0.      0       0.      0
 3      4        6       5                               3
 2  REAC
 -1      1.                                    1.
                                                                             27971
 4       7                                               1
 3  REAC
         -1.     -1        1.                  1.
                                                                             02591
  7      8                                                   2
 4  REAC
-4.666  -.5075  0.277778 0.      1.5341  0.      0 2     1
                                                                             02817
  8      9                                                       1
 5  SEPAR
0.04051 0.76344 0       0.      0       1       1       0
0 95949 0.23656 0.98817 0.08    1.      0       0       1
 9      10        12      11                             3
 6  SEPAR
0.      0.0273  0       0       0.      1       1       0
0       0.      0.      0.      0.      0.      0       0
 13     14       15
 7  SPLIT
 0      .8883    1117
 0.
15      19       20                                          2
 8  MIXER
 0.
 0
 10     18       19                                  13      3
 9  SEPAR
 0.     0. 0    0.      0.      0.      1 0     1 0     0 0
 0.
16      17       18
10  MIXER
  0
 .0
 2      12       20                                  21      3
11  SEPAR
0.      0.9764  0.      0.      0.      0       0       0
0.9936  0       0 9948  1       0       0       0       0 2581
 21     22       24      23                              3
12  MIXER
 0.
 0.
 1      24                                           3       2
13  SPLIT
        7333     2667
 0.
 6      25       26                                          2
```

Figure III-3. Example of a Datafile Used with SIMO

SUBROUTINE WRITES - This subroutine prints stream variables (Kayihan, 1979).

SUBROUTINE WRITEE - This subroutine prints equipment parameters (Kayihan, 1979).

SUBROUTINE WRITEX (LFIRST, LLAST) - This subroutine prints equipment parameters for modules LFIRST through LLAST.

SUBROUTINE SIMSO (I1, I2, I3, IP, NSIG) - A subroutine to solve mass balances using the sequential modular approach. If the problem has design constraints, subroutine SPEC must be used in conjunction with subroutine SIMSO. Parameters:

I1 - Total number of streams torn.

I2(I) - Number of each stream torn I = 1, I1.

I3 - Total number of design constraints.

IP - Printing Parameter. For IP = 1 no printing of stream variables or equipment parameters before solution. For IP = 2 stream variables are printed before solution. For IP = 3 stream variables and equipment parameters are printed before solution.

NSIG - Number of digits of accuracy required for the solution.

SUBROUTINE SPEC (N3, NAME, NUE, NUP, NACO, NUS, NCO, VAL) - A subroutine to specify design constraints imposed to the problem. Parameters:

N3 -	Total number of design constraints (max = 19)
NAME(I) -	Name of the modules which have equipment parameters manipulated. If a flowrate is manipulated, NAME(I) - 'FLOW;' I = 1, N3.
NUE(I) -	Number of the module or flowrate specified in NAME(I), I = 1, N3.
NUP(I) -	Number of the equipment parameter being manipulated. IF EQP(L,J) is manipulated NUP(I) = J. IF NAME(I) - 'FLOW,' NUP(I) = 0, I = 1, N3.
NACO(I) -	Name of constraint being imposed. There are only two possibilities 'FLOW' for molar flowrate or 'COMP' for composition. I = 1, N3.
NUS(I) -	Number of the stream which has the constraint NACO(I), I = 1, N3.
NCO(I) -	Number of the component being specified. If a flow rate is specified NCO(I) = 0, I = 1, N3.
VAL(I) -	Numerical value of the design constraint, I = 1, N3.

SUBROUTINE FCN (X, F, N) - This subroutine is used by subroutine MPDLM to evaluate function values at the tentative solution vector X.

SUBROUTINE TEARI (X, F, IT, N) - This subroutine assigns the values of X (from MPDLM) to the stream connection equations.

SUBROUTINE CONTI (X, F, IT, N) - This subroutine assigns the values of X (from MPDLM) to the manipulated equipment parameters or manipulated flowrates.

SUBROUTINE TEARO (X, F, IT, N) - This subroutine evaluates function values from stream connection equations.

SUBROUTINE CONTO (X, F, IT, N) - This subroutine evaluates function values from design constraint equations.

SUBROUTINE THREE (X, F, N, IP) - This subroutine performs three successive substitution iterations to initialize all stream variables.

Interconnection of the Subroutines

Subroutine MPDLM, the nonlinear solver subroutine, in its iterative procedure evaluates a tentative solution vector X, which is used to calculate function values. The variables in vector X correspond to the molar flowrate of each component in the torn stream(s) and to the manipulated variable used to meet some design specification. The function of subroutines TEARI and CONTI is to assign the variables of the vector X to the flowsheet variables, that is, each element of vector X is assigned to variables used by the equipment modules.

We will use a simple example to show the flow of information between the subroutines of SIMO. All compositions used are in mole percent.

Example III-1

A stream containing 50 percent of A and 50 percent B is to be separated into two streams, one containing 90 percent of A and the other 90 percent of B. The block diagram of such a system is shown in figure III-4.

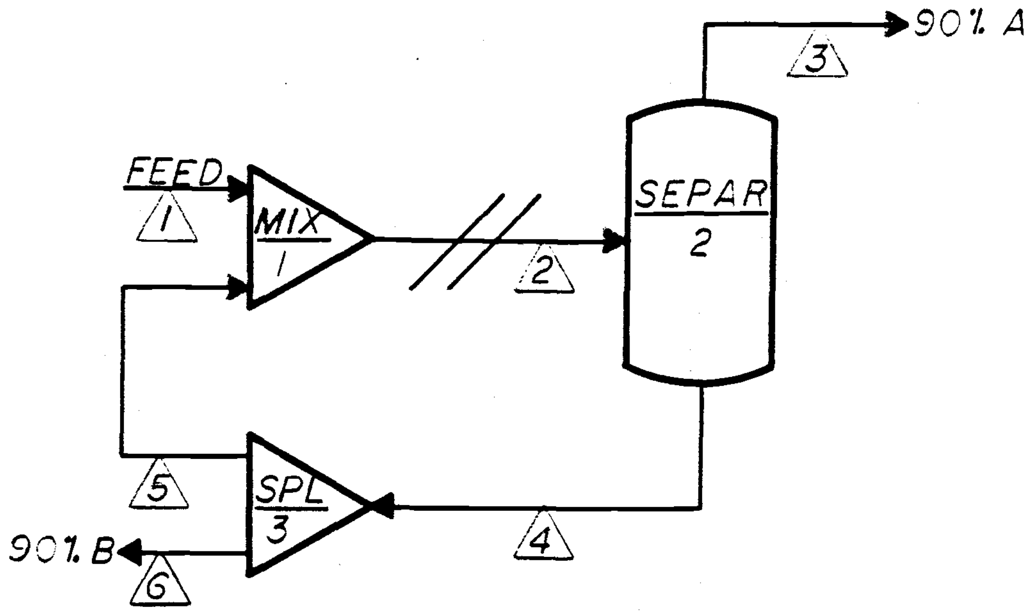
For this problem we choose stream 2 as the tear stream. The design constraints imposed for the problem are: COMP(3,1) = 0.9; COMP(6,2) = 0.9. The variables manipulated to meet that specification are EQP(2,1) and EQP(2,2).

To solve this problem the first step is to create a data file with all the known parameters and an initial estimate of the variables in the torn stream. Table III-1 shows one possibility of such values.

Table III-1. Possible Initial Guesses for Example III-1

STREAM	FLOW	COMP(I,1)	COMP(I,2)
1	100	.5	.5
2	150 1/	.3 1/	.7 1/
3	--	--	--
4	--	--	--
5	--	--	--
6	--	--	--
MODULE	**EQP(I,1)**	**EQP(I,2)**	**EQP(I,3)**
1	NR	NR	NR
2	.9 1/	.1 1/	NR
3	NR	0.5	0.5

NR - not required; 1/ - estimated



FLOW(1)	= 100		
COMP(1,1)	= COMP(1,2)		= 0.5
SPLIT FRACTION	= EQP(3,2)		= 0.5
SEPARATION FRACTION	= EQP(2,1),	EQP(2,2)	= ?
COMP(3,1)	= 0.9		
COMP(6,2)	= 0.9		

Figure III-4. Block Diagram of Example III-1

Next the main program and Subroutine FSIS are created (FSIS with the modules in its sequential order of calculation),

```
SUBROUTINE FSIS (X, F, IT, N)
CALL SEPAR (2)
CALL SPLIT (3)
CALL MIXER (1)
RETURN
END
```

And the main program,

```
PROGRAM EXAMPLE
INTEGER NUE(2), NUP(2), NUS(2), NCO(2)
REAL VAL(2)
CHARACTER * 5, NAME(2), NACO(2)
I1 = 1
I2 = 2
I3 = 2
NAME(1) = 'SEPAR'
NAME(2) = 'SEPAR'
NACO(1) = 'COMP'
NACO(2) = 'COMP'
NUE(1) = 2
NUE(2) = 2
NUP(1) = 1
NUP(2) = 2
NUS(1) = 3
NUS(2) = 6
NCO(1) = 1
NCO(2) = 2
VAL(1) = 0.9
VAL(2) = 0.9
IP = 3
NSIG = 4
CALL SPEC(I3,NAME, NUE, NUP, NACO, NUS, NCO, VAL)
CALL SIMSO(I1, I2, I3, IP, NSIG)
CALL WRITES
CALL WRITEE
STOP
END
```

Subroutine SPEC sets up 7 vectors with all the information in a COMMON block, so the information can be shared by other subroutines.

The information given as CHARACTER (NAME and NACO) is coded into INTEGER type variables. Furthermore, SPEC also codes the type of modules containing manipulated parameters because the equipment parameters used by each module have different allocations. For instance, a reactor module (say, module 7) can have only the conversion of a reactant manipulated and that variable is stored at EQP(7,20). A splitter module does not use the first equipment parameter, EQP(I,1). In short, the variables in the COMMON block are a numerical code of the design specifications, in this form it is easy to identify in which modules are the manipulated variables, and which equipment parameters are variables.

Subroutine SIMSO will first determine the total number of unknowns (and equations) that MPDLM will be solving. This task is rather easy because the number of variables in each torn stream is equal to the number of component flowrates of that stream; the number of design constraints is specified by the user. So, if there are two streams torn and 10 components in each stream and 3 design specifications, the total number of variables is $N = 2 * 10 + 3 = 23$. In the example III-1, $N = 1 * 2 + 2 = 4$.

The first NC elements of X, NC being the number of components in each stream, are assigned to the molar flow rates of the first tear stream. The next NC elements of X are assigned to the molar flow rates of the second tear stream, etc. After the molar flow rates of all tear streams are assigned to X, the initial estimate of the manipulated variables are assigned to X.

Before MPDLM is called by SIMSO, three successive substitution iterations are performed. There are two reasons to do so. First, a check of the consistency of the initial guesses is performed. There is a possibility that the initial estimates will generate negative flow rates, or some unexpected results, so the three initial iterations "stabilize" the initial guesses. If some unexpected result is generated, the user has the option of terminating the run. Second, the successive substitution operations are used as a convenient way of checking that the number of equations and the number of unknowns are the same.

When MPDLM is called, the control of the program passes to this nonlinear solver subroutine, and it will either find a solution or fail to solve the problem. It is not rare to have a simulation fail either because the maximum number of iterations allowed is reached or the unknowns calculated by MPDLM during the iterations are infeasible. Infeasible situations include negative flow rates and values of split fractions, conversions of a reactant, or separation fractions outside the interval [0,1]. In either case, a new set of initial guesses is required.

Subroutine MPDLM is linked with the flowsheet through subroutine FCN. The sole purpose of FCN is to evaluate function values at vector X during the iterations. FCN, in turn, will call the following subroutines: TEARI, CONTI, FSIS, TEARO and CONTO. As earlier discussed, TEARI and CONTI assign the values of X to FLOW(I), COMP(I,J) and EQP(K,L). Now that the torn stream(s) and manipulated variables are defined, subroutine FSIS is called and one pass over the flowsheet is performed. Next TEARO and CONTO evaluate function values. TEARO calculates the values of the residual functions of stream connection equations and CONTO calculates the values of the residual functions of design specification (constraints) equations. When MPDLM reaches the solution, the control of the program goes back to SIMSO, and from SIMSO to the user supplied main program.

A block diagram of the hierarchy of SIMO is presented in figure III-5. A block diagram of the most important operations performed by SIMO in the example III-1 is presented in figure III-6.

A feature introduced in SIMO is the possibility of introducing algebraic equations relating some flowsheet variables. For instance, assume that in the example of figure III-3, we would like to find a split fraction, ∂_1, which satisfies the following equation:

$$G(\partial_1) = (\mathrm{FLOW}(5))^2 - \mathrm{FLOW}(4) = 0 \qquad \text{(III-1)}$$

Equation III-1 is another constraint imposed on the problem. The following commands would be added to the main program of page 72.

```
N3 = 3
NAME(3) = 'SPLIT'
NACO(3) = 'USER'
NUE(3) = 3
NUP(3) = 2
NUS(3) = 0
NCO(3) = 0
VAL(3) = 0
```

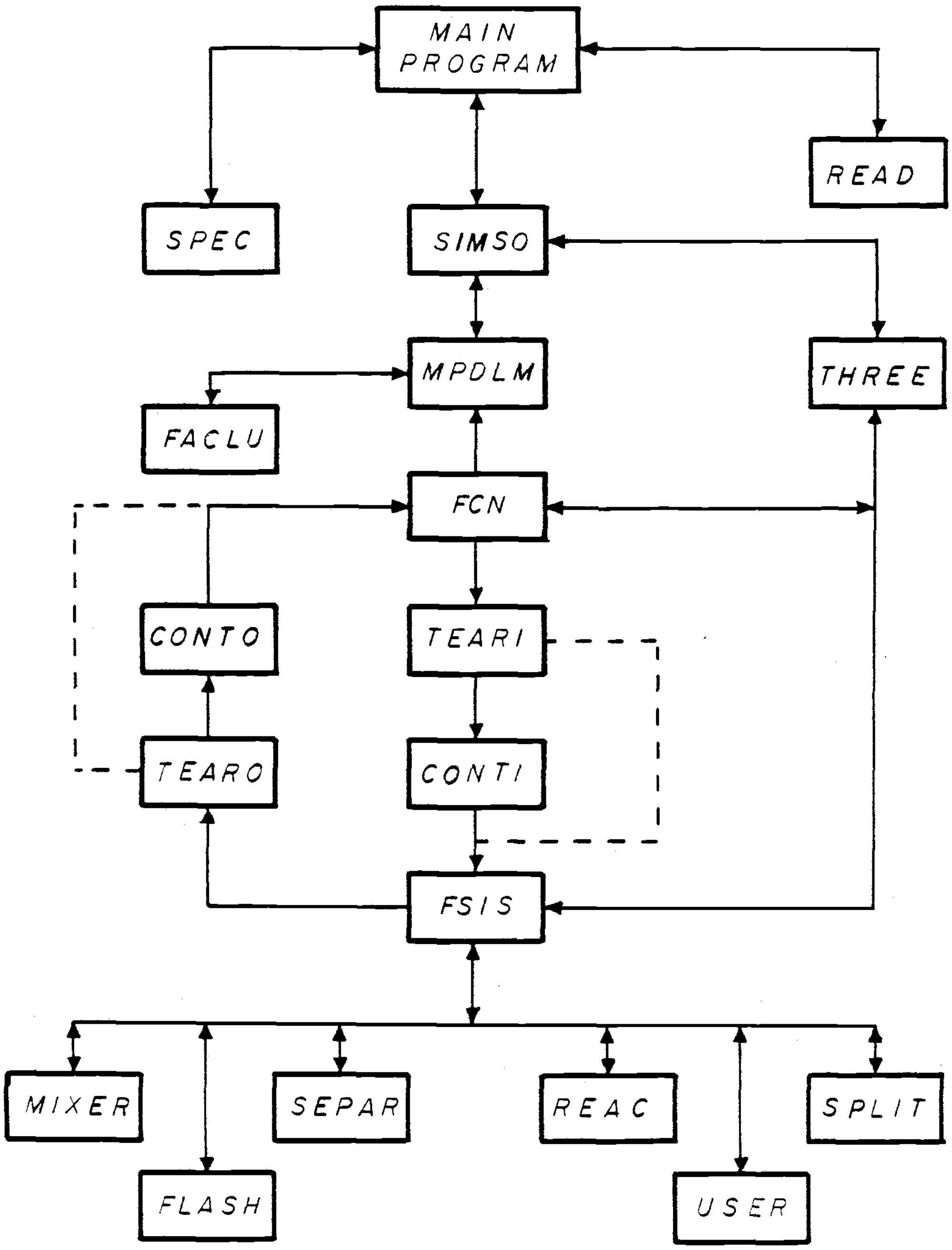


Figure III-5. Hierarchy of SIMO Subroutines

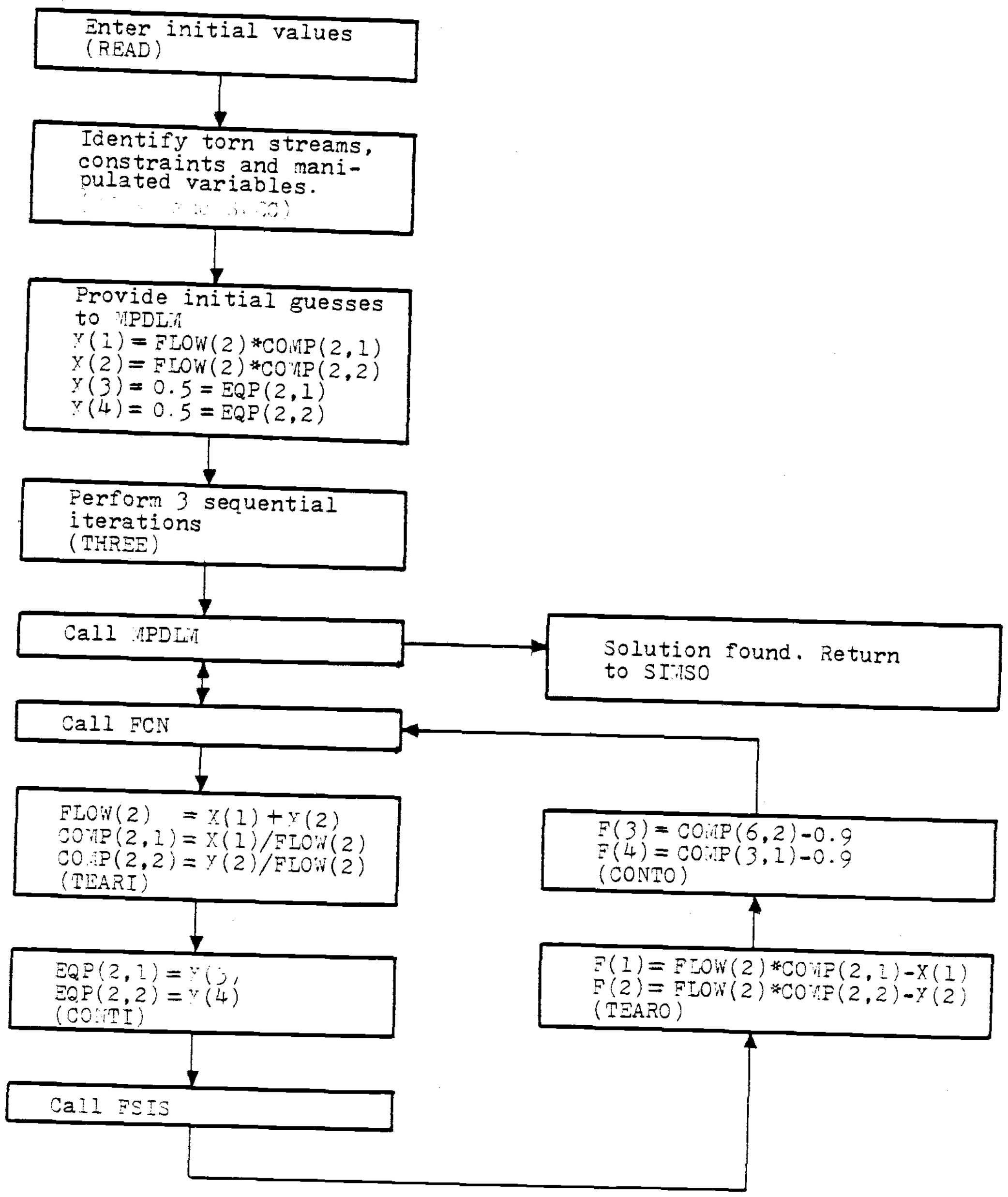


Figure III-6. Most Important Operations Performed by SIMO for the Example III-1

and subroutine FSIS becomes:

```
SUBROUTINE FSIS (X, F, IT, N)
COMMON(S1) FLOW(30), ...
CALL SEPAR(2)
CALL SPLIT(3)
CALL MIXER(1)
F(IT) = FLOW(5) ** 2 - FLOW(4)
IT = IT + 1
RETURN
END
```

EQSS Library

This library of subroutines is used to perform chemical process simulations using the equation-based approach.

The general structure of EQSS is similar to the SIMO library. Actually some subroutines from SIMO are also used with EQSS. In a similar fashion as in SIMO, EQSS can be divided in three major categories:

1. Equipment Modules
2. Support Subroutines
3. Nonlinear Equations Solver

Equipment Modules

As in the simultaneous modular approach, each equipment module has a mathematical model of the material balances around that equipment. Although the equations are essentially the same, they are written in the form F(X) = 0; in the simultaneous modular approach some algebraic manipulation had to be done in order to calculate output streams, given input streams and equipment parameters. It was quite simple to generate the source code of the equipment modules as the equations are simple material balances. However, some decisions had to be made regarding the use of 1) molar flow rates of each species, or 2) total molar flow rate and mole fractions as the independent variables. Using the molar flow rate of each species may be attractive when simple simulations are performed, that is, simulations without design constraints. In this case almost all equations are linear, the exceptions being the flash module and, depending on the case, the reactor module.

Early tests with EQSS showed that the gain in having most of the equations linear was offset by the flash module. The equilibrium relations in a flash drum have the following form:

$$F(N) = y_i - k_i x_i \qquad \text{(III-1)}$$

where:

i	=	component index
x_i	=	mole fraction of the liquid leaving the system
y_i	=	mole fraction of the vapor leaving the system
k_i	=	equilibrium constant for component i
N	=	(N_{vi}, N_{li}) = molar flow rate
l	=	subscript indicating liquid phase
v	=	subscript indicating vapor phase

In solving nonlinear systems using Newton-based approaches, the Jacobian of the system is required every iteration. The ith row of the Jacobian corresponding to equation III-1 would be:

$$J_i = \frac{\partial F_i(N)}{\partial N_j} = \frac{\partial(y_j - k_j x_j)}{\partial N_j} \qquad \text{(III-2)}$$

assuming that only three components are present, i = 1,3

$$x_i = \frac{N_{1i}}{N_{11} + N_{12} + N_{13}} \qquad \text{(III-3a)}$$

and

$$y_i = \frac{N_{vi}}{N_{v1} + N_{v2} + N_{v3}} \qquad \text{(III-3b)}$$

Equation III-2 is equivalent to:

$$\frac{\partial\, F_i(N)}{\partial N_j} = \frac{\partial(y_j)}{\partial N_j} - \frac{\partial(k_j x_j)}{\partial N_j} \qquad j = 1,3 \qquad \text{(III-4)}$$

and

$$\frac{\partial(y_i)}{\partial N_j} = \frac{\partial\left[\dfrac{N_{vj}}{N_{v1} + N_{v2} + N_{v3}}\right]}{\partial N_j} \qquad j = 1,3 \qquad \text{(III-5)}$$

or

$$\frac{\partial(y_i)}{\partial N_j} = \frac{N_{v1} + N_{v2} + N_{v3} - N_{vj}\,(\sum_{k=1}^{3} N_{vk})}{(N_{v1} + N_{v2} + N_{v3})^2} \qquad k \neq j \qquad \text{(III-6)}$$

Similarly,

$$\frac{\partial(k_j x_j)}{\partial N_j} = \frac{-k_j\,[N_{11} + N_{12} + N_{13} - N_{1j}\,(\sum_{k=1}^{3} N_{1k})]}{(N_{11} + N_{12} + N_{13})^2} \qquad \text{(III-7)}$$

where $k \neq j$

It is clear that the rows corresponding to equation III-2 are dense. For the equation-based approach we must explore the sparsity of the nonlinear system. The formulation using molar flow rates of individual components would create a row with 2 * NC elements for a flash drum. If k values are considered variables the number of elements goes to 2 * NC + 1. On the other hand, using total flow rates and mole fractions, only two elements per row are generated (three elements if k values are not constant).

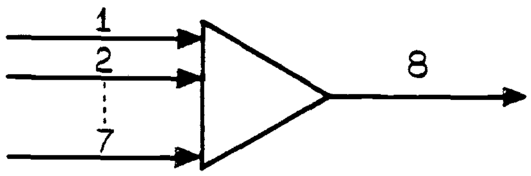
In this work we will use total flow rates and mole fractions as independent variables.

We will now present equipment subroutines. The following nomenclature will be used.

W_i	=	molar flow rate of the ith stream (mole/unit of time)
x_{ij}	=	mol fraction of the jth component in the ith stream
U_i	=	ith equipment parameter
F(I)	=	numerical value of the ith equation
NC	=	number of components
IT	=	internal counter

The equations will be written for the case of two components. The extension to NC components is straightforward.

SUBROUTINE SMIXER (NE, N, F, IT) - This subroutine simulates the mixing of up to 7 incoming streams. No equipment parameters are required.



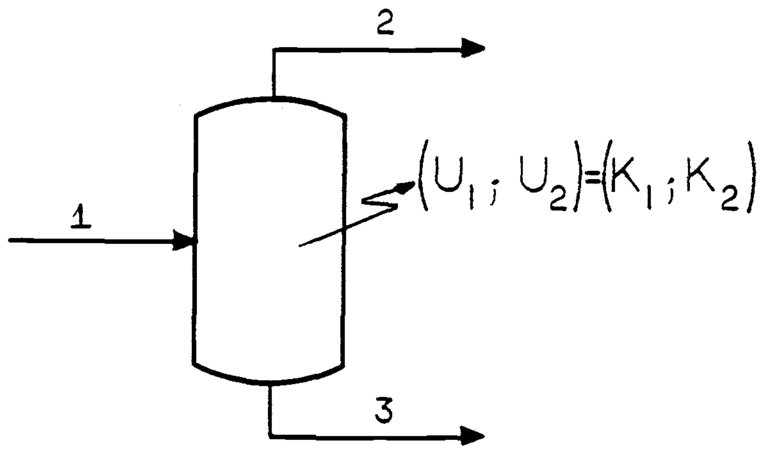
$$F(1) = W_8 * X_{81} - \sum_{i=1}^{7} W_i * X_{i1} = 0$$

$$F(2) = W_8 * X_{82} - \sum_{i=1}^{7} W_i * X_{i2} = 0$$

$$F(3) = 1 - X_{81} - X_{82} = 0$$

Total Number of Equations = NC + 1

SUBROUTINE SFLASH (NE, N, F, IT) - This subroutine simulates an isothermal flash. There are NC equipment parameters: $U_i = k_i$ equilibrium constants.



$$F(1) = W_1 * X_{11} - W_2 * X_{21} - W_3 * X_{31} = 0$$

$$F(2) = W_1 * X_{12} - W_2 * X_{22} - W_3 * X_{32} = 0$$

$$F(3) = X_{21} - k_1 * X_{31} = 0$$

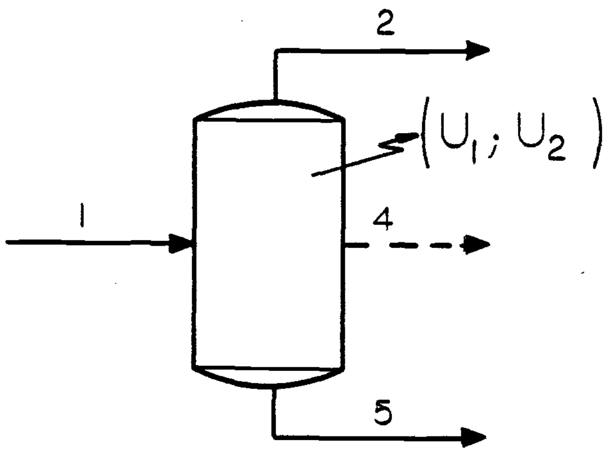
$$F(4) = X_{22} - k_2 * X_{32} = 0$$

$$F(5) = 1 - X_{21} - X_{22} = 0$$

$$F(6) = 1 - X_{31} - X_{32} = 0$$

Total Number of Equations = 2 * NC + 2

<u>SUBROUTINE SSEPAR</u> (NE, N, F, IT) - This subroutine can for example simulate a simple distillation column. There are NC or 2 * NC equipment parameters (2 or three output streams): separation fractions.



$$F(1) = W_1 * X_{11} * U_1 - W_2 * X_{21} = 0$$

$$F(2) = W_1 * X_{12} * U_2 - W_2 * X_{22} = 0$$

$$F(3) = W_1 * X_{11} - W_2 * X_{21} - W_3 * X_{31} = 0$$

$$F(4) = W_1 * X_{12} - W_2 * X_{22} - W_3 * X_{32} = 0$$

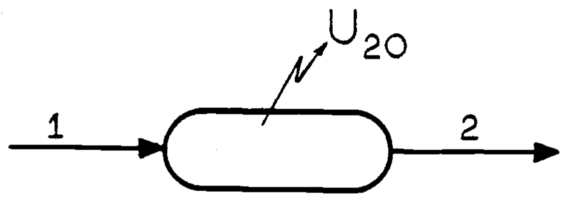
$$F(5) = 1 - X_{21} - X_{22} = 0$$

$$F(6) = 1 - X_{31} - X_{32} = 0$$

$$U_i = \frac{W_2 * X_{2i}}{W_1 * X_{1i}}$$

Total Number of Equations = 2 * NC + 2

SUBROUTINE SREACTOR (NE, N, F, IT) - This subroutine simulates a simple reactor. Equipment parameters are the stoichiometric coefficients and the conversion of a reactant.



Assume the conversion, U_{20}, is for component 1, in addition U_1 and U_2 are the stoichiometric coefficients of components 1 and 2.

$$R = W_1 * X_{11} * U_{20}/(-U_1)$$

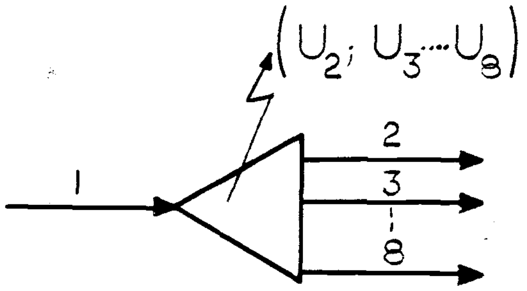
$$F(1) = W_2 * X_{21} - W_1 * X_{11} - U_1 * R = 0$$

$$F(2) = W_2 * X_{22} - W_1 * X_{12} - U_2 * R = 0$$

$$F(3) = 1 - X_{21} - X_{22} = 0$$

Total Number of Equations = NC + 1

<u>SUBROUTINE SSPLIT</u> (NE, N, F, IT) - This subroutine simulates a splitter. Up to 7 product streams may be specified. Equipment parameter: split fraction (up to 7).



Assume that stream 1 is split into two streams, 2 and 3. Then,

$$F(1) = W_1 - W_2 - W_3 = 0$$

$$F(2) = W_2 - W_1 * U_2 = 0$$

$$F(3) = X_{11} - X_{21} = 0$$

$$F(4) = X_{12} - X_{22} = 0$$

$$F(5) = X_{11} - X_{31} = 0$$

$$F(6) = X_{12} - X_{31} = 0$$

If equipment parameters are also variables, one more equation is required:

$$F(7) = 1 - U_2 - U_3 = 0$$

Total Number of Equations = L + 2 * NC where L is the number of output streams.

The user has the option of inserting special purpose modules. The module must have the form

SUBROUTINE USER (NE, N, F, IT)

On input, IT is the number of the next equation. On output IT must be equal to IT + NEQ, where NEQ is the number of equations in the USER module. For example,

```
F(IT)      = user equation 1
F(IT + 1)  = user equation 2
F(IT + 2)  = user equation 3
IT         = IT + 3
```

Support Subroutines

This block of subroutines are used to enter data, print results and prepare the system of nonlinear equations for solution. Four subroutines from SIMO are also used with EQSS: Subroutines READ, WRITES, WRITEE and WRITEX.

SUBROUTINES SREAD (NSTR, NEQ, NC) - This subroutine is used to identify equipment modules used in the simulation, as well as identifying which flowsheet variables are known constants or unknowns to be calculated. SREAD will assign values of 0 or 1 to vectors IFK(N), ICK(N x N) and IEK(N x N). If a flow rate is a known constant, the corresponding element of IFK will be one. Similarly, ICK and IEK will be one if a composition or equipment parameter are a known constant. Otherwise, IFK, ICK and IEK will be assigned zero.

Another vector, NAME, will carry the name of each module used in the simulation. The parameters required by SREAD are:

NSTR - total number of streams

NEQ - total number of equipment modules

NC - number of components in each stream.

SREAD requires a data file; an example is shown in figure III-7.

SUBROUTINE IDEN - This subroutine plays an important role in setting up the system of nonlinear equations. The subroutine has four major duties:

1. Identifies equipment modules and assigns to vector ID(I) the values shown in table III-2.

Table III-2. Values of Vector ID for a Given Equipment Module

Ith Equipment Module	ID(I)
SPLITTER	1
MIXER	2
REACTOR	3
SEPARATOR	4
FLASH	5
USER	6

2. Vector INOP(I) is assigned the total number of streams entering and leaving the Ith module. Matrix ITOP(I,J) is assigned the number of each stream leaving and entering the Ith equipment module, J = 1, INOP(I).

```
C
C
C
X12XXX12X01020304050607080910
  1    1  1 1 1 1 1 1 1 1 1 1
  2    1  1 1 1 1 1 1 1 1 1 1
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 16
 17
 18    1
 19
12XAAAAAX01020304050607080910
 1 SPLIT    1 1
 2 MIXER
 3 SEPAR  1 1 1 1 1 1 1 1 1 1
 4 MIXER
 5 SEPAR  1 1 1 1 1 1 1 1 1 1
 6 MIXER
 7 SEPAR  1 1 1 1 1 1 1 1 1 1
 8 SEPAR  1 1 1 1 1 1 1 1 1 1
 9 SPLIT    0 0
10 SEPAR  1 1 1 1 1 1 1 1 1 1
C
C
C
```

Figure III-7. Example of a Datafile Required by SREAD

3. Vector IEC(I) is assigned with the total number of unknown equipment parameters of the I-th module. IEK(I,J) is assigned with the number of each unknown parameter of the I-th equipment, J = 1, IEC(I).

4. When sequential iterations are used to generate initial estimates of all variables, some variables, which are known constants, may have their value changed by the sequential pass. When subroutine SREAD is called, it assigns to a matrix CX(I,J) the numerical values of known flow rates (FLOW(I)) and known mole fractions (COMP(I,J)). Subroutine IDEN reassigns the values of CX(I,J) to FLOW(I) and COMP(I,J). The use of sequential iterations to generate initial estimates of all variables will be treated in detail at the end of this chapter.

SUBROUTINE SIMMAN (NSIG, MS) - This subroutine serves as a "manager" of the library. First, it performs a scaling of all flow rates, by searching for the maximum flow rate, FMAX, and then dividing all flow rates by FMAX. Second, it checks the consistency of the system of nonlinear equations. If the number of equations and variables are not equal, an error message is printed and the program stops. If the system of equations is consistent, stream variables and equipment parameters with the known constants and initial guesses are printed.

Last, SIMMAN calls MPDLM, the nonlinear solver subroutine.

SUBROUTINE FCN (X, F, N) - This subroutine is used by MPDLM to pass the vector X, with tentative values to the solution, and to pass to MPDLM function values. FCN will call two other subroutines, VAROUT and FUVAL. In the following pages both subroutines will be described.

SUBROUTINE VARIN (X, IT) - This subroutine is used by SIMMAN to assign the initial guesses of flow rates, compositions and equipment parameter to the vector X. The variables are stored in X by equipment modules. For instance, module number 1 has 1 input stream and two output streams, say, streams 1, 2, and 3. Also there are 2 unknown equipment parameters and each stream has 3 components. Vector X will have the following variables assigned:

X(1) = FLOW(1)	X(5) = FLOW(2)	X(9) = FLOW(3)
X(2) = COMP(1,2)	X(6) = COMP(2,1)	X(10) = COMP(3,1)
X(3) = COMP(1,2)	X(7) = COMP(2,2)	X(11) = COMP(3,2)
X(4) = COMP(1,3)	X(8) = COMP(2,3)	X(12) = COMP(3,3)
	X(13) = EQP(1,1)	
	X(14) = EQP(1,2)	

SUBROUTINE VAROUT (X, IT, N) - This subroutine does exactly the opposite of VARIN. It assigns the values of X to the stream variables and equipment parameters.

SUBROUTINE FUVAL (F, IT, N) - FUVAL calls all equipment modules subroutines. Each equipment module subroutine will evaluate function values which will be stored in vector F and later on used by MPDLM.

Nonlinear Solvers Subroutine

The subroutine used to solve the nonlinear system is MPDLM which was described in the SIMO section. The only difference is that it uses sparse matrix techniques and Schubert's update formula. Details may be found in Chapter II.

SUBROUTINE SPAMA2 (N, IS, LB, X) - Solves a system of N linear equations. The source code of this subroutine was obtained from Rodriques (1979). It uses sparse matrix techniques to speed up execution time and save computer memory.

Interconnection of the Subroutines

Once the data file is created and the values read through subroutine SREAD, the next step is to call subroutine SIMMAN. This subroutine will first perform a scaling of the flow rates, then call subroutines IDEN, VARIN and FUVAL, check the consistency of the system, stop the program or print stream variables and equipment parameters, and then call MPDLM. Now the control of the program passes to the nonlinear solver subroutine. MPDLM will call subroutine FCN several times with tentative solution vectors. FCN in turn calls subroutine VAROUT, which assigns the values of X to the stream variables and equipment parameters. Next, FCN calls FUVAL and all the function values are evaluated. FCN returns these function values to MPDLM.

A block diagram of the strategy is shown in figure III-8.

An important point that should be addressed is how to supply initial estimates of all variables.

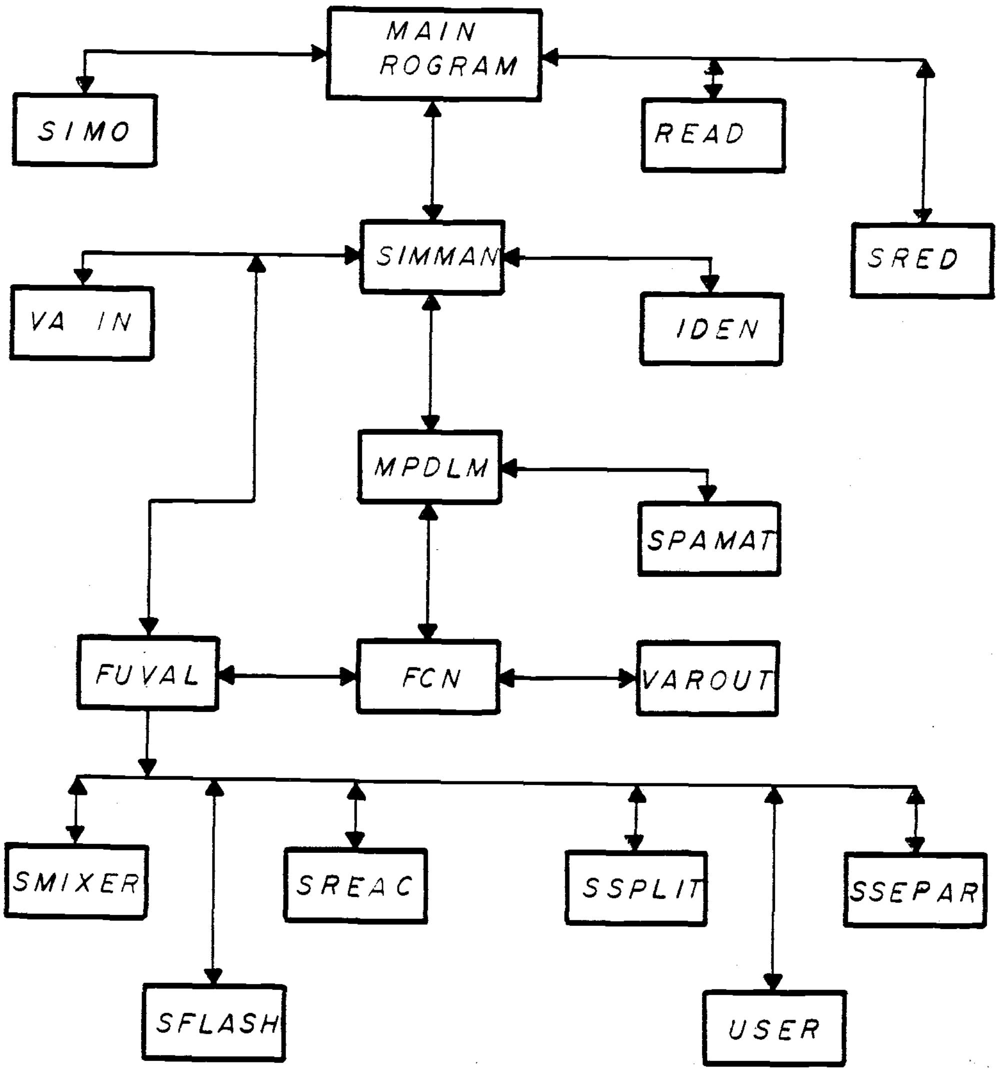


Figure III-8. Flow of Information Inside EQSS

There are two possibilities. One is to supply the initial estimates in a data file through subroutine READ. The task is very simple if the problem has a few modules and a dozen streams. However, if there are several modules and streams, and the number of components is high the task is tedious and error prone.

The second possibility is to perform one sequential iteration. As SIMO and EQSS have the same structure, and both use the same common block for the stream variables and equipment parameters, performing one sequential iteration in the system, that is, using the equipment modules from SIMO to evalute stream variables is an attractive alternative.

Nevertheless, the strategy has a serious drawback. The modules from SIMO required the complete definition of the input stream(s) and the required equipment parameters. Once the calculations are performed in a given module, all residual functions of that module when calculated by the equivalent module of EQSS are equal to zero. For instance, assume that we want to solve the problem of figure III-3 using EQSS, and we want to supply the initial guesses through SIMO subroutines. We would write a small subroutine with the equipment modules in its sequential order of calculation, and would supply an initial guess for the tear stream and unknown equipment parameters. The subroutine would be:

```
SUBROUTINE IGUESS
CALL SEPAR(2)
CALL SPLIT(3)
CALL MIXER(1)
RETURN
END
```

So, for a given set of initial guesses of stream 2 and equipment parameters EQP(2,1) and EQP(2,2), the separator module would calculate the values of streams 3 and 4. Those values are the exact solution of the separator module. With the values obtained for stream 4, the splitter module will calculate stream 5 and 6, which are the exact solution of the splitter equations. The same exact solution will be obtained with the mixer module. Of course, now stream 2 has different values than the initial guess, unless the initial guess was the solution. Besides the residual functions from the separator module, all other residual functions of the system were equal to zero.

The fact that most of the equations were mathematically satisfied, although the complete system was not, created unexpected results from MPDLM. Most of the times it would not converge. In a few occasions it would converge, but the number of iterations was excessively high.

To overcome this problem we introduced "errors" in all variables calculated by the sequential iteration. Truncating the values of the variables at the third or fourth decimal place was enough to solve all problems.

CHAPTER IV

PERFORMANCE OF EQSS AND SIMO

As discussed in previous chapters, the basic problem encountered in process simulation is the need to efficiently solve a set of nonlinear equations. In Chapter II we showed an algorithm to solve nonlinear equations which, we believe, is one of the most efficient available in the open literature.

The critical point is to define the measures used to compare different approaches to solve the equations of a chemical process simulation. In general, execution time plays an important role in commercial simulations, but, as computers become faster and cheaper, execution time will become less important. It is more important to have a simulator which will not fail to solve simple problems, or will not require several runs with different initial guesses to achieve convergence in a more complex problem.

In this work our basis for comparison will be first, the number of iterations required to solve a specific problem, and second, execution time.

Each approach (sequential modular, simultaneous modular or equation-based) uses a different subroutine to solve the set of nonlinear equations. The sequential modular approach uses Wegstein's method (subroutine WEGSMD) to solve problems without design specifications; if design specifications are part of the problem, subroutines WEGSMD and SECNEW are used. The simultaneous modular approach uses version 1 of subroutine MPDLM. The equation-based approach uses version 2 of MPDLM (see Chapter II). For all three approaches the stopping criteria used in the nonlinear solver subroutine is the same, i.e., 1×10^{-4}, which means that each variable had a change smaller or equal to 1×10^{-4} between the last two iterations.

Five problems were simulated using the simultaneous modular approach (SIMO), the equation-based approach (EQSS) and for comparison, the sequential modular approach (SEQ). Four of the problems have at least two versions, one with 1 or more design specifications (constraints) and one without constraints.

Of the five problems, four are from the open literature (Cavett's four-flash drum problem, ammonia plant, nitric acid plant, and gasoline recovery); the remaining problem is a section of a 2 ethyl-hexanal plant in which ethanal is produced by the catalytic dehydrogenation of ethanol. The data for the ethanal plant was obtained from Elekeiroz do Nordeste Industria Quimica S.A. (Brazil) through the document EN-001-00-TU3-015.

In table IV-1 we summarize the problems simulated.

Table IV-1. Summary of Test Problems

Problem	Number of Design Specifications		
	EQSS	SIMO	SEQUENTIAL
Ammonia Plant	0,1	0,2	0,2
Nitric Acid Plant	7	0,1,7	0,1
Gasoline Recuperation	1	0,1	0,1
Ethanal Plant	---	0,1	0,1
Cavett Problem	0	0	0

It should be noted that only material balances are performed in either approach, and the influence of physical property calculations can not be analyzed. It is well known that physical property calculations account for a significant amount of computer's time in process simulation, however, there is no loss of generality in this work as all problems are set up in the same manner.

IV-1 Cavett's Four Flash-Unit Problem

This test problem was suggested by Cavett (Cavett, 1963). The problem has been used by several authors to compare the performance different tear streams sets on convergence. See, for example, Westerberg et al. (1979), Rosen and Pauls (1975), and Shachan and Motard (1974).

The block diagram of the process is presented in figure IV-1. Feed compositions and equilibrium constants (k-values) are presented in table IV-2.

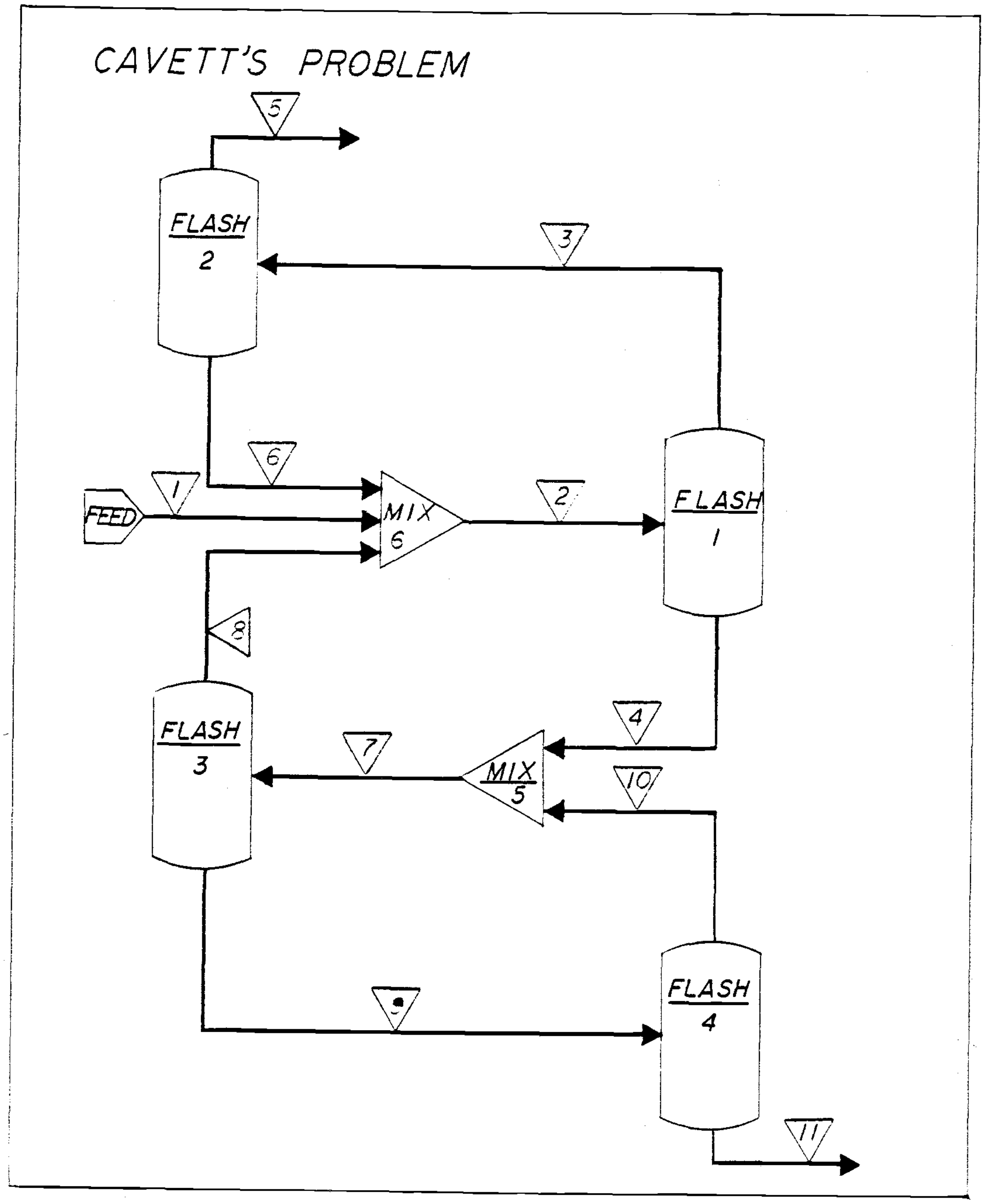


Figure IV-1. Block Diagram of Cavett's Problem

Table IV-2. Feed Composition and K-Values Used in Cavett's Problem (Rosen and Pauls, 1977)

		K-Values			
Component	Mole Fraction	Flash 1	Flash 2	Flash 3	Flash 4
N_2	0.0131	24.260	5.940	149.700	620.800
CO_2	0.1816	4.640	1.510	21.100	72.300
H_2S	0.0124	2.030	0.890	8.280	27.100
CH_4	0.1096	10.300	3.090	52.900	200.100
Ethane	0.0876	2.660	1.000	11.200	39.300
Propane	0.0838	0.943	0.502	3.290	10.800
Isobutane	0.0221	0.445	0.310	1.340	4.220
n-Butane	0.0563	0.342	0.246	0.990	3.070
Isopentane	0.0289	0.164	0.155	0.417	1.220
n-Pentane	0.0413	0.132	0.126	0.327	0.944
n-Hexane	0.0646	0.051	0.064	0.107	0.290
n-Heptane	0.0954	0.022	0.035	0.039	0.101
n-Octane	0.0675	0.008	0.017	0.013	0.033
n-Nonane	0.0610	0.004	0.009	0.005	0.012
n-Decane	0.0304	0.002	0.005	0.002	0.004
n-Undecane	0.0444	0.0008	0.003	0.0009	0.002
Temperature (K)	--	322	311	309	303
Pressure (bar)	--	9.6	56.2	4.39	1.91

For this problem, streams 2 and 7 were chosen as tear streams. As initial estimates for the stream variables of the torn streams, we used the stream variables of the feed (stream 1). Neither the tear streams, nor the initial estimates of streams 2 and 7, are the best, but in simulations with all three approaches - sequential, simultaneous modular and equation based - the same set of initial conditions were used, thus a fair comparison could be made.

The results obtained are summarized in table IV-3.

Table IV-3. Results for Cavett's Problem

Method	ITER[1/]	FEVAL[2/]	CPU-TIME[3/]	# EQ[4/]
EQSS	5	--	143.995	170
SIMO	6	43	21.167	32
SEQ	105	105	5.902	32

[1/] Number of iterations performed by the nonlinear solver. For EQSS and SIMD iterations were performed by MPDLM, for SEQ iterations were performed by WEGSMD or SECNEW.

[2/] Number of flowsheet evaluations.

[3/] Execution time in a CDC CYBER 175; opt. = 0.

[4/] Number of equations being solved.

In Cavett's problem, the sequential modular approach was superior to the equation based approach and simultaneous modular approach in terms of execution time. However, in terms of iterations required for solution the equation based approach and the simultaneous modular were far superior. Unfortunately, EQSS is rather slow to solve the linear system; the same happens with SIMO. But it is clear that only a few iterations are required to reach a solution.

The number of sequential iterations required by SIMO, 43, were used to perform the following operations: 3 iterations to initialize the problem, 32 iterations to evalute the Jacobian, and 8 sequential iterations required by MPDLM. In this case, the 32 flowsheet evaluations to obtain the Jacobian did not contribute significantly to the execution time. Roughly, 2.5 CPU were required to evaluate the Jacobian.

The main programs used for the three methods are listed in table IV-4. The solution of the problem is presented in table IV-5.

Westerberg et al. (1979) reported 73 sequential iterations to solve Cavett's problem with the sequential modular approach. Unfortunately, they did not report the initial estimates of the tear streams, or which procedure was used to initialize the variables. We believe our implementation of the sequential modular approach is correct, so the discrepancy in the number of iterations must be due to the initial estimate of the solution.

As it was earlier mentioned, Cavett's problem has been extensively used to evaluate process simulators. Chen (1982) reports that 5 iterations were required to solve Cavett's problem by the simultaneous modular approach using the same tear streams as in this work. However, the execution time Chen reports is 2.64 CPU-second, which is one order of magnitude smaller than the execution time obtained in this work. The computer used in Chen's

Table IV-4. Main Programs Used to Simulate Cavett's Problem: SCA - sequential modular approach
ECA - equation-based approach
SMCA - simultaneous modular approach

```
      PROGRAM SCA(TAPE5.TAPE6.OUTPUT=TAPE6)
      EXTERNAL FSIS
      COMMON /S1/ NS.NSTR(30).FLOW(30).NC.CNAME(20).COMP(30.20)
      COMMON /E1/ NE.NEQP(20).EQNAME(20).EQP(20.20).IEQP(20.8)
      COMMON /MWEG/ NSEQ
      COMMON /P31/ ITOT
      REAL X(32).F(32)
      ITOT=0
      NSEQ=450
      N=32
      NT=300
      EPS=1 E-4
      K=0
      CALL REAO(11.6)
      00 10 I=1.16
      X(I)=FLOW(2)*COMP(2.I)
   10 X(I+16)=FLOW(7)*COMP(7.I)
      CALL WEGSMO(N.X.NT.EPS.FSIS.K)
      WRITE(6.20) ITOT
   20 FORMAT(//.5X."CONVERGENCE ACHIEVED IN ".I4."  ITERATIONS ".//)
      CALL WRITES
      CALL WRITEE
      STOP
      ENO
C
C
C
C
      SUBROUTINE FSIS(M.X.F)
      REAL X(M).F(M)
      COMMON /S1/ NS.NSTR(30).FLOW(30).NC.CNAME(20).COMP(30.20)
      COMMON /E1/ NE.NEQP(20).EQNAME(20).EQP(20.20).IEQP(20.8)
      COMMON /P31/ ITOT
      ITOT=ITOT+1
      FLOW(2)=0.
      FLOW(7)=0.
      DO 10 I=1.16
      FLOW(2)=FLOW(2)+X(I)
   10 FLOW(7)=FLOW(7)+X(I+16)
      DO 20 I=1.16
      COMP(2.I)=X(I)/FLOW(2)
   20 COMP(7.I)=X(I+16)/FLOW(7)
      CALL FLASH(1)
      CALL FLASH(2)
      CALL FLASH(3)
      CALL FLASH(4)
      CALL MIXER(5)
      CALL MIXER(6)
      IF(ITOT.EQ.1) THEN
      CALL WRITES
      CALL WRITEE
      ELSE
      ENOIF
      00 30 I=1.16
      F(I)=FLOW(2)*COMP(2.I)
   30 F(I+16)=FLOW(7)*COMP(7.I)
      RETURN
      END
```

```
C
      PROGRAM ECA(TAPE5.TAPE6.TAPE9)
      COMMON /S1/ NS.NSTR(30).FLOW(30).NC.CNAME(20) COMP(30 20)
      COMMON /E1/ NE.NEQP(20).EQNAME(20).EQP(20.20) IEQP(20.8)
      CALL REAO(11.6)
      CALL SREAO(11.6.16)
      00 250 IF=1.3
      CALL FLASH(1)
      CALL FLASH(2)
      CALL FLASH(3)
      CALL FLASH(4)
      CALL MIXER(5)
      CALL MIXER(6)
  250 CONTINUE
      CALL WRITES
      00 10 I=1.NS
      IS=NINT(100*FLOW(I))
      FLOW(I)=IS/100
      DO 10 L=1.NC
      IS=NINT(COMP(I.L)*10000)
      COMP(I.L)=IS/10000
   10 CONTINUE
      CALL SIMMAN(2.2)
      STOP
      END
C
C



      PROGRAM SMCA(TAPE5.TAPE6)
      INTEGER N2(2)
      CALL REAO(11.6)
      N1=2
      N2(1)=2
      N2(2)=7
      N3=0
      IP=3
      NSIG=4
      CALL SIMSO(N1.N2.N3.IP.NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(X.F.IT.N)
      REAL X(N).F(N)
      CALL FLASH(1)
      CALL FLASH(2)
      CALL FLASH(3)
      CALL FLASH(4)
      CALL MIXER(5)
      CALL MIXER(6)
      RETURN
      END
```

Table IV-5. Solution of Cavett's Problem

```
 STREAM VARIABLES
 NSTR   FLOW    N2        CO2       H2S       CH4       C2H6      C3H8      IBUT      NBUT      IPENT     NPENT
 1  3452.10    .0131     .1816     .0124     .1096     .0876     .0838     .0221     .0563     .0289     .0413
 2  5595.16    .0091     .1799     .0178     .0856     .1112     .1810     .0394     .0831     .0264     .0347
 3  2279.66    .0210     .3363     .0255     .1841     .1764     .1747     .0227     .0388     .0066     .0071
 4  3315.50    .0009     .0725     .0126     .0179     .0663     .1853     .0510     .1136     .0400     .0537
 5  1703.51    .0265     .3676     .0247     .2221     .1764     .1397     .0145     .0219     .0028     .0026
 6   576.15    .0045     .2435     .0278     .0719     .1764     .2783     .0468     .0890     .0178     .0204
 7  3988.76    .0007     .0645     .0122     .0153     .0622     .2075     .0634     .1410     .0445     .0572
 8  1566.91    .0018     .1530     .0262     .0378     .1391     .3594     .0749     .1401     .0241     .0254
 9  2421.85    .0000     .0072     .0032     .0007     .0124     .1092     .0559     .1415     .0577     .0778
10   673.26    .0000     .0252     .0104     .0026     .0419     .3167     .1245     .2758     .0663     .0746
11  1748.59    .0000     .0003     .0004     .0000     .0011     .0293     .0295     .0898     .0544.    .0790

 STREAM VARIABLES
 NSTR   FLOW    N-HEX     N-HEP     N-OCT     N-NON     N-DEC     N-UND
 1  3452.10    .0646     .0954     .0675     .0610     .0304     .0444
 2  5595.16    .0442     .0612     .0422     .0379     .0188     .0274
 3  2279.66    .0037     .0022     .0006     .0003     .0001     .0000
 4  3315.50    .0720     .1018     .0709     .0637     .0317     .0463
 5  1703.51    .0008     .0003     .0000     .0000     .0000     .0000
 6   576.15    .0122     .0080     .0021     .0010     .0002     .0000
 7  3988.76    .0661     .0878     .0596     .0532     .0264     .0385
 8  1566.91    .0109     .0055     .0013     .0004     .0001     .0001
 9  2421.85    .1018     .1411     .0974     .0873     .0434     .0633
10   673.26    .0368     .0190     .0044     .0014     .0002     .0002
11  1748.59    .1268     .1881     .1332     .1204     .0600     .0877

  EQUIPMENT PARAMETERS          (2 X EQP(I,J))/IEQP(I,J)

                  -----------   MODULE # ( 1) =  FLASH       ----------
  24.260     4.640     2.030   10.300     2.660       .943      .445      .342      .164      .132
    .051      .022      .008     .004      .002       .000      .000      .000      .000      .000
   2         3         4         0         0          0         0         0

                  -----------   MODULE # ( 2) =  FLASH       ----------
   5.940     1.510      .890    3.090     1.000       .502      .310      .246      .155      .126
    .064      .035      .017     .009      .005       .003      .000      .000      .000      .000
   3         5         6         0         0          0         0         0

                  -----------   MODULE # ( 3) =  FLASH       ----------
 149.700    21.100     8.280   52.090    11.200      3.290     1.340      .990      .417      .327
    .107      .039      .013     .005      .002       .001      .000      .000      .000      .000
   7         8         9         0         0          0         0         0

                  -----------   MODULE # ( 4) =  FLASH       ----------
 620.800    72.300    27.100  200.100    39.300     10.800     4.220     3.070     1.220      .944
    .290      .101      .033     .012      .004       .002      .000      .000      .000      .000
   9        10        11         0         0          0         0         0

                  -----------   MODULE # ( 5) =  MIXER       ----------
    .000      .000      .000     .000      .000       .000      .000      .000      .000      .000
    .000      .000      .000     .000      .000       .000      .000      .000      .000      .000
   4        10         0         0         0          0         7         2

                  -----------   MODULE # ( 6) =  MIXER       ----------
    .000      .000      .000     .000      .000       .000      .000      .000      .000      .000
    .000      .000      .000     .000      .000       .000      .000      .000      .000      .000
   1         6         8         0         0          0         2         3
```

work and in this work are the same. The time required to evaluate the Jacobian, and the time required for one flowsheet evaluation are roughly the same in both works. However, the time required by Chen's nonlinear solver is 0.15 CPU-seconds; in this work 18 CPU-seconds, which is more than 100 times higher. Either there is a discrepancy in Chen's time units, or our implementation of the nonlinear solver is rather inefficient.

IV-2 Ammonia Plant Simulation

In this problem an ammonia production plant using the Haber process is simulated. The data required for that problem was taken from Myers and Seider (1976).

A block diagram of the process is presented in figure IV-2. The process is rather simple. The feed consists of hydrogen (H_2) and nitrogen (N_2) gas containing argon (Ar) and methane (CH_4) impurities. The reaction between hydrogen and nitrogen to obtain ammonia (NH_3) is performed using an iron catalyst at 200 atm and about $800^{o}K$:

$$N_2 + 3H_2 \rightleftarrows 2NH_3$$

The outlet from the reactor (REACTOR2) is cooled and it goes to a high pressure flash drum (FLASH3). The liquid from FLASH3 contains most of the ammonia, and the vapor contains unreacted gasses and impurities. The vapor stream from FLASH3 goes to a splitter (SPL5) where 2 percent of the vapor is purged. The liquid stream from FLASH3 is the input stream of a low pressure flash drum

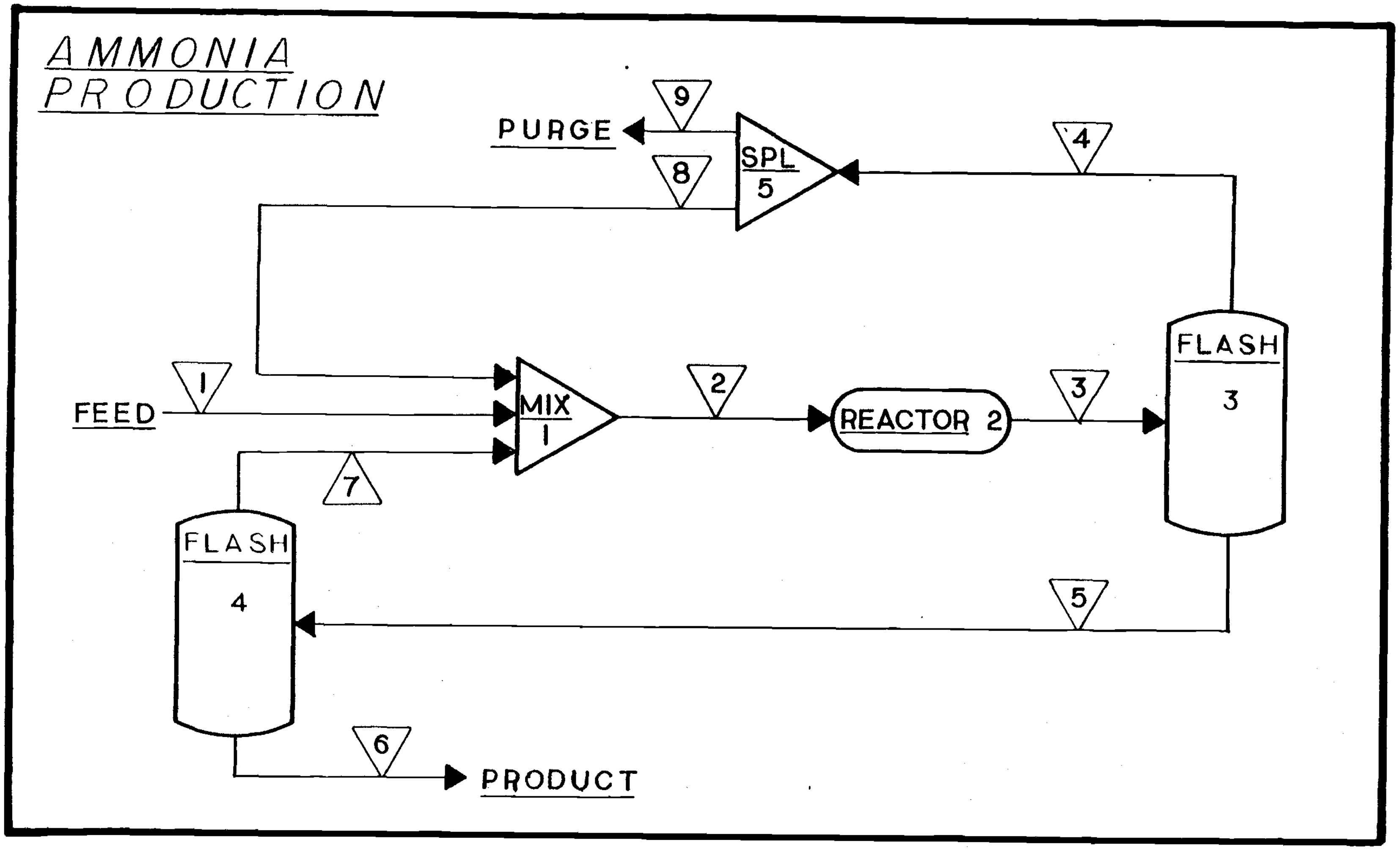


Figure IV-2. Block Diagram of the Ammonia Plant

(FLASH4). The liquid stream from FLASH4 is almost pure ammonia, the vapor stream contains a small amount of N_2, H_2 and NH_3 which are recycled.

The feed composition, as well as the vapor liquid equilibrium constants for FLASH3 and 4, are presented in table IV-6.

Table IV-6. Feed and K Values for the NH_3 Plant Problem

COMPONENT	MOLE FRACTION	MOLAR FLOW RATE	K-FLASH3	K-FLASH4
NH_3	0.0	0.0	0.06	0.28
N_2	0.24	24.0	105	2400
H_2	0.743	74.3	90	1750
Ar	0.006	0.6	100	1400
CH_4	0.011	1.1	33	500

The conversion of a reactant was not supplied, but the chemical equilibrium constant is given as k = 0.35, or

$$0.35 = (NH_3)^2/[(N_2)(H_2)^3] \qquad \text{(IV-1)}$$

In using the sequential modular approach, the conversion of a reactant must be specified, so before the reactor module is called, the conversion γ is calculated by solving a nonlinear equation derived from equation IV-1 with the interval-halving method.

In the simultaneous modular approach and equation-based approach, equation IV-1 is included in the system of nonlinear equations, and the conversion γ becomes a manipulated variable, that is, another unknown.

The results obtained for this problem are summarized in table IV-7.

Table IV-7. Results for the NH_3 Plant Problem

	ITER	FEVAL	CPU-TIME	# EQ
EQSS	4	--	8.37	48
SIMO	4	15	1.454	6
SEQ	119	119	2.54	5

Table IV-8 presents the main program used for the three approaches; table IV-9 presents the solution of the problem.

It can be seen that the simultaneous modular approach achieved convergence in a few iterations and the execution time was the lowest of the three approaches. The number of equations was very small (6 equations), so the overhead of SIMO was quite small and it did not contribute substantially to execution time. However, MPDLM used 0.2935 CPU-seconds per iteration, whereas Wegstein's subroutine spent 0.021 CPU-seconds per iteration to solve the problem using the sequential modular approach.

EQSS solved the problem in a few iterations, however execution time was rather high. The execution time required to evaluate the Jacobian was approximately 1 CPU-second, so MPDLM used about 1.8 CPU-seconds per iteration. It should be mentioned that when the same problem was again solved using MPDLM, but the linear system was solved with a full matrix technique; the execution time jumped to 45

Table IV-8. Programs Used to Simulate the NH_3 Plant Problem: SENH3 - sequential modular approach
SMNH - simultaneous modular approach
ENH - equation-based approach

```
      PROGRAM SENH3(TAPE5,OUTPUT,TAPE6)
      EXTERNAL FSIS
      COMMON /P31/ F(5),FN1,ITOT
      REAL X(5)
      ITOT=0
      CALL READ(9,5)
      NT=200
      N=5
      EPS=0 0001
      K=0
      X(1)=150
      X(2)=700
      X(3)=50
      X(4)=50
      X(5)=50
      WRITE(6,222)
  222 FORMAT(/,10X," INITIAL ESTIMATES OF TORN STREAMS AND",
     1          /,10X," EQUIPAMENT PARAMETERS:",/)
      CALL WRITES
      CALL WRITEE
      CALL WEGSMD(N,X,NT,EPS,FSIS,K)
      WRITE(6,10) ITOT
   10 FORMAT(//5X,"CONVERGENCE ACHIEVED IN ",I3," ITERATIONS",//)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(M,X,F)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      REAL X(M),F(M)
      COMMON /P31/ F1(5),FN1,ITOT
      EXTERNAL CONST
      ITOT=ITOT+1
      FLOW(2)=0
      DO 10 I=1,5
   10 FLOW(2)=FLOW(2)+X(I)
      DO 20 I=1,5
      COMP(2,I)=X(I)/FLOW(2)
   20 F1(I)=X(I)
      FN1=FLOW(2)
      YL=0
      YR=AMIN1(F1(1),F1(2)/3 )
      CALL INTHLV(YL,YR,Y,20,CONST,0)
      EQP(2,20)=Y/F1(1)
      CALL REAC(2)
      CALL FLASH(3)
      CALL FLASH(4)
      CALL SPLIT(5)
      CALL MIXER(1)
      DO 30 I=1,5
   30 F(I)=FLOW(2)*COMP(2,I)
      RETURN
      END
C
C
C
C
      FUNCTION CONST(Y)
      COMMON /P31/ F(5),FN1,ITOT
      EQK=0 35
      FT=FN1-2 *Y
      A=EQK*(F(1)-Y)*(F(2)-3 *Y)**3
      B=FT*(F(3)+2 *Y)
      CONST=A-B*B
      RETURN
      END
C
C
```

```
      PROGRAM SMNH(TAPE5,TAPE6)
      CHARACTER*4 NAME(2),NACO(2)
      INTEGER NUE(2),NUP(2),NUS(2),NCO(2)
      REAL VAL(2)
C
      CALL READ(9,5)
      N1=1
      N2=2
      N3=1
      NSIG=4
      NAME(1)='USE2'
      NUE(1)=2
      NUP(1)=20
      NACO(1)='USER'
      NUS(1)=0
      NCO(1)=0
      VAL(1)=0
C
      IP=3
      CALL SPEC(N3,NAME,NUE,NUP,NACO,NUS,NCO,VAL)
      CALL SIMSO(N1,N2,N3,IP,NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(X,F,IT,N)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      DIMENSION X(N),F(N)
      CALL REAC(2)
      F1=COMP(3,3)*COMP(3,3)
      F2=COMP(3,2)**3
      F(IT)=F1/(COMP(3,1)*F2)-0 35
      CALL FLASH(3)
      CALL FLASH(4)
      CALL SPLIT(5)
      CALL MIXER(1)
      RETURN
      END
C


C
C
C
      PROGRAM ENH(TAPE5,TAPE9,TAPE6)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      CALL READ(9,6)
      CALL SREAD(9,6,5)
      CALL REAC(2)
      CALL FLASH(3)
      CALL FLASH(4)
      CALL SPLIT(5)
      CALL MIXER(1)
      DO 10 I=1,NS
      DO 10 L=1,NC
      S=COMP(I,L)*200
      L1=NINT(S)
      COMP(I,L)=L1/200
   10 CONTINUE
      CALL SIMMAN(4,2)
      STOP
      END
C
C
C
C
      SUBROUTINE USER1(NN,N,F,IT)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      REAL F(N)
      F(IT)=COMP(3,3)*COMP(3,3)/(COMP(3,1)*COMP(3,2)**3)-0 35
      IT=IT+1
      RETURN
      END
C
```

Table IV-9. Solution of the NH_3 Plant Problem

```
STREAM  VARIABLES :
NSTR   FLOW    N2         H2        NH3        AR        CH4
1    100.00   .2400     .7430     .0000      0060      0110
2    788.58   .1735     .6674     .0520      0380      0691
3    745.20   .1545     6189      1132       0402     .0732
4    701.95   .1639     6566      0593       0427      0775
5     43.25   .0016     0073      9884       0004      0023
6       .66   .0991     .4587     2798       0266      1357
7     42.58   .0000     0003      9994       0000      0003
8    687.91   .1639     .6566     0593       0427     .0775
9     14.04   .1639     6566      0593       0427      0775

 EQUIPMENT  PARAMETERS:       (2 X EQP(I,J))/IEQP(I,J)

                 ----------     MODULE # ( 1) =  MI·XER       ----------
    .000       .000       000      .000      000      000      .000      000      000      000
     000        000       000       000      000      000       000      000      000      000
    1          8          6        0         0        0         2        3

                 ----------     MODULE # ( 2) =  REACTOR      ----------
  -1 000    -3.000      2 000      000      000      000       000      000      000      000
     000       .000       000      .000      000      000       000      000      000      159
    2          3          1        0         0        0         0.       1

                 ----------     MODULE # ( 3) =  FLASH        ----------
 105.000    90.000       .060  100.000   33 000      000       000      000      000      000
     000       000        000      000      000     .000       000      000      000      000
    3      .   4          5        0         0        0         0        0

                 ----------     MODULE # ( 4) =  FLASH        ----------
2400.0001750.000         2801400 000  500 000      000      .000      000      000      000
     000       000        000      000      000     .000       000      000      000      000
    5          6          7        0         0        0         0        0

                 ----------     MODULE # ( 5) =  SPLITTER     ----------
     000       .020       980      .000     .000      000      .000      000      000      000
     000       .000       000      .000     .000      000       000      000      000      000
    4          9          8        0         0        0         0        2
```

CPU-seconds. Thus, there was a considerable advantage in using sparse-matrix techniques in MPDLM, including SPAMAT, to solve the linear system. It is clear, however, that an even faster subroutine must be used for the equation-based approach to be competitive with the other methods.

The same problem was solved imposing a design specification on the flow rate of stream 9 (purge stream) equal to 10 mole/unit of time. The split fraction in SPL5 was manipulated to meet this design specification.

To solve the design problem with the sequential modular approach, we used subroutine SECNEW to solve the design specification equation and Wegstein's method to solve stream connection equations. In the other two approaches little had to be changed other than entering a few parameters indicating that FLOW(9) is equal to 10 and EQP(5,2) was an unknown.

The number of iterations and execution time for the design problem are presented in table IV-10.

Table IV-10. Results for the NH_3 Plant Problem With One Design Specification

	ITER	FEVAL	CPU-TIME	# EQ
EQSS	5	--	10.077	50
SIMO	15	36	2.867	7
SEQ	5	471	8.61	6

A few comments are appropriate at this time. The absurd number of sequential iterations required by the sequential modular approach is a typical result of nested nonlinear solvers. To guarantee a maximum error of 1×10^{-4} in the outer loop (SECNEW), the inner loop (Wegstein) should not have an error greater than or equal to 1×10^{-4}. A few preliminary tests showed that one order of magnitude was enough, so the convergence criteria for the inner loop was set equal to 1×10^{-5}. Table IV-11 shows how the iterations were distributed.

Table IV-11. Sequential Iterations Required by SECNEW

SECNEW Iteration	Wegstein's Iterations
1	154
1a (derivative)	9
2	110
3	133
4	40
5	34
TOTAL	471

Wegstein's methods require a set of initial guesses at each iteration of SECNEW. For the first iteration, the user provides the set of initial estimates. The following iterations of SECNEW use the converged values of the torn streams variables of the previous iteration. For instance, the initial guesses for the stream variables for the third iteration of SECNEW used the converged values of stream variables of the second iteration of SECNEW.

SIMO solved the problem in 15 iterations and it took 2.87 CPU-seconds for solution, 3 times faster than the sequential modular and almost 4 times faster than the equation based approach. Again, the equation based approach converged fast, but the execution time was too high.

The solution of the problem is presented in table IV-12, and the main programs in table IV-13.

IV-3 Ethanal Production

The dehydrogenation of ethyl alcohol to produce ethanal is a process used since the beginning of the century. Later, when petroleum feedstock became cheaper and abundant, ethylene was used as raw material. After the petroleum crises of the 1970's, dehydrogenation of ethanol became attractive to some countries where ethanol could be produced cheaply. Several countries, for example Brazil and India, use the process today.

The dehydrogenation of ethanol is performed over a copper catalyst at about 1 atm and $600^{o}K$.

$$CH_3CH_2OH \rightleftarrows CH_3CHO + H_2$$

Unfortunately, there are several other reactions in series and in parallel occuring in the reactor. Besides ethanal and hydrogen, acetic acid, ethyl acetate, higher aldehydes and alchohols (4 or more carbon atoms) and gases (CO, CO_2, C_3H_8) are also obtained, in

Table IV-12. Solution of the NH_3 Plant Problem with One Design Specification

```
STREAM VARIABLES
NSTR   FLOW    N2       H2      NH3      AR      CH4
1   100.00   .2400   .7430   .0000   .0060   .0110
2  1054.05   .1456   .6466   .0538   .0548   .0992
3  1008.77   .1297   .6083   .1011   .0572   .1036
4   963.32   .1358   .6367   .0593   .0599   .1084
5    45.45   .0013   .0071   .9877   .0006   .0033
6      .73   .0783   .4245   .2798   .0356   .1817
7    44.72   .0000   .0002   .9993   .0000   .0004
8   953.32   .1358   .6367   .0593   .0599   .1084
9    10.00   .1358   .6367   .0593   .0599   .1084

EQUIPMENT PARAMETERS:          (2 X EQP(I,J))/IEQP(I,J)

                   -----------   MODULE # (  1) =  MIXER      -----------
   .000     .000     .000      .000     .000     .000     .000     .000     .000     .000
   .000     .000     .000      .000     .000     .000     .000     .000     .000     .000
   1        8        6         0        0        0        2        3

                   -----------   MODULE # (  2) =  REACTOR    -----------
 -1.000   -3.000    2.000      .000     .000     .000     .000     000      .000     .000
   .000     .000     .000      .000     .000     .000     .000     .000     000      .147
   2        3        1         0        0        0        0        1

                   -----------   MODULE # (  3) =  FLASH      -----------
105.000   90.000     .060   100.000   33.000     .000     .000     .000     .000     .000
   .000     .000     .000      .000     .000     .000     .000     .000     .000     .000
   3        4        5         0        0        0        0        0

                   -----------   MODULE # (  4) =  FLASH      -----------
2400.0001750.000     .2801400.000  500.000     .000     .000     .000     000      .000
   .000     .000     .000      .000     .000     .000     .000     .000     000      .000
   5        6        7         0        0        0        0        0

                   -----------   MODULE # (  5) =  SPLITTER   -----------
   .000     .010     .990      .000     .000     .000     .000     .000     000      .000
   .000     .000     .000      .000     .000     .000     .000     .000     000      .000
C  4        9        8         0        0        0        0        2
C
```

Table IV-13. Programs Used to Simulate the NH_3 Plant Problem With One Design Specification:
SC1NH - sequential modular approach
SMC2NH - simultaneous modular approach

```
C
      PROGRAM SC1NH(TAPE5,TAPE6)
      EXTERNAL SUB
      COMMON /P31/ FI(5),FN1,ITOT
      ITOT=0
      EPS=1 E-4
      NT=50
      X=0 02
      CALL READ(9 5)
      WRITE(6,222)
  222 FORMAT(/,10X," INITIAL ESTIMATES OF TORN STREAMS AND",
     1          /,10X," EQUIPAMENT PARAMETERS ",/)
      CALL WRITES
      CALL WRITEE
      CALL SECNEW(X,NT,EPS,SUB)
      WRITE(6,10) NT,ITOT
   10 FORMAT(//,5X,"CONVERGENCE ACHIEVED IN ",I4," ITERATIONS",
     1          /,5X,"NUMBER OF FLOWSHEET EVALUATIONS= ",I4,//)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE SUB(X,F)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      REAL Y(5)
      EXTERNAL FSIS
      NT=300
      N=5
      EPS=1 E-5
      K=0
      DO 10 I=1,5
   10 Y(I)=FLOW(2)*COMP(2,I)
      EQP(5,2)=X
      EQP(5,3)=1 -X
      CALL WEGSMD(N,Y,NT,EPS,FSIS,K)
      F=FLOW(9)-10 0
      RETURN
      END
C
C
C
C
      SUBROUTINE FSIS(M,X,F)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      REAL X(M),F(M)
      COMMON /P31/ FI(5),FN1,ITOT
      EXTERNAL CONST
      ITOT=ITOT+1
      FLOW(2)=0
      DO 10 I=1,5
   10 FLOW(2)=FLOW(2)+X(I)
      DO 20 I=1,5
      COMP(2,I)=X(I)/FLOW(2)
   20 FI(I)=X(I)
      FN1=FLOW(2)
      YL=0
      YR=AMIN1(FI(1),FI(2)/3 )
      CALL INTHLV(YL,YR,Y,20,CONST,0)
      EQP(2,20)=Y/FI(1)
      CALL REAC(2)
      CALL FLASH(3)
      CALL FLASH(4)
      CALL SPLIT(5)
      CALL MIXER(1)
      DO 30 I=1,5
   30 F(I)=FLOW(2)*COMP(2,I)
      RETURN
      END
C
```

```
C
C
C
C
      FUNCTION CONST(Y)
      COMMON /P31/ F(5),FN1,ITOT
      EQK=0 35
      FT=FN1-2 *Y
      A=EQK*(F(1)-Y)*(F(2)-3 *Y)**3
      B=FT*(F(3)+2 *Y)
      CONST=A-B*B
      RETURN
      END
C
C
C
      PROGRAM SMC2NH(TAPE5,TAPE6)
      CHARACTER*4 NAME(2),NACO(2)
      INTEGER NUE(2),NUP(2),NUS(2),NCO(2)
      REAL VAL(2)
C
      CALL READ(9,5)
      N1=1
      N2=2
      N3=2
      NSIG=4
      NAME(1)='USE2'
      NUE(1)=2
      NUP(1)=20
      NACO(1)='USER'
      NUS(1)=0
      NCO(1)=0
      VAL(1)=0
C
      NAME(2)='SPLIT'
      NUE(2)=5
      NUP(2)=2
      NACO(2)='FLOW'
      NUS(2)=9
      NCO(2)=0
      VAL(2)=10
      IP=3
      CALL SPEC(N3,NAME,NUE,NUP,NACO,NUS,NCO,VAL)
      CALL SIMSO(N1,N2,N3,IP,NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(X,F,IT,N)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      DIMENSION X(N),F(N)
      CALL REAC(2)
      F1=COMP(3,3)*COMP(3,3)
      F2=COMP(3,2)**3
      F(IT)=F1/(COMP(3,1)*F2)-0 35
      CALL FLASH(3)
      CALL FLASH(4)
      CALL SPLIT(5)
      CALL MIXER(1)
      RETURN
      END
C
```

small quantities. Some of the products, even at small quantities, have an attractive commercial value.

The entire process is presented in figure IV-3. Although the dehydrogenation of ethanol is performed in a single reactor, we could not simulate the complex series of reactions in a single reactor module. We used three reactors in series, each of them performing the indicated reactions:

<u>First Stage</u>

$$\text{ETHOL} \rightleftarrows \text{ETHAL} + H_2$$

ETHOL = ethanol
ETHAL = ethanal

<u>Second Stage</u>

$$\text{ETHAL} + H_2O \rightleftarrows \text{EE} + H_2$$

EE = acetic acid

<u>Third Stage</u>

$$4.66\ \text{ETHOL} + 0.508\ \text{ETHAL} \rightleftarrows .278\ H_2O + 1.534\ \text{EEE} + 0.26 + 1\ \text{ORG}$$

EEE = ethyl acetate

ORG = higher alcohols and aldeydes

These are not the true stoichiometric coefficents; they will be used in this work because there is no precise information available about the reaction.

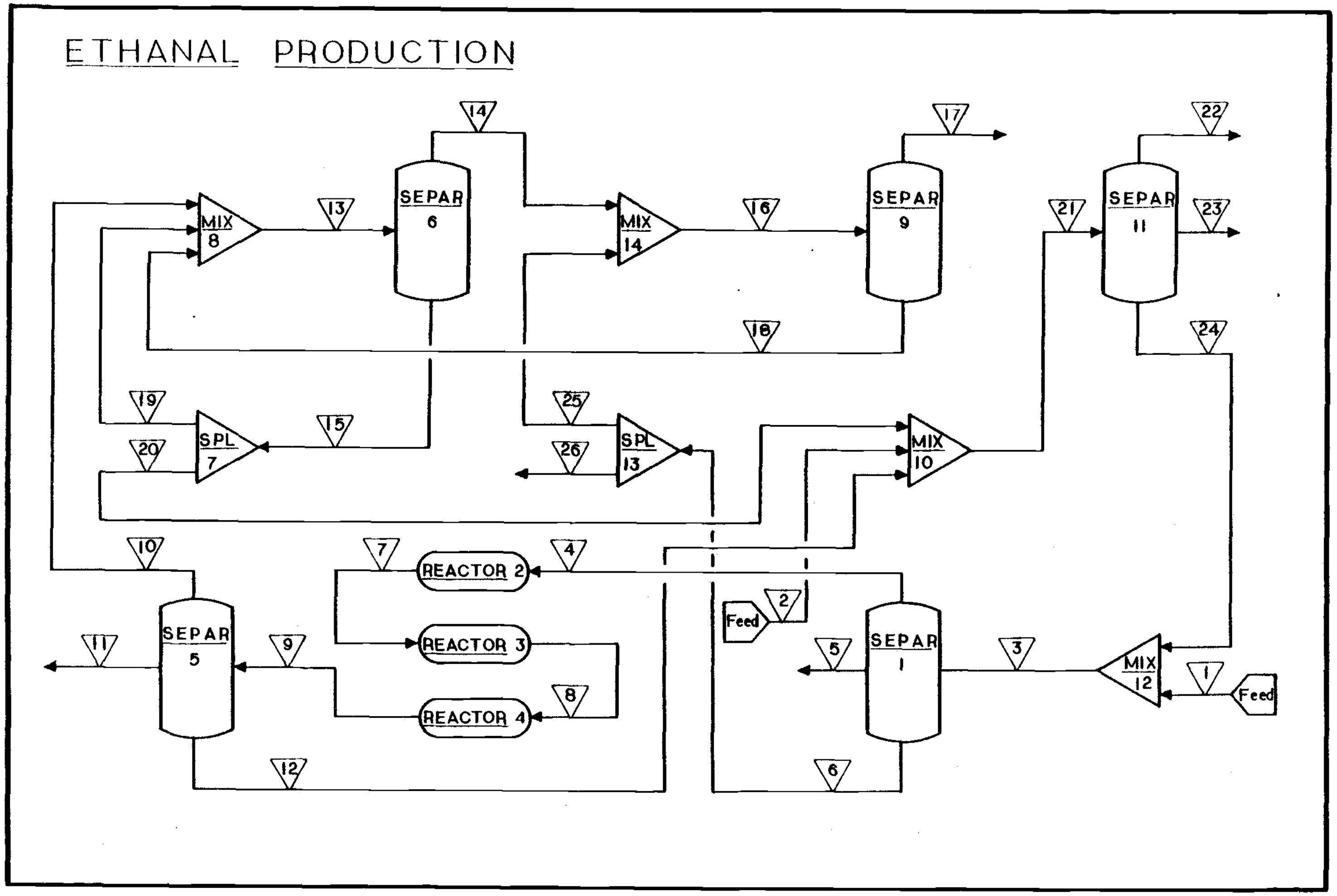


Figure IV-3. Block Diagram of the Ethanal Plant

A few comments about the process should be made. The conversion of ethanal is about 33 percent per pass. Separator 5 (SEPAR5) performs a rough separation of the components: stream 11 contains acetic acid and water, stream 10 contains gases (H_2, CO_2, CO, C_3H_3), ethanol and ethanal. The system containing MIX8 and 14, SEPAR6 and 9 and SPL7 have the sole purpose of separating ethanal from hydrogen. The separation is quite difficult as ethanal has a low boiling point (21^oC at 1 atm). Separator 11 has three outlet streams: stream 22 as pure ethanal, stream 23 as ethyl acetate, and stream 24 with ethanol and water. Separator 1 removes the excess of water and a few impurities.

Although the block diagram of figure IV-3 is a simplification of the actual process, we could not solve the simulation of the process using the equation-based approach. The nonlinear subroutine was set up to a maximum of 200 simultaneous equations and this problem requires the solution of 216 equations. Unfortunately, even modifying all subroutines to accept 216 equations, we could not simulate the ethanal plant because we would reach the maximum allowed computer memory allocation for the type of account we had. As an alternative we decided to simulate this process for three different conditions: with no constraints, with one constraint, and with one constraint and two new equipment modules. The simulations were then solved with both the sequential and simultaneous modular approaches.

The execution time and iterations required for both methods in the case of the simple simulation (no design specifications) are summarized in table IV-14.

Table IV-14. Results for the Ethanal Plant Problem

	ITER	FEVAL	CPU-TIME	# EQ
SIMO	2	22	2.422	16
SEQ	111	111	4.47	16

For this problem the tear streams were streams 3 and 13. The main programs may be found in table IV-15, and the solution of the problem in table IV-16.

It is easy to see that the simultaneous modular approach was far superior to the sequential modular approach. The execution time was almost halfed using SIMO, in addition only two iterations were required to reach a solution.

The same problem was solved imposing as a design specification the flow rate of the product equal to 65 (mole/unit of time) and specifying the conversion of Reactor 2 as a manipulated variable.

Table IV-17 summarizes the execution time and iterations required to solve the design problem.

Table IV-15. Main Programs Used to Simulate the Ethanal Plant Problem:
SEN - sequential modular approach
SMEN - simultaneous modular approach

```
      PROGRAM SEN(TAPE5,TAPE6)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      REAL  X(16)
      EXTERNAL FSIS
      COMMON /P31/ ITOT
      CALL READ(26,14)
      WRITE(6,222)
  222 FORMAT(/,10X," INITIAL ESTIMATES OF TORN STREAMS AND",
     1           /,10X,"EQUIPAMENT PARAMETERS",/)
      CALL WRITES
      CALL WRITEE
      ITOT=0
      N=16
      NT=200
      EPS=1 E-4
      K=0
      DO 20 I=1,8
      X(I)=FLOW(3)*COMP(3,I)
   20 X(I+8)=FLOW(13)*COMP(13,I)
      CALL WEGSMD(N,X,NT,EPS,FSIS,K)
      WRITE(6,10) ITOT
   10 FORMAT(//,5X,"CONVERGENCE ACHIEVED IN ",I4,"  ITERATIONS",/)
      CALL WRITES
      CALL WRITEE
      STOP
      ENO
C
C
C
C
      SUBROUTINE FSIS(M,X,F)
      REAL X(M),F(M)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      COMMON /P31/ ITOT
      ITOT=ITOT+1
      FLOW(3)=0
      FLOW(13)=0
      DO 10 I=1,8
      FLOW(3)=FLOW(3)+X(I)
   10 FLOW(13)=FLOW(13)+X(I+8)
      DO 20 I=1,8
      COMP(3,I)=X(I)/FLOW(3)
   20 COMP(13,I)=X(I+8)/FLOW(13)
      CALL SEPAR(1)
      CALL REAC(2)
      CALL REAC(3)
      CALL REAC(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL SPLIT(13)
      CALL MIXER(14)
      CALL SEPAR(9)
      CALL SPLIT(7)
      CALL MIXER(8)
      CALL MIXER(10)
      CALL SEPAR(11)
      CALL MIXER(12)
      IF(ITOT EQ 1) THEN
      CALL WRITES
      CALL WRITEE
      ELSE
      ENDIF
      DO 30 I=1,8
      F(I)=FLOW(3)*COMP(3,I)
   30 F(I+8)=FLOW(13)*COMP(13,I)
      RETURN
      END
```

```
      PROGRAM SMEN(TAPE5,TAPE6,OUTPUT=TAPE6)
      REAL VAL(3)
      INTEGER NUE(3),NUP(3),NUS(3),NCO(3),N2(3)
      CHARACTER*4,NAME(3),NACO(3)
      CALL READ(26,14)
      N1=2
      N2(1)=3
      N2(2)=13
      N3=0
      IP=3
      NSIG=4
      CALL SIMSO(N1,N2,N3,IP,NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
      SUBROUTINE FSIS(X,F,IT,N)
      REAL X(N),F(N)
      CALL SEPAR(1)
      CALL REAC(2)
      CALL REAC(3)
      CALL REAC(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL SPLIT(13)
      CALL MIXER(14)
      CALL SEPAR(9)
      CALL SPLIT(7)
      CALL MIXER(8)
      CALL MIXER(10)
      CALL SEPAR(11)
      CALL MIXER(12)
      RETURN
      ENO
```

Table IV-16. Solution of the Ethanal Plant Problem

NSTR	FLOW	ETHOL	ETHAL	H2O	EE	EEE	H2	G	ORG
1	263.77	.2616	.0000	.7384	.0000	.0000	.0000	.0000	.0000
2	39.18	.0266	.0383	.9059	.0000	.0183	.0000	.0000	.0109
3	1114.96	.2063	.0000	.7929	.0004	.0000	.0000	.0000	.0003
4	278.36	.8264	.0000	.1736	.0000	.0000	.0000	.0000	.0000
5	2.86	.0000	.0000	.8713	.0000	.0000	.0000	.0000	.1287
6	833.74	.0000	.0000	.9994	.0006	.0000	.0000	.0000	.0000
7	342.70	.4835	.1877	.1410	.0000	.0000	.1877	.0000	.0000
8	342.70	.4835	.1829	.1362	.0049	.0000	.1926	.0000	.0000
9	340.53	.4728	.1826	.1378	.0049	.0045	.1938	.0006	.0029
10	120.19	.0543	.3949	.0000	.0000	.0000	.5492	.0017	.0000
11	2.09	.0000	.0000	.2658	.7342	.0000	.0000	.0000	.0000
12	218.26	.7079	.0674	.2125	.0006	.0070	.0000	.0000	.0046
13	6034.82	.0097	.0724	.9064	.0005	.0000	.0109	.0000	.0000
14	78.13	.0000	.1526	.0000	.0000	.0000	.8448	.0026	.0000
15	5956.69	.0098	.0713	.9183	.0006	.0000	.0000	.0000	.0000
16	689.51	.0000	.0173	.8862	.0005	.0000	.0957	.0003	.0000
17	66.21	.0000	.0000	.0000	.0000	.0000	.9970	.0030	.0000
18	623.30	.0000	.0191	.9803	.0006	.0000	.0000	.0000	.0000
19	5291.33	.0098	.0713	.9183	.0006	.0000	.0000	.0000	.0000
20	665.36	.0098	.0713	.9183	.0006	.0000	.0000	.0000	.0000
21	922.80	.1756	.0690	.7509	.0005	.0024	.0000	.0000	.0015
22	62.16	.0000	1.0000	.0000	.0000	.0000	.0000	.0000	.0000
23	9.45	.1097	.1589	.3811	.0000	.2382	.0000	.0000	.1120
24	851.19	.1892	.0000	.8098	.0006	.0000	.0000	.0000	.0004
25	611.38	.0000	.0000	.9994	.0006	.0000	.0000	.0000	.0000
26	222.36	.0000	.0000	.9994	.0006	.0000	.0000	.0000	.0000

```
EQUIPMENT PARAMETERS:        (2 X EQP(I,J))/IEQP(I,J)

                     ----------      MODULE # ( 1) =  SEPAR     ----------
   1.000     .000      .055      .000     .000     .000     .000     .000     .000     .000
    .000     .000      .943     1.000     .000     .000     .000     .000     .000     .000
    3        4         6         5        0        0        0        3

                     ----------      MODULE # ( 2) =  REAC      ----------
  -1.000    1.000      .000      .000     .000    1.000     .000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .280
    4        7         0         0        0        0        0        1

                     ----------      MODULE # ( 3) =  REAC      ----------
    .000   -1.000    -1.000     1.000     .000    1.000     .000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .026
    7        8         0         0        0        0        0        2

                     ----------      MODULE # ( 4) =  REAC      ----------
  -4.688    -.508      .278      .000    1.534     .000     .200    1.000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .028
    8        9         0         0        0        0        0        1

                     ----------      MODULE # ( 5) =  SEPAR     ----------
    .041     .763      .000      .000     .000    1.000    1.000     .000     .000     .000
    .959     .237      .988      .080    1.000     .000     .000    1.000     .000     .000
    9        10        12        11       0        0        0        3

                     ----------      MODULE # ( 6) =  SEPAR     ----------
    .000     .027      .000      .000     .000    1.000    1.000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    13       14        15        0        0        0        0        0

                     ----------      MODULE # ( 7) =  SPLIT     ----------
    .000     .888      .112      .000     .000     .000     .000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    15       19        20        0        0        0        0        2

                     ----------      MODULE # ( 8) =  MIXER     ----------
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    10       18        19        0        0        0        13       3

                     ----------      MODULE # ( 9) =  SEPAR     ----------
    .000     .000      .000      .000     .000    1.000    1.000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    16       17        18        0        0        0        0        0

                     ----------      MODULE # (10) =  MIXER     ----------
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    2        12        20        0        0        0`       21       3

                     ----------      MODULE # (11) =  SEPAR     ----------
    .000     .976      .000      .000     .000     .000     .000     .000     .000     .000
    .994     .000      .995     1.000     .000     .000     .000     .258     .000     .000
  21       22        24        23        0        0        0        3

                     ----------      MODULE·# (12) =  MIXER     ----------
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    1        24        0         0        0        0        3        2

                     ----------      MODULE # (13) =  SPLIT     ----------
    .000     .733      .267      .000     .000     .000     .000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    6        25        26        0        0        0        0        2

                     ----------      MODULE # (14) =  MIXER     ----------
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    .000     .000      .000      .000     .000     .000     .000     .000     .000     .000
    14       25        0         0        0        0        16       2
```

Table IV-17. Results for the Ethanal Plant Problem with One Design Specification

	ITER	FEVAL	CPU-TIME	# EQ
SIMO	3	25	4.523	17
SEQ	7	435	12.983	17

Again the simultaneous modular approach was far better than the sequential modular approach. The execution time with SIMO was almost one-third that when using SIMO. The number of sequential iterations is only about 5 percent of the number with SEQ.

The main programs are presented in tables IV-18 and the solution is presented in table IV-19.

From table IV-19 it can be seen that the recycle stream 19 has a rather high flow rate. To reduce the flow rate, we will introduce two flash drums in series after separator 5. The idea is to reduce the amount of ethanal entering the separation area (SEPAR6 and 9). We will maintain the separation factors and we will impose the concentration of ethanal in stream 18 to remain the same, that is, COMP(18,2) = 0.0191. We will let the split fraction of splitter 13 be a manipulated variable to meet the design specification. The vapor-liquid equilibrium constants are shown in table IV-20, as well as the main program for the simultaneous modular approach. The solution of the problem may be found in table IV-21.

Again, the simultaneous modular approach was far better than the sequential modular approach. The execution time was about

Table IV-18. Main Program Used to Simulate the Ethanal Plant Problem with One Design Specification:
SC1EN - sequential modular approach
SMC1EN - simultaneous modular approach

```
C
      PROGRAM SC1EN(TAPE5,TAPE6)
      COMMON /P31/ ITOT
      EXTERNAL SUB
      ITOT=0
      EPS=1.E-4
      NT=50
      X= 30
      CALL READ(26,14)
C
      WRITE(6,222)
  222 FORMAT(/,10X," INITIAL ESTIMATES OF TORN STREAMS AND",
     1         /,10X," EQUIPAMENT PARAMETERS ",/)
      CALL WRITES
      CALL WRITEE
      CALL SECNEW(X,NT,EPS,SUB)
      WRITE(6,10) NI,ITOT
   10 FORMAT(//,5X,"CONVERGENCE ACHIEVED IN ",I4," ITERATIONS",
     1         /,5X,"NUMBER OF FLOWSHEET EVALUATIONS= ",I4,//)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE SUB(X,F)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      REAL Y(16)
      EXTERNAL FSIS
      DO 10 I=1,8
      Y(I)=FLOW(3)*COMP(3,I)
   10 Y(I+8)=FLOW(13)*COMP(13,I)
      N=16
      NT=300
      EPS=1.E-5
      K=0
      EQP(2,20)=X
      CALL WEGSMO(N,Y,NT,EPS,FSIS,K)
      F=FLOW(22)-65.0
      RETURN
      END
C
C
C
C
      SUBROUTINE FSIS(M,X,F)
      REAL X(M),F(M)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      COMMON /P31/ ITOT
      ITOT=ITOT+1
      FLOW(3)=0
      FLOW(13)=0
      DO 10 I=1,8
      FLOW(3)=FLOW(3)+X(I)
   10 FLOW(13)=FLOW(13)+X(I+8)
      DO 20 I=1,8
      COMP(3,I)=X(I)/FLOW(3)
   20 COMP(13,I)=X(I+8)/FLOW(13)
      CALL SEPAR(1)
      CALL REAC(2)
      CALL REAC(3)
      CALL REAC(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL SPLIT(13)
      CALL MIXER(14)
      CALL SEPAR(9)
      CALL SPLIT(7)
      CALL MIXER(8)
      CALL MIXER(10)
      CALL SEPAR(11)
      CALL MIXER(12)
      DO 30 I=1,8
      F(I)=FLOW(3)*COMP(3,I)
   30 F(I+8)=FLOW(13)*COMP(13,I)
      RETURN
      END
C
C
C
```

```
      PROGRAM SMC1EN(TAPE5,TAPE6,OUTPUT=TAPE6)
      REAL VAL(3)
      INTEGER NUE(3),NUP(3),NUS(3),NCO(3),N2(3)
      CHARACTER*4,NAME(3),NACO(3)
      CALL READ(26,14)
      N1=2
      N2(1)=3
      N2(2)=13
      N3=1
      NAME(1)='REAC'
      NUE(1)=2
      NUP(1)=20
      NACO(1)='FLOW'
      NUS(1)=22
      NCO(1)=0
      VAL(1)=65
      IP=3
      NSIG=4
      CALL SPEC(N3,NAME,NUE,NUP,NACO,NUS,NCO,VAL)
      CALL SIMSO(N1,N2,N3,IP,NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
      SUBROUTINE FSIS(X,F,IT,N)
      REAL X(N),F(N)
      CALL SEPAR(1)
      CALL REAC(2)
      CALL REAC(3)
      CALL REAC(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL SPLIT(13)
      CALL MIXER(14)
      CALL SEPAR(9)
      CALL SPLIT(7)
      CALL MIXER(8)
      CALL MIXER(10)
      CALL SEPAR(11)
      CALL MIXER(12)
      RETURN
      END
```

Table IV-19. Solution of the Ethanal Plant Problem With One Design Specification

```
 STREAM  VARIABLES
 NSTR   FLOW    ETHOL    ETHAL     H2O      EE       EEE      H2        G        ORG
 1    263.77   .2616    .0000    .7384    .0000    .0000    .0000    .0000    .0000
 2     39.18   .0266    .0383    .9059    .0000    .0183    .0000    .0000    .0109
 3   1037.36   .1478    .0000    .8515    .0005    .0000    .0000    .0000    .0002
 4    201.60   .7605    .0000    .2395    .0000    .0000    .0000    .0000    .0000
 5      2.74   .0000    .0000    .9106    .0000    .0000    .0000    .0000    .0894
 6    833.03   .0000    .0000    .9994    .0006    .0000    .0000    .0000    .0000
 7    268.67   .3210    .2496    .1797    .0000    .0000    .2496    .0000    .0000
 8    268.67   .3210    .2432    .1733    .0065    .0000    .2561    .0000    .0000
 9    267.54   .3132    .2432    .1745    .0065    .0030    .2572    .0004    .0019
10    121.99   .0278    .4072    .0000    .0000    .0000    .5641    .0009    .0000
11      2.15   .0000    .0000    .2568    .7432    .0000    .0000    .0000    .0000
12    143.41   .5607    .1073    .3218    .0010    .0056    .0000    .0000    .0036
13   6025.29   .0050    .0759    .9071    .0006    .0000    .0114    .0000    .0000
14     81.40   .0000    .1533    .0000    .0000    .0000    .8454    .0013    .0000
15   5943.89   .0051    .0748    .9195    .0006    .0000    .0000    .0000    .0000
16    692.26   .0000    .0180    .8819    .0006    .0000    .0994    .0002    .0000
17     68.91   .0000    .0000    .0000    .0000    .0000    .9985    .0015    .0000
18    623.34   .0000    .0200    .9794    .0006    .0000    .0000    .0000    .0000
19   5279.96   .0051    .0748    .9195    .0006    .0000    .0000    .0000    .0000
20    663.93   .0051    .0748    .9195    .0006    .0000    .0000    .0000    .0000
21    848.52   .1002    .0786    .8176    .0006    .0018    .0000    .0000    .0011
22     65.00   .0000   1.0000    .0000    .0000    .0000    .0000    .0000    .0000
23      7.93   .0685    .1981    .4537    .0000    .1911    .0000    .0000    .0886
24    773.59   .1090    .0000    .8900    .0007    .0000    .0000    .0000    .0003
25    610.86   .0000    .0000    .9994    .0006    .0000    .0000    .0000    .0000
26    222.17   .0000    .0000    .9994    .0006    .0000    .0000    .0000    .0000

  EQUIPMENT  PARAMETERS        (2 X EQP(I,J))/IEQP(I,J)

                  ----------   MODULE # ( 1) =  SEPAR      ----------
  1.000     .000      .055    .000     .000     .000     .000     .000      .000     .000
   .000     .000      .943   1.000     .000     .000     .000     .000      .000     .000
  3        4         6        5        0        0        0        3

                  ----------   MODULE # ( 2) =  REAC       ----------
 -1.000    1.000      .000    .000     .000    1.000     .000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .438
  4        7         0        0        0        0        0        1

                  ----------   MODULE # ( 3) =  REAC       ----------
   .000   -1.000    -1.000   1.000     .000    1.000     .000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .026
  7        8         0        0        0        0        0        2

                  ----------   MODULE # ( 4) =  REAC       ----------
 -4.666    -.508      .278    .000    1.534     .000     .200    1.000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .028
  8        9         0        0        0        0        0        1

                  ----------   MODULE # ( 5) =  SEPAR      ----------
   .041     .763      .000    .000     .000    1.000    1.000     .000      .000     .000
   .959     .237      .988    .080    1.000     .000     .000    1.000      .000     .000
  9       10        12       11        0        0        0        3

                  ----------   MODULE # ( 6) =  SEPAR      ----------
   .000     .027      .000    .000     .000    1.000    1.000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
 13       14        15        0        0        0        0        0

                  ----------   MODULE # ( 7) =  SPLIT      ----------
   .000     .888      .112    .000     .000     .000     .000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
 15       19        20        0        0        0        0        2

                  ----------   MODULE # ( 8) =  MIXER      ----------
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
 10       18        19        0        0        0       13        3

                  ----------   MODULE # ( 9) =  SEPAR      ----------
   .000     .000      .000    .000     .000    1.000    1.000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
 16       17        18        0        0        0        0        0

                  ----------   MODULE # (10) =  MIXER      ----------
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
  2       12        20        0        0        0       21        3

                  ----------   MODULE # (11) =  SEPAR      ----------
   .000     .976      .000    .000     .000     .000     .000     .000      .000     .000
   .994     .000      .995   1.000     .000     .000     .000     .258      .000     .000
 21       22        24       23        0        0        0        3

                  ----------   MODULE # (12) =  MIXER      ----------
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
  1       24         0        0        0        0        3        2

                  ----------   MODULE # (13) =  SPLIT      ----------
   .000     .733      .267    .000     .000     .000     .000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
  6       25        26        0        0        0        0        2

                  ----------   MODULE # (14) =  MIXER      ----------
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
   .000     .000      .000    .000     .000     .000     .000     .000      .000     .000
 14       25         0        0        0        0       16        2
```

Table IV-20. K-Values and Main Program Used to Simulate the Modified Ethanal Plant Problem

COMPONENT	K-FLASH(15)	K-FLASH(16)
ETHOL	0.002	0.100
ETHAL	0.050	1.100
H_2	140.000	820.000
G	4.000	43.000

```
      PROGRAM ENFLCO(TAPE5,TAPE6)
      CHARACTER*4,NAME,NACO
      INTEGER N2(2)
      CALL READ(30,16)
      N1=2
      N2(1)=3
      N2(2)=13
      N3=1
      IP=3
      NSIG=4
      NAME='SPLI'
      NUE=13
      NUP=2
      NACO='COMP'
      NUS=18
      NCO=2
      VAL=0.0191
      CALL SPEC(N3,NAME,NUE,NUP,NACO,NUS,NCO,VAL)
      CALL SIMSO(N1,N2,N3,IP,NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(X,F,IT,N)
      REAL X(N),F(N)
      CALL SEPAR(1)
      CALL REAC(2)
      CALL REAC(3)
      CALL REAC(4)
      CALL SEPAR(5)
      CALL FLASH(15)
      CALL FLASH(16)
      CALL SEPAR(6)
      CALL SPLIT(13)
      CALL MIXER(14)
      CALL SEPAR(9)
      CALL SPLIT(7)
      CALL MIXER(8)
      CALL MIXER(10)
      CALL SEPAR(11)
      CALL MIXER(12)
      RETURN
      END
```

Table IV-21. Solution of the Modified Ethanal Plant Problem

```
STREAM  VARIABLES
NSTR    FLOW    ETHOL   ETHAL   H2O     EE      EEE     H2      G       ORG
 1      263 77  2616    0000    7384    0000    0000    0000    0000    0000
 2       39 18  0266    0383    9059    0000    0183    0000    0000    0109
 3      613 75  3748    0000    6243    0003    0000    0000    0000    0006
 4      250 97  9165    0000    0835    0000    0000    0000    0000    0000
 5        1 45  0000    0000    7457    0000    0000    0000    0000    2543
 6      361 33  0000    0000    9994    0006    0000    0000    0000    0000
 7      315 31  5255    2041    0664    0000    0000    2041    0000    0000
 8      315 31  5255    1988    0611    0053    0000    2083    0000    0000
 9      313 15  5142    1985    0625    0053    0049    2108    0006    0032
10      120 19  0543    3949    0000    0000    0000    5492    0017    0000
11        1 77  0000    0000    1311    8689    0000    0000    0000    0000
12      191 19  8080    0769    1011    0007    0080    0000    0000    0052
13     1368 62  0009    0702    8801    0005    0000    0482    0001    0000
14       58 82  0000    0381    0000    0000    0000    9591    0029    0000
15     1299 80  0009    0719    9267    0005    0000    0000    0000    0000
16      203 44  0000    0129    6613    0004    0000    3244    0010    0000
17       66 20  0000    0000    0000    0000    0000    9970    0030    0000
18      137 24  0000    0191    9803    0006    0000    0000    0000    0000
19     1154 61  0009    0719    9267    0005    0000    0000    0000    0000
20      145 19  0009    0719    9267    0005    0000    0000    0000    0000
21      418 99  3868    1520    4520    0005    0054    0000    0000    0034
22       62 16  0000    0000    0000    0000    0000    0000    0000    0000
23        6 84  1516    2196    1439    0000    3291    0003    0005    1548
24      349 98  4601    0000    5383    0006    0000    0000    0000    0011
25      134 62  0000    0000    9994    0006    0000    0000    0000    0000
26      226 71  0000    0000    9994    0006    0000    0000    0000    0000
27       68 82  0003    0433    0000    0000    0000    9540    0024    0000
28       51 37  1266    8659    0000    0000    0000    0068    0005    0000
29        7 95  0147    9380    0000    0000    0000    0438    0035    0000
30       43 42  1471    8527    0000    0000    0000    0001    0001    0000

EQUIPMENT  PARAMETERS                 ( 2  X  EQP( I   J ) ) / IEQP( I  J )

                   -----------   MODULE #  (  1) =   SEPAR        -----------
  1 000      000        055       000       000       000       000       000       000       000
    000      000        943     1 000       000       000       000       000       000       000
   3        4          6        5         0         0         0         3

                   -----------   MODULE #  (  2) =   REAC         -----------
 -1 000    1 000        000       000       000     1 000       000       000       000       000
    000      000        000       000       000       000       000       000       000       280
   4        7          0        0         0         0         0         1

                   -----------   MODULE #  (  3) =   REAC         -----------
    000   -1 000     -1 000     1 000       000     1 000       000       000       000       000
    000      000        000       000       000       000       000       000       000       026
   7        8          0        0         0         0         0         2

                   -----------   MODULE #  (  4) =   REAC         -----------
 -4.866   - 508         278       000     1 534       000       200     1 000       000       000
    000      000        000       000       000       000       000       000       000       028
   8        9          0        0         0         0         0         1

                   -----------   MODULE #  (  5) =   SEPAR        -----------
    041      763        000       000       000     ! 000     1 000       000       000       000
    959      237        988       080     1 000       000       000     ! 000       000       000
   9        10         12       11        0         0         0         3

                   -----------   MODULE #  (  6) =   SEPAR        -----------
    000      027        000       000       000     1 000     1 000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   13       14         15       0         0         0         0         0

                   -----------   MODULE #  (  7) =   SPLIT        -----------
    000      888        112       000       000       000       000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   15       19         20       0         0         0         0         2

                   -----------   MODULE #  (  8) =   MIXER        -----------
    000      000        000       000       000       000       000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   27       18         19       29        0         0         13        4

                   -----------   MODULE #  (  9) =   SEPAR        -----------
    000      000        000       000       000     1 000     1 000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   16       17         16       0         0         0         0         0

                   -----------   MODULE #  ( 10) =   MIXER        -----------
    000      000        000       000       000       000       000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   2        12         20       30        0         0         21        4

                   -----------   MODULE #  ( 11) =   SEPAR        -----------
    000      976        000       000       000       000       000       000       000       000
    994      000        995     1 000       000       000       000       258       000       000
   21       22         24       23        0         0         0         3

                   -----------   MODULE #  ( 12) =   MIXER        -----------
    000      000        000       000       000       000       000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   1        24         0        0         0         0         3         2

                   -----------   MODULE #  ( 13) =   SPLIT        -----------
    000      373        627       000       000       000       000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   6        25         26       0         0         0         0         2

                   -----------   MODULE #  ( 14) =   MIXER        -----------
    000      000        000       000       000       000       000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   14       25         0        0         0         0         16        2

                   -----------   MODULE #  ( 15) =   FLASH        -----------
    002      050      1 000     1 000     1 000 140   000     4 000     1 000       000       000
    000      000        000       000       000       000       000       000       000       000
   10       27         28       0         0         0         0         0

                   -----------   MODULE #  ( 16) =   FLASH        -----------
    100    1 100      1 000     1 000     1 000 820   000    43 000       000       000       000
    000      000        000       000       000       000       000       000       000       000
   28       29         30       0         0         0         0         0
```

30 percent and the number of sequential flowsheet evaluations was about 6 percent in the simultaneous modular approach when compared with the sequential modular approach. Table IV-22 summarizes the results obtained.

Table IV-22. Results for the Modified Ethanal Plant Problem

	ITER	FEVAL	CPU-TIME	# EQ
SIMO	14	55	9.95	17
SEQ	18	960	26.560	17

Furthermore, we had to provide better initial guesses, that is, closer to the solution to solve the problem with the sequential modular approach. The introduction of two flash drums increased the nonlinearities and was the main reason for the poor performance of the sequential modular approach.

IV-4 Nitric Acid Plant

The nitric acid plant problem was taken from Perkins (1975). A block diagram of the process is presented in figure IV-4.

This problem is very interesting because the way it was set up by Perkins, with 10 design constraints, it can be solved in less than 10 minutes using a hand calculator. Using any of the packages available - SIMO, EQSS or SIMFLOW - it would take a few hours to set up the problem (main program, data files, etc.). This problem teaches an important lesson about process simulation. It does not matter how complex a problem may look at first sight, a critical

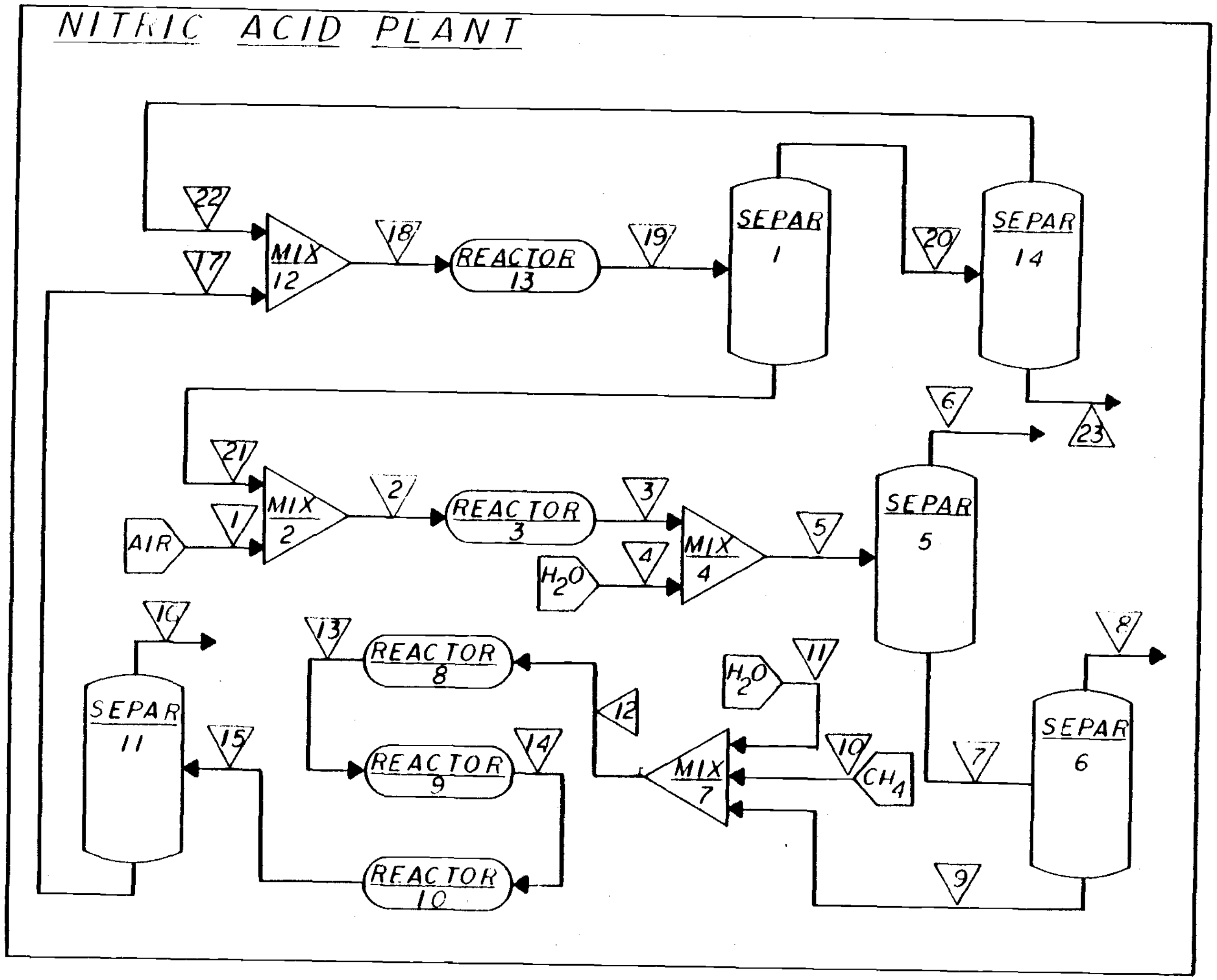


Figure IV-4. Block Diagram of the Nitric Acid Plant

analysis of the problem must be done before we attempt to solve it. Computer packages are powerful tools if we fully understand what we want to simulate; otherwise, there is no use for them.

Perkins used this problem to show the performance of his implementation of a simultaneous modular simulator. It should be pointed out that he does not call his implementation simultaneous modular, but sequential modular with a different convergence module.

The process may be divided into three parts. First, is the production of hydrogen. This is done by Reactors 8, 9 and 10. The first reactor performs a stream reforming yielding hydrogen and oxygen,

$$H_2O \rightleftarrows H_2 + 1/2\ O_2$$

The second and third reactors perform an oxidation of methane as follows,

$$2CH_4 + O_2 \rightleftarrows 2CO + 4H_2$$

$$CH_4 + O_2 \rightleftarrows CO_2 + 2H_2$$

Second is the production of ammonia from nitrogen and hydrogen,

$$N_2 + 3H_2 \rightleftarrows 2NH_3$$

Third is the oxidation of ammonia yielding nitric acid,

$$NH_3 + 2O_2 \rightleftarrows HNO_3 + H_2O$$

Table IV-23 summarizes the reactions in the nitric acid plant

Table IV-23. Reactions in the Nitric Acid Plant Problem

REACTION	REACTOR	CONVERSION[1/]
$H_2O \rightleftarrows H_2 + 1/2O_2$	8	100%, H_2O
$2CH_4 + O_2 \rightleftarrows 2CO + 4H_2$	9	4.7%, O_2
$CH_4 + O_2 \rightleftarrows CO_2 + 2H_2$	10	100%, O_2
$N_2 + 3H_2 \rightleftarrows 2NH_3$	13	25%, N_2
$NH_3 + 2O_2 \rightleftarrows HNO_3 + H_2O$	3	100%, NH_3

[1/] Percentage conversion in relation to the indicated reactant.

The separation fractions in each separator are presented in table IV-24.

Table IV-24. Separation Fractions as Percent of Individual Input Flow Rate

SEPAR	N_2	H_2	O_2	CH_4	NH_3	HNO_3	H_2O	CO	CO_2
1	100	100	--	--	--	--	--	--	--
5	--	--	--	--	--	100	100	--	--
6	93.6	--	91.3	--	--	--	--	--	--
11	--	--	--	--	--	--	--	100	100
14	100	99	--	--	100	--	--	--	--

The simulation of the nitric acid plant without design specifications was performed using the conversions shown in table IV-23, separation factors shown in table IV-24, and stream 18 was chosen as the tear stream.

The execution time and number of iterations for the simulation without constraints is presented in table IV-25.

Table IV-25. Results for the Nitric Acid Plant Problem

	ITER	FEVAL	CPU-TIME	# EQ
EQSS	2	--	38.405	190
SIMO	2	15	1.297	9
SEQ	45	45	2.012	9

The number of sequential iterations in the simultaneous modular approach was about 30 percent of the number required by the sequential modular approach. The execution time was about 60 percent of the time required by the simultaneous modular. Again the equation based approach converged quickly but the execution time was high.

The main programs are shown in table IV-26 and the solution in table IV-27.

Another version of this problem was solved imposing the concentration of nitrogen in stream 18 equal to 0.25. The separation factor of N_2 in separator 6 was used as a manipulated variable. The results are summarized in table IV-28.

Table IV-26. Main Programs Used to Simulate the Nitric Acid Plant:
SNO - sequential modular approach
SMNO - simultaneous modular approach
ENO - equation-based approach

```
      PROGRAM SNO(TAPE5 TAPE6)
      COMMON /P31/ ITOT
      REAL Y(9)
      EXTERNAL FSIS
      COMMON /S1/ NS NSTR(30).FLOW(30).NC.CNAME(20).COMP(30.20)
      COMMON /E1/ NE NEQP(20).EQNAME(20).EQP(20.20).IEQP(20.8)
      ITOT=0
      EPS=1 E-4
      CALL READ(23 14)
      NT=500
      N=9
      EPS=1 E-4
      K=0
      DO 5 I=1.9
    5 Y(I)=FLOW(18)*COMP(18.I)
      CALL WEGSMO(N Y.NT.EPS FSIS.K)
      WRITE(6.10) ITOT
   10 FORMAT(//.5X "CONVERGENCE ACHIEVED IN" I4." ITERATIONS".//)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(N.X.F)
      COMMON /S1/ NS NSTR(30).FLOW(30).NC.CNAME(20) COMP(30.20)
      COMMON /E1/ NE NEQP(20).EQNAME(20).EQP(20.20).IEQP(20.8)
      REAL X(N).F(N)
      COMMON /P31/ ITOT
      ITOT=ITOT+1
      FLOW(18)=0
      DO 10 I=1.9
   10 FLOW(18)=FLOW(18)+X(I)
      DO 20 I=1.9
   20 COMP(18.I)=X(I)/FLOW(18)
      CALL REAC(13)
      CALL SEPAR(1)
      CALL MIXER(2)
      CALL REAC(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL MIXER(7)
      CALL REAC(8)
      CALL REAC(9)
      CALL REAC(10)
      CALL SEPAR(11)
      CALL SEPAR(14)
      CALL MIXER(12)
      DO 30 I=1.9
   30 F(I)=FLOW(18)*COMP(18.I)
      RETURN
      END
```

```
      PROGRAM SMNO(INPUT TAPE5.TAPE6)
      CALL READ(23 14)
C
      N1=1
      N2=18
      N3=0
      IP=3
      NSIG=4
      CALL SIMSO(N1 N2 N3 IP.NSIG)
      CALL WRITES
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(X F II.N)
      REAL X(N).F(N)
      CALL REAC(13)
      CALL SEPAR(1)
      CALL MIXER(2)
      CALL REAC(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL MIXER(7)
      CALL REAC(8)
      CALL REAC(9)
      CALL REAC(10)
      CALL SEPAR(11)
      CALL SEPAR(14)
      CALL MIXER(12)
      RETURN
      END
```

```
      PROGRAM ENO(TAPE5.TAPE9.TAPE6)
      COMMON /S1/ NS NSTR(30).FLOW(30).NC.CNAME(20) COMP(30 20)
      COMMON /E1/ NE.NEQP(20).EQNAME(20).EQP(20.20) IEQP(20 8)
      CALL READ(23 14)
      CALL SREAD(23 14 9)
      CALL FSIS
      CALL FSIS
      CALL FSIS
C
      DO 100 I=1.NS
      IS=NINT(100*FLOW(I))
      FLOW(I)=IS/100
      DO 100 L=1.NC
      IS=ABS(COMP(I.L))*10000
      COMP(I L)=IS/10000
  100 CONTINUE
      CALL SIMMAN(2 2)
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS
      CALL REAC(13)
      CALL SEPAR(1)
      CALL SEPAR(14)
      CALL MIXER(2)
      CALL REAC(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL MIXER(7)
      CALL REAC(8)
      CALL REAC(9)
      CALL REAC(10)
      CALL SEPAR(11)
      CALL MIXER(12)
      RETURN
      END
```

Table IV-27. Solution of the Nitric Acid Plant Problem

```
STREAM  VARIABLES
NSTR   FLOW    N2       O2       NH3      HNO3     H2O      CO       CO2      CH4      H2
  1    9.35   .7900    .2100    .0000    .0000    .0000    .0000    .0000    .0000    .0000
  2   10.30   .7170    .1906    .0924    .0000    .0000    .0000    .0000    .0000    .0000
  3    9.35   .7900    .0065    .0000    .1018    .1018    .0000    .0000    .0000    .0000
  4    1.27   .0000    .0000    .0000    .0000   1.0000    .0000    .0000    .0000    .0000
  5   10.62   .6955    .0057    .0000    .0896    .2092    .0000    .0000    .0000    .0000
  6    3.17   .0000    .0000    .0000    .2999    .7001    .0000    .0000    .0000    .0000
  7    7.45   .9918    .0082    .0000    .0000    .0000    .0000    .0000    .0000    .0000
  8    6.97   .9920    .0080    .0000    .0000    .0000    .0000    .0000    .0000    .0000
  9     .48   .9898    .0102    .0000    .0000    .0000    .0000    .0000    .0000    .0000
 10     .38   .0000    .0000    .0000    .0000    .0000    .0000    .0000   1.0000    .0000
 11     .71   .0000    .0000    .0000    .0000   1.0000    .0000    .0000    .0000    .0000
 12    1.57   .3038    .0031    .0000    .0000    .4528    .0000    .0000    .2403    .0000
 13    1.92   .2477    .1871    .0000    .0000    .0000    .0000    .0000    .1959    .3692
 14    1.97   .2414    .1737    .0000    .0000    .0000    .0172    .0000    .1737    .3940
 15    2.31   .2056    .0000    .0000    .0000    .0000    .0146    .1480    .0000    .6317
 16     .38   .0000    .0000    .0000    .0000    .0000    .0900    .9102    .0002    .0000
 17    1.94   .2456    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .7544
 18    6.77   .2811    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .7189
 19    5.82   .2453    .0000    .1635    .0000    .0000    .0000    .0000    .0000    .5912
 20    4.87   .2932    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .7068
 21     .95   .0000    .0000   1.0000    .0000    .0000    .0000    .0000    .0000    .0000
 22    4.83   .2953    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .7047
 23     .03   .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000   1.0000

 EQUIPMENT PARAMETERS:          (2 X EQP(I,J))/IEQP(I,J)

                  ----------    MODULE # ( 1) = SEPAR       ----------
  1.000     .000     .000       .000     .000     .000     .000    1.000    1.000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
 19       20       21          0        0        0        0        0

                  ----------    MODULE # ( 2) = MIXER       ----------
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
  1       21        0          0        0        0        2        2

                  ----------    MODULE # ( 3) = REAC        ----------
   .000   -2.000   -1.000      1.000    1.000     .000     .000     .000     .000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000    1.000
  2        3        0          0        0        0        0        3

                  ----------    MODULE # ( 4) = MIXER       ----------
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
  3        4        0          0        0        0        5        2

                  ----------    MODULE # ( 5) = SEPAR       ----------
   .000     .000     .000      1.000    1.000     .000     .000     .000     .000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
  5        6        7          0        0        0        0        0

                  ----------    MODULE # ( 6) = SEPAR       ----------
   .936     .919     .000       .000     .000     .000     .000     .000     .000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
  7        8        9          0        0        0        0        0

                  ----------    MODULE # ( 7) = MIXER       ----------
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
  9       10       11          0        0        0       12        3

                  ----------    MODULE # ( 8) = REACT       ----------
   .000     .500     .000       .000   -1.000     .000     .000     .000    1.000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000    1.000
 12       13        0          0        0        0        0        5

                  ----------    MODULE # ( 9) = REACT       ----------
   .000   -1.000     .000       .000     .000    2.000     .000   -2.000    4.000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .047
 13       14        0          0        0        0        0        2

                  ----------    MODULE # (10) = REACT       ----------
   .000   -1.000     .000       .000     .000     .000    1.000   -1.000    2.000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000    1.000
 14       15        0          0        0        0        0        2

                  ----------    MODULE # (11) = SEPAR       ----------
   .000    1.000     .000       .000     .000    1.000    1.000    1.000     .000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
 15       16       17          0        0        0        0        0

                  ----------    MODULE # (12) = MIXER       ----------
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
 17       22        0          0        0        0       18        2

                  ----------    MODULE # (13) = REACT       ----------
 -1.000     .000    2.000       .000     .000     .000     .000     .000   -3.000     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .250
 18       19        0          0        0        0        0        1

                  ----------    MODULE # (14) = SEPAR       ----------
  1.000     .000     .000       .000     .000     .000     .000     .000     .990     .000
   .000     .000     .000       .000     .000     .000     .000     .000     .000     .000
 20       22       23          0        0        0        0        0
```

Table IV-28. Results of the Nitric Acid Plant Problem with One Design Specification

	ITER	FEVAL	CPU-TIME	# EQ
EQSS	2	--	38.01	190
SIMO	2	16	1.685	10
SEQ	3	255	6.732	10

Again, the simultaneous modular approach was far better than the sequential modular approach. Compared with SEQ, the execution time if SIMO was about 30 percent and the number of sequential iterations about 6 percent. The execution time in the equation based approach was again rather high, although it did converge in two iterations. In addition, there is no significant difference in the execution time of EQSS when comparing the problem with one design specification and without design specification. In the equation based approach, when we specify one variable as fixed and allow a corresponding parameter to be a variable we can expect this behavior, that is, the same execution time.

Table IV-29 presents the main programs of the HNO_3 plant with one design specification and table IV-30 the solution of the problem.

The last HNO_3 plant simulation problem had 7 design specifications. Because of the large number of constraints, we did not attempt to solve the problem with the sequential modular approach, it was solved only with the simultaneous modular approach and with the equation based approach.

Table IV-29. Main Programs Used to Simulate the Nitric Acid Plant Problem with One Design Specification: SCINO - sequential modular approach
SMCINO - simultaneous modular approach

```
      PROGRAM SCINO(TAPE5.TAPE6)
      COMMON /MWEG/ NSEQ
      EXTERNAL SUB
      COMMON /P31/ ITOT
      NSEQ=0
      ITOT=0
      NT=50
      EPS=1 E-4
      X=0 93
      CALL READ(23.14)
      CALL SECNEW(X,NT EPS SUB)
      WRITE(6 10) NI ITOT
   10 FORMAT(//,5X."CONVERGENCE ACHIEVEO IN".I4," ITERATIONS",
     1          /,5X "NUMBER OF FLOWSHEET EVALUATIONS = ",I4,//)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE SUB(X F)
      EXTERNAL FSIS
      COMMON /S1/ NS NSTR(30),FLOW(30).NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE NEQP(20) EQNAME(20),EQP(20.20) IEQP(20,8)
      REAL Y(9)
      NT=500
      N=9
      EPS=1 E-5
      K=0
      DO 5 I=1,9
    5 Y(I)=FLOW(18)*COMP(18,I)
      EQP(6,2)=X
      CALL WEGSMD(N,Y,NT,EPS,FSIS,K)
      F=0 25-COMP(18,1)
      RETURN
      END
C
C
C
C
      SUBROUTINE FSIS(N,X,F)
      COMMON /S1/ NS.NSTR(30).FLOW(30).NC.CNAME(20).COMP(30.20)
      COMMON /E1/ NE.NEQP(20).EQNAME(20).EQP(20.20).IEQP(20.8)
      REAL X(N).F(N)
      COMMON /P31/ ITOT
      ITOT=ITOT+1
      FLOW(18)=0
      DO 10 I=1 9
   10 FLOW(18)=FLOW(18)+X(I)
      DO 20 I=1.9
   20 COMP(18,I)=X(I)/FLOW(18)
      CALL REAC(13)
      CALL SEPAR(1)
      CALL MIXER(2)
      CALL REAC(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL MIXER(7)
      CALL REAC(8)
      CALL REAC(9)
      CALL REAC(10)
      CALL SEPAR(11)
      CALL SEPAR(14)
      CALL MIXER(12)
      DO 30 I=1 9
   30 F(I)=FLOW(18)*COMP(18.I)
      RETURN
      END
```

```
      PROGRAM SMCINO(TAPE5.TAPE6)
      REAL VAL(10)
      COMMON /S1/ NS.NSTR(30).FLOW(30).NC.CNAME(20).COMP(30.20)
      COMMON /E1/ NE.NEQP(20).EQNAME(20).EQP(20,20).IEQP(20.8)
      INTEGER NUE(10).NUP(10).NUS(10).NCO(10)
      CHARACTER*4 NAME(10).NACQ(10)
      CALL REAO(23.14)
C
      N1=1
      N2=18
      N3=1
      IP=3
      NSIG=2
      NAME(1)='SEPA'
      NUE(1)=6
      NUP(1)=2
      NACO(1)='COMP'
      NUS(1)=18
      NCO(1)=1
      VAL(1)=0.25
C
      CALL SPEC(N3,NAME.NUE,NUP,NACO.NUS.NCO,VAL)
      CALL SIMSO(N1.N2.N3.IP,NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      ENO
C
C
C
C
      SUBRQUTINE FSIS(X.F,IT.N)
      REAL X(N),F(N)
      CALL REAC(13)
      CALL SEPAR(1)
      CALL MIXER(2)
      CALL REAC(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL MIXER(7)
      CALL REAC(8)
      CALL REAC(9)
      CALL REAC(10)
      CALL SEPAR(11)
      CALL SEPAR(14)
      CALL MIXER(12)
      RETURN
      END
C
C
```

Table IV-30. Solution of the Nitric Acid Plant Problem with One Design Specification

```
 STREAM  VARIABLES
 NSTR   FLOW    N2        O2        NH3       HNO3      H2O       CO        CO2       CH4       H2
   1    9.35   .7900     .2100     .0000     .0000     .0000     .0000     .0000     .0000     .0000
   2   10.30   .7174     .1907     .0919     .0000     .0000     .0000     .0000     .0000     .0000
   3    9.35   .7900     .0077     .0000     .1012     .1012     .0000     .0000     .0000     .0000
   4    1.26   .0000     .0000     .0000     .0000    1.0000     .0000     .0000     .0000     .0000
   5   10.61   .6961     .0068     .0000     .0891     .2080     .0000     .0000     .0000     .0000
   6    3.15   .0000     .0000     .0000     .3000     .7000     .0000     .0000     .0000     .0000
   7    7.46   .9904     .0096     .0000     .0000     .0000     .0000     .0000     .0000     .0000
   8    6.98   .9904     .0096     .0000     .0000     .0000     .0000     .0000     .0000     .0000
   9     .48   .9899     .0101     .0000     .0000     .0000     .0000     .0000     .0000     .0000
  10     .38   .0000     .0000     .0000     .0000     .0000     .0000     .0000    1.0000     .0000
  11     .71   .0000     .0000     .0000     .0000    1.0000     .0000     .0000     .0000     .0000
  12    1.56   .3026     .0031     .0000     .0000     .4536     .0000     .0000     .2407     .0000
  13    1.92   .2467     .1874     .0000     .0000     .0000     .0000     .0000     .1962     .3697
  14    1.97   .2403     .1739     .0000     .0000     .0000     .0172     .0000     .1739     .3946
  15    2.31   .2047     .0000     .0000     .0000     .0000     .0147     .1482     .0000     .6325
  16     .38   .0000     .0000     .0000     .0000     .0000     .0900     .9100     .0000     .0000
  17    1.93   .2445     .0000     .0000     .0000     .0000     .0000     .0000     .0000     .7555
  18    7.57   .2500     .0000     .0000     .0000     .0000     .0000     .0000     .0000     .7500
  19    6.62   .2143     .0000     .1429     .0000     .0000     .0000     .0000     .0000     .6429
  20    5.67   .2500     .0000     .0000     .0000     .0000     .0000     .0000     .0000     .7500
  21     .95   .0000     .0000    1.0000     .0000     .0000     .0000     .0000     .0000     .0000
  22    5.63   .2519     .0000     .0000     .0000     .0000     .0000     .0000     .0000     .7481
  23     .04   .0000     .0000     .0000     .0000     .0000     .0000     .0000     .0000    1.0000

  EQUIPMENT  PARAMETERS        (2 X EQP(I,J))/IEQP(I,J)

                         -----------     MODULE #  (  1)  =   SEPAR        -----------
   1.000       .000       .000       .000       .000       .000       .000      1.000      1.000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
  19         20         21          0          0          0          0          0

                         -----------     MODULE #  (  2)  =   MIXER        -----------
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
   1         21          0          0          0          0          2          2

                         -----------     MODULE #  (  3)  =   REAC         -----------
    .000     -2.000     -1.000      1.000      1.000       .000       .000       .000       .000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000      1.000
   2          3          0          0          0          0          0          3

                         -----------     MODULE #  (  4)  =   MIXER        -----------
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
   3          4          0          0          0          0          5          2

                         -----------     MODULE #  (  5)  =   SEPAR        -----------
    .000       .000       .000      1.000      1.000       .000       .000       .000       .000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
   5          6          7          0          0          0          0          0

                         -----------     MODULE #  (  6)  =   SEPAR        -----------
    .936       .933       .000       .000       .000       .000       .000       .000       .000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
   7          8          9          0          0          0          0          0

                         -----------     MODULE #  (  7)  =   MIXER        -----------
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
   9         10         11          0          0          0         12          3

                         -----------     MODULE #  (  8)  =   REACT        -----------
    .000       .500       .000       .000     -1.000       .000       .000       .000      1.000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000      1.000
  12         13          0          0          0          0          0          5

                         -----------     MODULE #  (  9)  =   REACT        -----------
    .000     -1.000       .000       .000       .000      2.000       .000     -2.000      4.000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .047
  13         14          0          0          0          0          0          2

                         -----------     MODULE #  (10)  =   REACT        -----------
    .000     -1.000       .000       .000       .000       .000      1.000     -1.000      2.000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000      1.000
  14         15          0          0          0          0          0          2

                         -----------     MODULE #  (11)  =   SEPAR        -----------
    .000      1.000       .000       .000       .000      1.000      1.000      1.000       .000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
  15         16         17          0          0          0          0          0

                         -----------     MODULE #  (12)  =   MIXER        -----------
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
  17         22          0          0          0          0         18          2

                         -----------     MODULE #  (13)  =   REACT        -----------
  -1.000       .000      2.000       .000       .000       .000       .000       .000     -3.000       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .250
  18         19          0          0          0          0          0          1

                         -----------     MODULE #  (14)  =   SEPAR        -----------
   1.000       .000       .000       .000       .000       .000       .000       .000       .990       .000
    .000       .000       .000       .000       .000       .000       .000       .000       .000       .000
  20         22         23          0          0          0          0          0
```

Table IV-31 is a list of the design specifications and manipulated variables.

Table IV-31. Design Specifications for the Nitric Acid Plant Problem

Design Specification	Variable Manipulated
FLOW(6) = 3.174	EQP(6,1) (SEPARATOR)
COMP(6,5) = 0.7	EQP(6,2) (SEPARATOR)
COMP(7,1) = 0.992	EQP(9,20) (REACTOR)
COMP(9,2) = 0.01	FLOW(1) (Feed)
COMP(15,9) = 0.0	FLOW(4) (Feed)
COMP(16,7) = 0.91	FLOW(10) (Feed)
COMP(18,1) = 0.25	FLOW(11) (Feed)

For this problem the initial guesses were taken as ± 10 percent of the solution value, that is, each variable of the solution vector was multiplied by 0.9 or 1.1.

The results obtained are presented in table IV-32; the main program for SIMO is presented in table IV-33.

Table IV-32. Results for the Nitric Acid Plant with Seven Design Specifications

	ITER	FEVAL	CPU-TIME	# EQ
EQSS	2	--	59.259	190
SIMO	11	68	11.894	16

Table IV-33. Main Program Used to Simulate the Nitric Acid Plant Problem with Seven Design Specifications

```
      PROGRAM SMC7NO(INPUT,TAPE5,TAPE6)
      REAL VAL(10)
      INTEGER NUE(10),NUP(10),NUS(10),NCO(10)
      CHARACTER*4 NAME(10),NACO(10)
      CALL READ(23,14)
C
      N1=1
      N2=18
      N3=7
      IP=3
      NSIG=4
C
      NAME(1)='SEPA'
      NUE(1)=6
      NUP(1)=2
      NACO(1)='COMP'
      NUS(1)=18
      NCO(1)=1
      VAL(1)=0.25
C
C

      NAME(2)='SEPA'
      NUE(2)=6
      NUP(2)=1
      NACO(2)='COMP'
      NUS(2)=15
      NCO(2)=8
      VAL(2)=0.0
C
      NAME(3)='FLOW'
      NUE(3)=4
      NUP(3)=0
      NACO(3)='COMP'
      NUS(3)=6
      NCO(3)=5
      VAL(3)=.7
C
      NAME(4)='REAC'
      NUE(4)=9
      NUP(4)=20
      NACO(4)='COMP'
      NUS(4)=16
      NCO(4)=7
      VAL(4)=.91
C
      NAME(5)='FLOW'
      NUE(5)=10
      NUP(5)=0
      NACO(5)='COMP'
      NUS(5)=9
      NCO(5)=2
      VAL(5)=0.01
C
      NAME(6)='FLOW'
      NUE(6)=11
      NUP(6)=0
      NACO(6)='COMP'
      NUS(6)=7
      NCO(6)=1
      VAL(6)=0.992
C
      NAME(7)='FLOW'
      NUE(7)=1
      NUP(7)=0
      NACO(7)='FLOW'
      NUS(7)=6
      NCO(7)=0
      VAL(7)=3.174
C
      CALL SPEC(N3,NAME,NUE,NUP,NACO,NUS,NCO,VAL)
      CALL SIMSO(N1,N2,N3,IP,NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(X,F,IT,N)
      REAL X(N),F(N)
      CALL REAC(13)
      CALL SEPAR(1)
      CALL MIXER(2)
      CALL REAC(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL SEPAR(6)
      CALL MIXER(7)
      CALL REAC(8)
      CALL REAC(9)
      CALL REAC(10)
      CALL SEPAR(11)
      CALL SEPAR(14)
      CALL MIXER(12)
      RETURN
      END
```

The equation-based approach converged in a few iterations, whereas the simultaneous modular approach took almost 5 times more iterations than the equation-based approach. In terms of execution time, the simultaneous modular approach was far better. This problem shows that when the number of design specifications increases, the simultaneous modular approach requires more iterations, 4 or 5 times more, than the equation-based approach.

The same problem (seven design specifications), but with a better set of initial guesses, i.e., in the range of ±2 percent of the actual solution, required 16 iterations (55 sequential iterations) in the simultaneous modular approach, and 2 iterations in the equation-based approach. The execution time was about 7 CPU-seconds for SIMO and about 40 CPU-seconds for EQSS. The equation-based approach maintained the trend of 2-3 iterations to solve the problem as well as the simultaneous modular approach with 11-16 iterations.

In time, there is no discrepancy in the results obtained with the simultaneous modular approach. The first set of initial guesses required 11 iterations of MPDLM with 68 sequential iterations. The sequential iterations are distributed as follows:

3 - initialize the simulation

51 - initial evaluation of the Jacobian and two re-evluations of the Jacobian

14 - required by MPDLM

The second set of initial guesses required 16 iterations of MPDLM and 55 sequential iterations distributed as follows:

3 - initialize the simulation

34 - initialize evaluation of the Jacobian and one re-evalution of the Jacobian

18 - required by MPDLM

The extra re-evaluation of the Jacobian required by the first set of initial guesses increased the execution time, because the Jacobian matrix is factored every time the Jacobian is evaluated. On the other hand, the better approximation of the Jacobian obtained from that re-evaluation allowed the first system to converge in fewer iterations of MPDLM (11 against 16).

IV-5 Gasoline Recovery

This test problem was taken from Reklaitis (Reklaitis, 1983), it is problem 5-36, page 368.

A block diagram of the process is shown in figure IV-5.

The input gas stream has several hydrocarbons (from C1 to C9) which will be stripped of gasoline range components with decane. It is known the split fraction of splitters 1 and 9 are known equal to 0.75 and 0.264, respectively.

This problem is regarded as a challenge to process simulators when energy balances and physical property calculations are performed. However, because in this work only material balances are performed there was little challenge to either approach used.

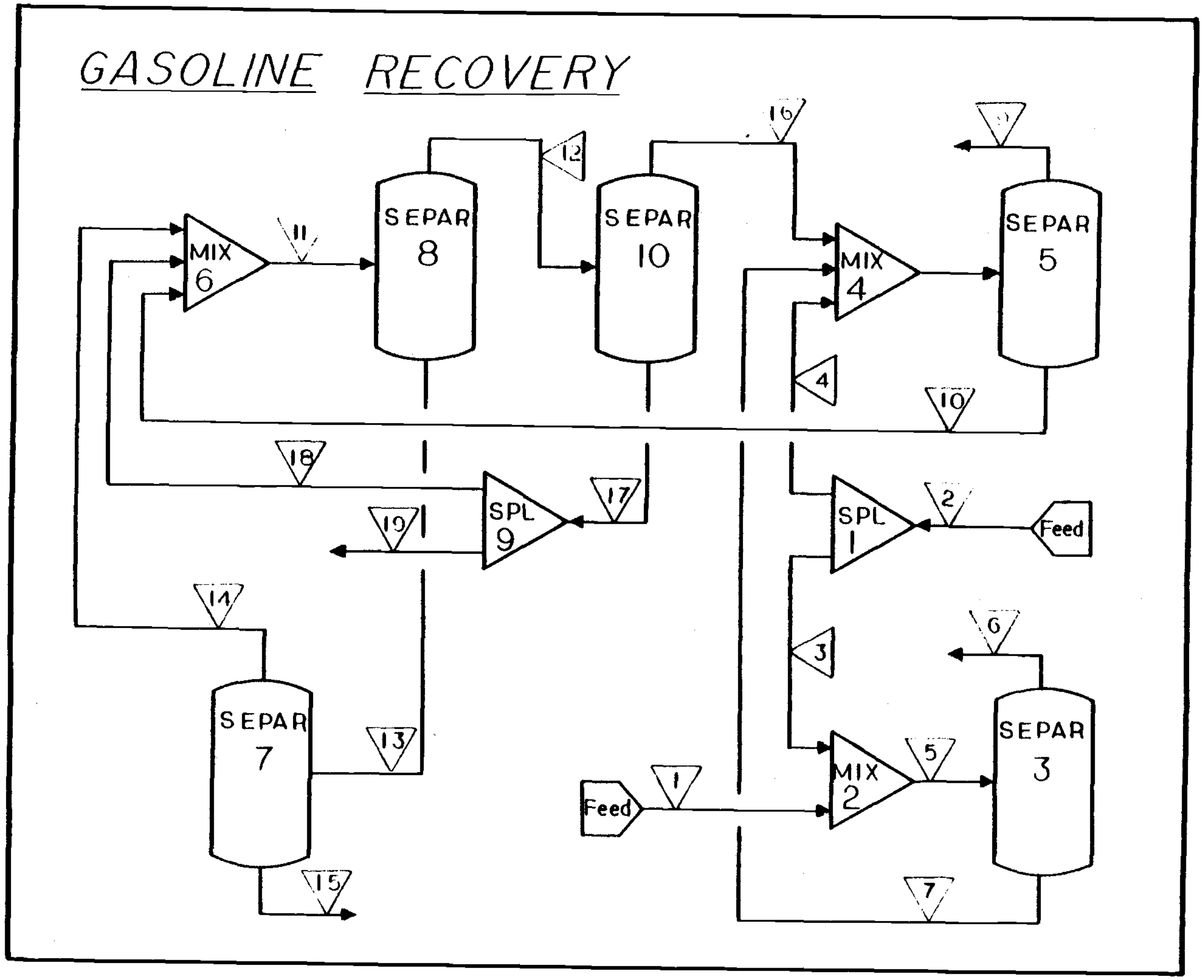


Figure IV-5. Block Diagram of the Gasoline Recovery Problem

The execution time and number of iterations are presented in table IV-34.

Table IV-34. Results for the Gasoline Recovery Problem

	ITER	FEVAL	CPU-TIME	# EQ
EQSS	4	--	81.320	187
SIMO	5	20	3.146	10
SEQ	3	3	0.916	10

Surprisingly, the number of iterations required by the equation based approach and by the simultaneous modular approach were higher than the sequential modular approach.

The main programs used are presented in table IV-35, the solution is presented in table IV-36.

The design version of the problem has the flow rate of stream 19 fixed and equal to 72 (moles/unit of time) and the split fraction of SPL9 is a manipulated variable. The results are summarized in table IV-37.

Table IV-37. Results for the Gasoline Recovery Problem with One Design Specification

	ITER	FEVAL	CPU-TIME	# EQ
EQSS	4	--	82.856	188
SIMO	2	17	1.908	11
SEQ	4	35	1.573	11

Table IV-35. Main Programs Used to Simulate the Gasoline Recovery Problem

```
      PROGRAM SC8(TAPE5,TAPE6)
      COMMON /ITT/ ICO
      REAL X(10),Y(10)
      EXTERNAL FSIS
      CALL READ(19,10)
      X(1)=0
      X(2)=175
      X(3)=175
      X(4)=175
      X(5)=175
      X(6)=175
      X(7)=175
      X(8)=175
      X(9)=175
      X(10)=2100
      N=10
      ICO=-1
      NT=100
      EPS=1.E-4
      K=0
      WRITE(6,222)
  222 FORMAT(/,10X," INITIAL ESTIMATES OF TORN STREAMS AND",
     1         /,10X,"EQUIPAMENT PARAMETERS",/)
      CALL WRITES
      CALL WRITEE
      CALL WEGSMD(N,X,NT,EPS,FSIS,K)
      WRITE(6,10) ICO
   10 FORMAT(///,10X,"CONVERGENCE ACHIEVED IN ",I4,"  ITERATIONS ",/)
      CALL WRITES
      CALL WRITEE
      STOP
      END
C
C
C
C
      SUBROUTINE FSIS(M,X,F)
      REAL X(M),F(M)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /ITT/ ICO
      ICO=ICO+1
      FLOW(11)=0
      DO 1 I=1,M
    1 FLOW(11)=FLOW(11)+X(I)
      DO 2 I=1,M
    2 COMP(11,I)=X(I)/FLOW(11)
      CALL SEPAR(8)
      CALL SEPAR(10)
      CALL SPLIT(9)
      CALL SPLIT(1)
      CALL MIXER(2)
      CALL SEPAR(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL SEPAR(7)
      CALL MIXER(6)
      DO 3 I=1,M
    3 F(I)=FLOW(11)*COMP(11,I)
      RETURN
      END
```

```
      PROGRAM SMC8(TAPE5,TAPE6)
      CHARACTER*4 NAME,NACO
      CALL READ(19,10)
      I1=1
      I2=11
      I3=0
      IP=3
      NSIG=4
      CALL SIMSO(I1,I2,I3,IP,NSIG)
      CALL WRITES
      CALL WRITEE
      STOP
      END


      SUBROUTINE FSIS(X,F,IT,N)
      REAL X(N),F(N)
      CALL SEPAR(8)
      CALL SEPAR(10)
      CALL SEPAR(7)
      CALL SPLIT(9)
      CALL SPLIT(1)
      CALL MIXER(2)
      CALL SEPAR(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL MIXER(6)
      RETURN
      END


      PROGRAM EC8(TAPE5,TAPE9,TAPE6)
      CALL READ(19,10)
      CALL SREAD(19,10,10)
      CALL FSIS
      CALL SIMMAN(3,2)
      STOP
      END


      SUBROUTINE FSIS
      CALL SEPAR(8)
      CALL SEPAR(10)
      CALL SEPAR(7)
      CALL SPLIT(9)
      CALL SPLIT(1)
      CALL MIXER(2)
      CALL SEPAR(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL MIXER(6)
      RETURN
      END
```

Table IV-36. Solution for the Gasoline Recovery Problem

```
 STREAM VARIABLES
 NSTR   FLOW   CH4      C2H6     C3H8     C4H10    C5H12    C6H14    C7H16    C8H18    C9H20     LEAN
  1   653.84  .5863    .0413    .0963    .0550    .0507    .0357    .0428    .0488    .0431     .0000
  2  2556.00  .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000    1.0000
  3  1917.00  .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000    1.0000
  4   639.00  .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0000    1.0000
  5  2570.84  .1491    .0105    .0245    .0140    .0129    .0091    .0109    .0124    .0110     .7457
  6   370.59  .8847    .0472    .0666    .0013    .0002    .0000    .0000    .0000    .0000     .0000
  7  2200.25  .0252    .0043    .0174    .0161    .0150    .0106    .0127    .0145    .0128     .8713
  8  3329.79  .0169    .0152    .0986    .0349    .0274    .0110    .0098    .0101    .0086     .7676
  9    86.36  .6423    .0971    .2063    .0241    .0094    .0001    .0000    .0000    .0000     .0207
 10  3243.43  .0002    .0130    .0957    .0352    .0279    .0113    .0101    .0103    .0088     .7875
 11  3781.93  .0002    .0116    .0930    .0478    .0404    .0192    .0180    .0175    .0134     .7388
 12   724.13  .0011    .0606    .4569    .1712    .1388    .0568    .0455    .0355    .0200     .0135
 13  3057.80  .0000    .0000    .0069    .0186    .0171    .0103    .0115    .0132    .0118     .9106
 14   398.35  .0000    .0002    .0426    .1025    .0925    .0485    .0462    .0456    .0340     .5879
 15  2659.46  .0000    .0000    .0015    .0061    .0058    .0046    .0063    .0084    .0085     .9590
 16   490.54  .0016    .0837    .5910    .1645    .1187    .0269    .0097    .0032    .0007     .0000
 17   233.58  .0001    .0122    .1754    .1854    .1810    .1197    .1206    .1034    .0605     .0419
 18    93.43  .0001    .0122    .1754    .1854    .1810    .1197    .1206    .1034    .0605     .0419
 19   140.15  .0001    .0122    .1754    .1854    .1810    .1197    .1206    .1034    .0605     .0419

  EQUIPMENT PARAMETERS          (2 X EQP(I,J))/IEQP(I,J)

                   ----------     MODULE # ( 1) =  SPLIT       ----------
      .000     .750      .250     .000     .000     .000     .000     .000     .000     .000
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
     2        3         4        0        0        0        0        2

                   ----------     MODULE # ( 2) =  MIXER       ----------
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
     1        3         0        0        0        0        5        2

                   ----------     MODULE # ( 3) =  SEPAR       ----------
      .855     .647      .392     .013     .002     .000     .000     .000     .000     .000
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
     5        6         7        0        0        0        0        0

                   ----------     MODULE # ( 4) =  MIXER       ----------
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
     4        7        16        0        0        0        8        3

                   ----------     MODULE # ( 5) =  SEPAR       ----------
      .986     .166      .054     .018     .009     .000     .000     .000     .000     .001
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
     8        9        10        0        0        0        0        0

                   ----------     MODULE # ( 6) =  MIXER       ----------
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
    10       19        14        0        0        0       11        3

                   ----------     MODULE # ( 7) =  SEPAR       ----------
      .000     .985      .807     .717     .706     .614     .524     .450     .375     .084
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
    13       14        15        0        0        0        0        0

                   ----------     MODULE # ( 8) =  SEPAR       ----------
     1.000     .998      .940     .685     .658     .567     .484     .389     .287     .004
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
    11       12        13        0        0        0        0        0

                   ----------     MODULE # ( 9) =  SPLIT       ----------
      .000     .400      .600     .000     .000     .000     .000     .000     .000     .000
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
    17       18        19        0        0        0        0        2

                   ----------     MODULE # (10) =  SEPAR       ----------
      .984     .935      .876     .651     .579     .321     .144     .062     .024     .000
      .000     .000      .000     .000     .000     .000     .000     .000     .000     .000
    12       16        17        0        0        0        0        0
C
```

The simultaneous modular approach had fewer iterations required than the sequential modular and equation based approach. The execution time was better with the sequential modular approach. The main programs for the simultaneous modular and sequential modular approach are presented in table IV-38; the solution of the problem is in table IV-39.

Table IV-38. Main Program Used to Simulate the Gasoline Recovery Problem with One Design Specification

```
      PROGRAM SCIC8(TAPE5,TAPE6)
      COMMON /P31/ ITOT
      EXTERNAL SUB
      ITOT=0
      EPS=1 E-4
      NT=50
      X= 4
      CALL REAO(19,10)
      WRITE(6,222)
  222 FORMAT(/,10X," INITIAL ESTIMATES OF TORN STREAMS ANO",
     1       /,10X," EQUIPAMENT PARAMETERS:",/)
      CALL WRITES
      CALL WRITEE
      CALL SECNEW(X,NT,EPS,SUB)
      WRITE(6,10) NT,ITOT
   10 FORMAT(//,5X,"CONVERGENCE ACHIEVED IN ",I4," ITERATIONS",
     1       /,5X,"NUMBER OF FLOWSHEET EVALUATIONS= ",I4,//)
      CALL WRITES
      CALL WRITEE
      STOP
      ENO
C
C
C
C
      SUBROUTINE SUB(X,F)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      REAL Y(10)
      EXTERNAL FSIS
      DO 10 I=1,10
   10 Y(I)=FLOW(11)*COMP(11,I)
      N=10
      NT=200
      EPS=1 E-5
      K=0
      EQP(9,2)=X
      EQP(9,3)=1 -X
      CALL WEGSMD(N,Y,NT,EPS,FSIS,K)
      F=FLOW(18)-72 O
      RETURN
      ENO
C
C
C
C
      SUBROUTINE FSIS(M,X,F)
      REAL X(M),F(M)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /P31/ ICO
      ICO=ICO+1
      FLOW(11)=0
      OO 1 I=1,M
    1 FLOW(11)=FLOW(11)+X(I)
      DO 2 I=1,M
    2 COMP(11,I)=X(I)/FLOW(11)
      CALL SEPAR(8)
      CALL SEPAR(10)
      CALL SPLIT(9)
      CALL SPLIT(1)
      CALL MIXER(2)
      CALL SEPAR(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL SEPAR(7)
      CALL MIXER(6)
      OO 3 I=1,M
    3 F(I)=FLOW(11)*COMP(11,I)
      RETURN
      END
```

```
      PROGRAM SMCIC8(TAPE5,TAPE6)
      CHARACTER*4 NAME,NACO
      CALL REAO(19,10)
      I1=1
      I2=11
      I3=1
      IP=3
      NAME='SPLI'
      NUE=9
      NUP=2
      NACO='FLOW'
      NUS=18
      NCO=1
      VAL=72
      CALL SPEC(I3,NAME,NUE,NUP,NACO,NUS,NCO,VAL)
      CALL SIMSO(I1,I2,I3,IP)
      CALL WRITES
      CALL WRITEE
      STOP
      ENO
C
C
C
C
      SUBROUTINE FSIS(X,F,IT,N)
      REAL X(N),F(N)
      CALL SEPAR(8)
      CALL SEPAR(10)
      CALL SEPAR(7)
      CALL SPLIT(9)
      CALL SPLIT(1)
      CALL MIXER(2)
      CALL SEPAR(3)
      CALL MIXER(4)
      CALL SEPAR(5)
      CALL MIXER(6)
      RETURN
      END
```

Table IV-39. Solution for the Gasoline Recovery Problem with One Design Specification

```
STREAM VARIABLES :
NSTR   FLOW    CH4     C2H6    C3H8    C4H10   C5H12   C6H14   C7H16   C8H18   C9H20     LEAN
 1   653.84  .5863   .0413   .0963   .0550   .0507   .0357   .0428   .0488   .0431     .0000
 2  2556.00  .0000   .0000   .0000   .0000   .0000   .0000   .0000   .0000   .0000    1.0000
 3  1917.00  .0000   .0000   .0000   .0000   .0000   .0000   .0000   .0000   .0000    1.0000
 4   639.00  .0000   .0000   .0000   .0000   .0000   .0000   .0000   .0000   .0000    1.0000
 5  2570.84  .1491   .0105   .0245   .0140   .0129   .0091   .0109   .0124   .0110     .7457
 6   370.59  .8847   .0472   .0666   .0013   .0002   .0000   .0000   .0000   .0000     .0000
 7  2200.25  .0252   .0043   .0174   .0161   .0150   .0106   .0127   .0145   .0128     .8713
 8  3416.91  .0165   .0154   .1115   .0388   .0304   .0114   .0098   .0099   .0084     .7480
 9    89.97  .6165   .0970   .2299   .0264   .0103   .0001   .0000   .0000   .0000     .0199
10  3326.94  .0002   .0132   .1083   .0391   .0309   .0118   .0101   .0101   .0086     .7677
11  3955.06  .0002   .0117   .1051   .0550   .0468   .0219   .0199   .0186   .0137     .7069
12   850.34  .0009   .0543   .4598   .1754   .1434   .0578   .0448   .0337   .0183     .0115
13  3104.72  .0000   .0000   .0080   .0221   .0204   .0121   .0131   .0145   .0125     .8973
14   427.44  .0000   .0002   .0469   .1150   .1045   .0540   .0498   .0474   .0340     .5481
15  2677.27  .0000   .0000   .0018   .0072   .0070   .0054   .0072   .0093   .0090     .9531
16   577.67  .0013   .0747   .5931   .1681   .1223   .0273   .0095   .0031   .0006     .0000
17   272.67  .0000   .0110   .1775   .1911   .1880   .1224   .1197   .0987   .0556     .0359
18    72.00  .0000   .0110   .1775   .1911   .1880   .1224   .1197   .0987   .0556     .0359
19   200.67  .0000   .0110   .1775   .1911   .1880   .1224   .1197   .0987   .0556     .0359

 EQUIPMENT  PARAMETERS:          (2 X EQP(I,J))/IEQP(I,J)

                 -----------      MODULE # ( 1) =  SPLIT     -----------
   .000      .750       .250      .000     .000     .000     .000     .000     .000     .000
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
  2         3          4         0        0        0        0        2

                 -----------      MODULE # ( 2) =  MIXER     -----------
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
  1         3          0         0        0        0        5        2

                 -----------      MODULE # ( 3) =  SEPAR     -----------
   .855      .647       .392      .013     .002     .000     .000     .000     .000     .000
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
  5         6          7         0        0        0        0        0

                 -----------      MODULE # ( 4) =  MIXER     -----------
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
  4         7          16        0        0        0        8        3

                 -----------      MODULE # ( 5) =  SEPAR     -----------
   .986      .166       .054      .018     .009     .000     .000     .000     .000     .001
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
  8         9          10        0        0        0        0        0

                 -----------      MODULE # ( 6) =  MIXER     -----------
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
 10        19          14        0        0        0       11        3

                 -----------      MODULE # ( 7) =  SEPAR     -----------
   .000      .985       .807      .717     .706     .614     .524     .450     .375     .084
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
 13        14          15        0        0        0        0        0

                 -----------      MODULE # ( 8) =  SEPAR     -----------
  1.000      .998       .940      .685     .658     .567     .484     .389     .287     .004
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
 11        12          13        0        0        0        0        0

                 -----------      MODULE # ( 9) =  SPLIT     -----------
   .000      .264       .736      .000     .000     .000     .000     .000     .000     .000
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
 17        18          19        0        0        0        0        2

                 -----------      MODULE # (10) =  SEPAR     -----------
   .984      .935       .876      .651     .579     .321     .144     .062     .024     .000
   .000      .000       .000      .000     .000     .000     .000     .000     .000     .000
 12        16          17        0        0        0        0        0
C
```

Final Remarks on Chapter IV

All results obtained are summarized as ratios; table IV-40 presents iteration ratios, table IV-41 presents CPU-time ratios.

Table IV-40. Iteration Ratios

		SIMO1/	SEQ1/	SIMO2/	SEQ2/
	A	.617	0.093	ND	ND
EQSS	B	.882	0.039	ND	ND
	C	.467	0.111	ND	ND
	A	1	0.131	1	0.104
SIMO	B	1	0.05	1	0.3
	C	1	0.971	1	0.069

1/ Total number of iterations of main nonlinear solver (MPDLM or SECNEW)

2/ Total number of flowsheet evaluations

A - all problems

B - only problems with no design specifications

C - only problems with one or more design specifications

ND- not defined

The ratios are defined as the number of iterations of the horizontal entry divided by the number of iterations of the vertical entry.

Table IV-41. CPU Time Ratios

		SIMO	SEQ	SEQ[1/]
	A	10.216	14.214	--
EQSS	B	10.113	17.177	--
	C	10.368	10.445	--
	A	1	0.697	0.439
SIMO	B	1	1.851	0.820
	C	1	0.372	0.372

[1/] without Cavett problem

It can be seen from table IV-40 that the equation-based approach had the best convergence performance. In the case of problems with one or more constraints the equation-based approach requires half of the iterations required by SIMO and 10 percent of those required by SECNEW.

The true comparison between SIMO and the sequential modular approach is in the total number of flowsheet evalutations. SIMO requires about 7 percent of the iterations required by the sequential modular approach for constrainted problems, about 30 percent for problems without constraints and 10 percent overall.

It is clear that the equation-based approach has the worst performance of the three methods in terms of execution time. The simultaneous modular approach performed best with design constraints, but without design constraints only marginal benefits were achieved.

CHAPTER V

CONCLUSIONS

The implementation of Chen and Stadtherr's modification of Powell's dogleg method proved to be an efficient alternative to solve systems of nonlinear equations. When used to solve nonlinear systems arising from chemical process simulations it was very reliable, even with poor initial guesses.

The equation based approach is an attractive alternative for chemical process simulations. However, in order to have the equation-based approach attractive for commercial applications, better algorithms to solve the set of linear equations must be incorporated. In addition, the Jacobian evaluation must be optimized. Knowing the modules which will be used in the simulation, it is easy to know the structure of the Jacobian; therefore, there is no need to perturb all variables to evaluate the Jacobian matrix by forward differences, only the ones that affect a given equation.

The simultaneous modular approach is better than the sequential modular approach for controlled simulations, that is, for simulations with design specifications.

The findings in this work are applicable to simulations where only material balances are performed. The results with simultaneous mass and energy balances, and physical properties estimation may result in different conclusions. However, we expect better performance of the equation-based approach and simultaneous modular approach when compared with the sequential modular approach. We expect that behavior because the execution time for each flowsheet pass will be higher when physical properties and energy balances are performed. In addition, we expect a significant increase in the severity of nonlinearities which will affect Wegstein's method performance.

The findings of this work agreed with results presented in the open literature regarding the three methods: the sequential modular, simultaneous modular and equation-based approach. However, the results of this work also indicate that much more research must be done with the equation-based approach.

Finally, two libraries of subroutines were developed, SIMO and EQSS, which provide powerful teaching aids for undergraduate level stoichiometry classes.

Future Work

It became quite clear while this work was in progress that the main problem associated with the simultaneous modular approach and equation based approach is the effective solution of a system of nonlinear equations. A few areas must be further researched; namely:

1. The evaluation of the Jacobian accounts for a significant amount of execution time. It would be interesting to have the Jacobian coded analytically. Although the task is tedious, it is feasible. We must remember that material

and energy balance equations have the same form for all equipment modules, that is, output = input + accumulation. Therefore, there is no need to code the analytical Jacobian in the case of material and energy balances equations for each module, a general subroutine would do it. If the analytical Jacobian is not expensive to evaluate it would be preferable to evaluate the Jacobian every iteration rather than to update it by secant formulas.

2. Although the results obtained with Schubert's update formula were fairly good, it would be interesting to analyze the effect of maintaining all known constant Jacobian elements, not just the zeros. In Chapter II we presented Schubert's update formula and we mentioned that in practice we hold as constants only the elements equal to zero. The question that should be answered is what is the gain, if any, of not updating all known constants, but only the ones equal to zero. This question is very important because some authors (for example, Lucia, 1982) reported unreliable results using Schurbert's update formula. Were the unreliable results due to Schubert's formula, or due to simplifications introduced to Schubert's update formula?

3. EQSS and SIMO should perform energy balances. As teaching aids, SIMO and EQSS fulfill the purpose of showing the performance of different approaches to solve process

flowsheets. On the other hand, the type of simulations that can be performed with SIMO and EQSS are rather simple, and their use in research is limited. Introducing energy balances and physical properties calculations would enable SIMO and EQSS to solve more complex problems.

4. A weak point in EQSS is the solution of the linear systems every iteration. A better (more efficient) subroutine must be incorporated to solve the linear system. There are several such algorithms published in the open literature. Unfortunately, our choice was not the best.

BIBILOGRAPHY

Bennett, J.M., "Triangular Factors of Modified Matrices." Numer. Math. 7, 217 (1965).

Broyden, C.G., "A Class of Methods for Solving Nonlinear Simultaneous Equations." Math. Comp. 19, 577 (1965).

______, "The Convergence of an Algorithm for Solving Sparse Nonlinear Systems." Math. Comp. 25, 114 (1971).

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

Chen, H.S. and Stadtherr, M.A., "A Modification of Powell's Dogleg Method for Solving Systems of Nonlinear Equations." Comput. Chem. Eng. 5, 143 (1981).

Chen, H.S., "Computational Strategies for Chemical Process Flowsheeting and Optimization." Ph.D. Thesis, University of Illinois, Urbana (1982).

Crowe, C.M., "On a Relationship Between Quasi-Newton and Dominant Eigen Value Methods for the Numerical Solution of Nonlinear Equations." Comput. Chem. Eng. 8, 35 (1984).

Dennis, Jr., J.E., and Schnabel, R.B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, 1st ed. (1983).

Duff, I.S., "A Survey of Sparse Matrix Research." Proc. of the IEEE 4, 500 (1977).

Gupta, P.K., Lavoie, R.C., and Radcliffe, R.R., "An Industry Evaluation of SPEEDUP." 1984 Annual Meeting of the A.I.Ch.E., San Francisco.

Gallun, S.E., and Holland, C.D., "A Modification of Broyden's Method for the Solution of Sparse Systesm - With Applications to Distillation Problems Described by Nonideal Thermodyanamic Functions." Comput. Chem. Eng. 4, 93 (1980).

Hlavacek, V., "Analyses of a Complex Plant - Steady State and Transient Behavior." Comput. Chem. Eng. 1, 75 (1977).

Kayihan, F., "SIMFLOWS: Simplified Modular Flowsheet Simulator." Chemical Engineering Department, Oregon State University, Rev. 2 (1979).

Lucia, A., "Some Results for one Iterated Projection of the Symmetric Schubert Update." Comput. Chem. Eng. 6, 283 (1982).

Mah, R.S.H., and Lin, T.C., "Comparison of Modified Newton's Methods." Comput. Chem. Eng. 4, 75 (1980).

Mahelec, V., Kluzik, H., and Evans, L.B., "Simultaneous Modular Algorithm for Steady-State Flowsheet Simulation and Design." Presented at 12th Symposium on Computer Applications in Chemical Engineering, CACE 79, Montreux (1979).

Motard, R.L., Shacham, M. and Rosen, E.M., "Steady State Process Simulation." AIChE J. 3, 417 (1975).

Myers, A.L., and Seider, W.D., Introduction to Chemical Engineering and Computer Calculations. Prentice-Hall, 1st ed. (1976).

Perkins, J.D., "Efficient Solution of Design Problems Using a Sequential-Modular Flowsheeting Program." Comput. Chem. Eng. 3, 375 (1979).

Perkins, J.D., and Sargent, R.W.H., "SPEEDUP: A Computer Program for Steady-State and Dynamic Simulation and Design of Chemical Processes." Selected Topics on Computer-Aided Process Design and Analyses. CEP Symposium Series, A.I.Ch.E., New York (1982).

Powell, M.J.D., "A Hybrid Method for Nonlinear Equations." Numerical Methods for Nonlinear Algebraic Equations, P. Rabinoiwitz (ed.), Gordon and Breach, New York (1970).

Reklaitis, G.V., Introduction to Material and Energy Balances. John Wiley and Sons, 1st ed. (1983).

Rodrigues, A.F., "Solucao de Sistemas Esparsos de Equacoes Algebricas Lineares por Metodos Diretos." M.S. Thesis, Escola Politecnica da Universidade de Sao Paulo, Brazil (1979).

Rosen, E.M., "A Machine Computation Method for Peforming Material Balances." Chem. Eng. Progress 58, 69 (1962).

______, "A Review of Quasi-Newton Methods in Nonlinear Equation Solving and Unconstrained Optimization." Proc. 21st ACM National Meeting, Washington D.C. (1966).

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

Schubert, L.K., "Modification of a Quasi-Newton Method for Nonlinear Equations With a Sparse Jacobian." Math. Comp. 25, 27 (1970).

Stadtherr, M.A., and Wood, E.S., "Sparse Matrix Methods for Equation-Based Chemical Process Flowsheeting - I." Comput. Chem. Eng. 8, 9 (1984).

Ibid. Part II, Comput. Chem. Eng. 8, 19 (1984).

Wegstein, J.H., "Accelerating Convergence of Iterative Processes." Commun. Assoc. Computing Machinery 6, 9 (1958).

Westerberg, A.W., Hutchinson, H.P., Motard, R.L., and Winter, P., Process Flowsheeting, Cambridge University Press, Cambridge (1979).

APPENDICES

Appendix A

Solution Equations

In this appendix we will show the equations solved in each equipment module used by the sequential and simultaneous modular approach.

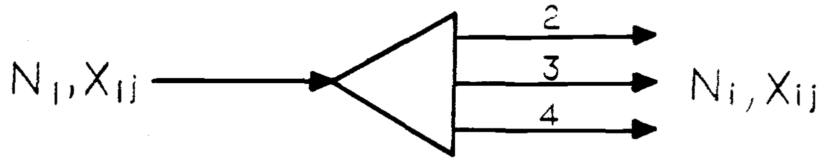
In either approach the input stream variables and equipment parameters are known. The following nomenclature will be used:

x_{ij}	=	mole fraction of the jth component in the ith stream
N_i	=	total molar flow rate of the ith stream
∂_k	=	kth split fraction
β_i	=	separation fraction of the ith component
μ_i	=	stoichiometric coefficient of the ith component
γ_k	=	conversion of the kth reactant
k_i	=	liquid-vapor constant of the ith component

<u>SPLITTER</u> n components

L streams out (max 7)

1 stream in



Equipment parameter: split fraction ∂_k

$$\partial_k = \frac{N_k}{N_1}$$

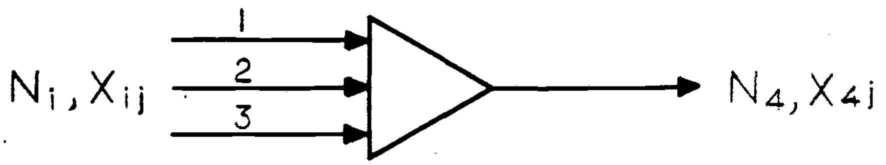
$$N_k = \partial_k N_1 \qquad k = 2, L$$

$$x_{kj} = x_{ij} \qquad k = 2, L; \quad j = 1, n$$

<u>MIXER</u> n components

L input streams (up to 7)

1 output stream



Equipment parameters: none

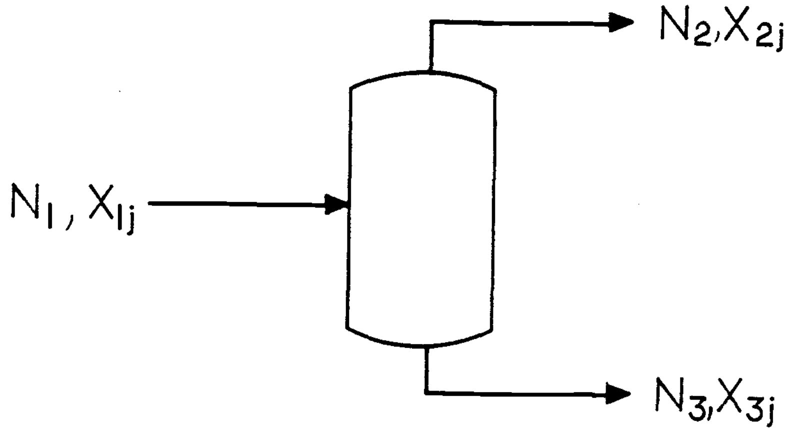
$$N_4 = \sum_{k=1}^{3} N_k$$

$$x_{4,j} = \frac{\sum_{k=1}^{3} N_k \, x_{k,j}}{N_4} \qquad j = 1, n$$

SEPARATOR n components

L output streams (up to 3)

1 input stream



Equipment parameters: $\beta_j = \dfrac{N_2\ x_{2,j}}{N_1\ x_{1,j}}$ $\quad j = 1, n$

$$N_3 = N_1 \sum_{J=1}^{n} (1 - \beta_j)\ x_{1,j}$$

$$N_2 = N_1 - N_3$$

$$x_{2,j} = \frac{\beta_j\ N_1\ x_{1j}}{N_2} \qquad j = 1, n$$

$$x_{3,j} = \frac{N_1\ x_{1,j} - N_2\ x_{2,j}}{N_3} \qquad j = 1, n$$

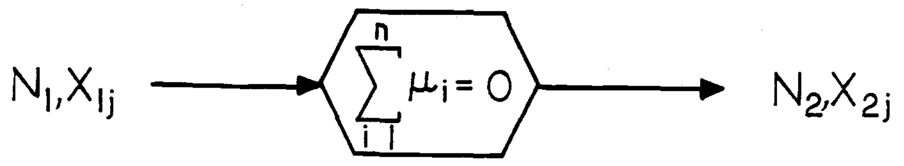
<u>REACTOR</u> n components

μ_i = stoichiometric coefficients

$\ominus$ reactant

$\oplus$ product

$\textcircled{0}$ inert



Equipment parameters: $$\gamma_k = \frac{N_1\, x_{1k} - N_2\, x_{2k}}{N_1\, x_{1k}}$$

Conversion of the k reactant

Define $$r = \frac{N_1\, x_{1k}\, \gamma_k}{-\mu_k}$$ extent of reaction

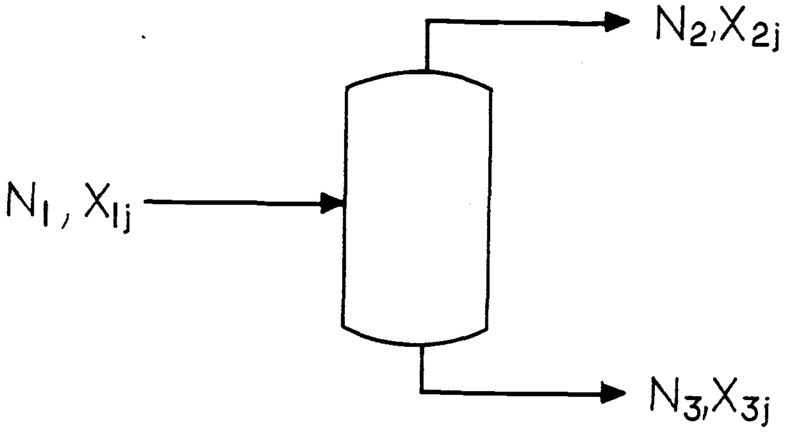
$$N_2 = N_1 + r \sum_{J=1}^{n} N_j$$

$$x_{2j} = \frac{N_1\, x_{1J} + r\, \mu_j}{N_2}$$

<u>FLASH</u> (Isothermal) n components

2 ouput streams

1 input stream



Equipment parameter: $K_j = \dfrac{x_{2j}}{x_{3j}}$ $\quad j = 1, n$

Define $\quad \alpha = \dfrac{N_3}{N_1}$

Solve

$$\sum_{i=1}^{n} \frac{x_{1i}}{1 - \alpha(1 - 1/k_i)} - 1 = 0$$

for α

$$N_3 = \alpha N_1$$

$$N_2 = N_1 - N_3$$

$$x_{2,J} = \frac{x_{1J}}{1 - \alpha(1 - 1/k_j)} \qquad J = 1, n$$

$$x_{3,J} = \frac{x_{2,J}}{k_J} \qquad J = 1, n$$

APPENDIX B

Chen and Stardtherr Modification of Powell's Dogleg Method to Solve Sets of Nonlinear Equations

In this appendix we will try to show how Chen and Stadtherr's algorithm functions. The references for this appendix are: Powell (1970), Broyden (1965), and Chen and Stadtherr (1981).

We will now present a brief review of methods for solving nonlinear equation, which are relevant to understand the algorithm.

Consider the solution of N nonlinear equations in N unknowns. We aim at the determination of the vector X^* such that

$$F(X^*) = 0 \tag{1}$$

where F, X, and 0 are N$^{\underline{th}}$ order vectors.

Several iterative procedures to determine X^*, solution vector of (1), have the general form:

$$X^{i+1} = X^i - H^i F^i T^i \tag{2}$$

where:

i	=	iteration number
X^i	=	i$^{\underline{th}}$ estimate of the solution
F^i	=	function values at X^i
T^i	=	scaling factor
H^i	=	depends on the method used

To follow the convergence towards the solution, the euclidian norm of the functions is traced.

$$S(X) = \sum_{J=1}^{N} (F_J^i(X))^{\frac{1}{2}} = \| F^i(X) \| \qquad (3)$$

When a vector X^* reduced the euclidian norm to a determined small value ∂, we accept X^* as the solution to the system (1). Thus, X^* is a solution if,

$$\| F^i(X^*) \| \leq \partial \qquad (4)$$

It is desirable that at each iteration the euclidian norm of F(X) is reduced, so we would like that

$$\| F^{i+1}(X) \| \leq \| F^i(X) \| \qquad (5)$$

In the Newton-Raphson method H^i is chosen to be the inverse of the Jacobian of F(X). Equation (2) becomes

$$X^{i+1} = X^i - (J^i)^{-1} F^i \qquad (6)$$

and $T^i = 1$.

In the steepest descent method H^i is chosen to be $(J^i)^T$. The method is based on finding a minimum of $Z(X) = F(X)^T F(X)$. The gradient of Z(X) is $2J^T F(X)$ and the steepest descent direction of Z(X) is -grad Z(X) . So if we choose H^i to be $(J^i)^T$ equation (2) becomes

$$X^{i+1} = X^i - (J^i)^T F(X^i) T^i \qquad (7)$$

If we choose T^i properly $Z(X)$ will be reduced every iteration. Note that $Z(X) = \; S(X) \; ^2 = \| F^i(X) \| ^2$. The method may fail to solve (1) if a local minimum, different than the solution vector X^*, is reached. It can be shown that at the minimum

$$J^T F(X) \; = \; 0$$

since its not a zero minimum $F(X) \neq 0$, J^T must be singular. This is not a serious drawback, as several algorithms, for example the Newton-Raphson method, require J^{-1}. However, if such local minimum is not reached, the steepest descent algorithm will eventually reach the zero minimum. It has been found that the steepest descent algorithm is rather slow.

Another iterative method used is the Levenberg-Marquardt. In this method $H^i = \; (J^i)^T J^i + \lambda^i I \; ^{-1} (J^i)^T$ and T_i is set equal to 1. The parameter λ^i, Levenberg-Marquardt parameter, is chosen to be always greater than or equal to zero. Equation (2) becomes

$$X^{i+1} = X^i - \; (J^i)^T J^i + \lambda^i I \; ^{-1} (J^i)^T F^i \tag{8}$$

A few comments are in order at this time. If we set $\lambda^i = 0$, equation (8) reduces to equation (6), the Newton-Raphson method. If we let λ^i big enough, equation (8) reduces to equation (7), the steepest descent algorithm, where $T^i \simeq (\lambda^i I)^{-1}$. It can be shown that equation (5) is satisfied with a proper choice of $\lambda^i \quad 0$.

It is important to note that if $\lambda^i \quad 0$, the inverse matrix in (8) will always exist, which eliminates the drawback of the other algorithm, a singular Jacobian. However, as in the steepest descent algorithm, the L-M algorithm may find a local minimum in which $\| F(X) \| \neq 0$.

An important feature of the L-M algorithm is that, depending on the value of λ^i, it has good global convergence properties, as the steepest descent direction method or good local convergence property, as Newton-Raphson method.

Let us rewrite equation (8) in the following manner:

$$X^{i+1} = X^i + P^i \tag{9}$$

where

$$P^i = -[(J^i)^T(J^i) + \lambda^i I]^{-1}(J^i)^T F^i \tag{10}$$

and P^i is a correction step which value is chosen as to satisfy equation (5). Clearly, $P^i = f(\lambda^i)$, and the problem now is to find λ_i, so as to satisfy equation (5). As λ^i may assume any value in the interval $(0, +\infty)$ we will have to assume several values of λ and check whether equation (5) is satisfied or not.

To exemplify the procedure, consider the following system of equations:

$$\begin{aligned} f_1(X) &= x_1^2 + x_2^2 - 5 = 0 \\ f_2(X) &= 2x_1 + x_2 - 4 = 0 \end{aligned} \tag{11}$$

Let $X^1 = (1,1)^T$, thus $F^1(X) = (-3,-1)^T$ and the Jacobian is:

$$J = \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix}$$

Table B-1 shows several values of P^1 as a function of λ^1.

Table B-1. Correction step as a Function of λ

λ_i	P_1^i	P_2^i
0.0	-0.50	2.00
0.5	0.19	1.07
1.0	0.33	0.83
5.0	0.40	0.46
10.0	0.33	0.33

The L-M curve is shown in figure B-1. To obtain the curve a linear system had to be solved for each value of λ considered. Clearly, this procedure is time consuming as the number of equations in the system increases. Powell approximates the L-M curve by a broken curve as shown in figure B-2. The curve $\vec{BC}$ was obtained through the steepest descent direction of $S(X) = \| F(X) \|$. As it was previously discussed, as λ increases the L-M equation (8) tends towards the steepest descent direction, so the direction of $\vec{BC}$ is the steepest descent direction given by $G = -\text{grad } S(X) = -J^TF$ in the example,

$$G = \begin{bmatrix} 2 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} +3 \\ +1 \end{bmatrix} = \begin{bmatrix} 8 \\ 7 \end{bmatrix}$$

and

$$P^S = \overline{AB} = \mu G \qquad (12)$$

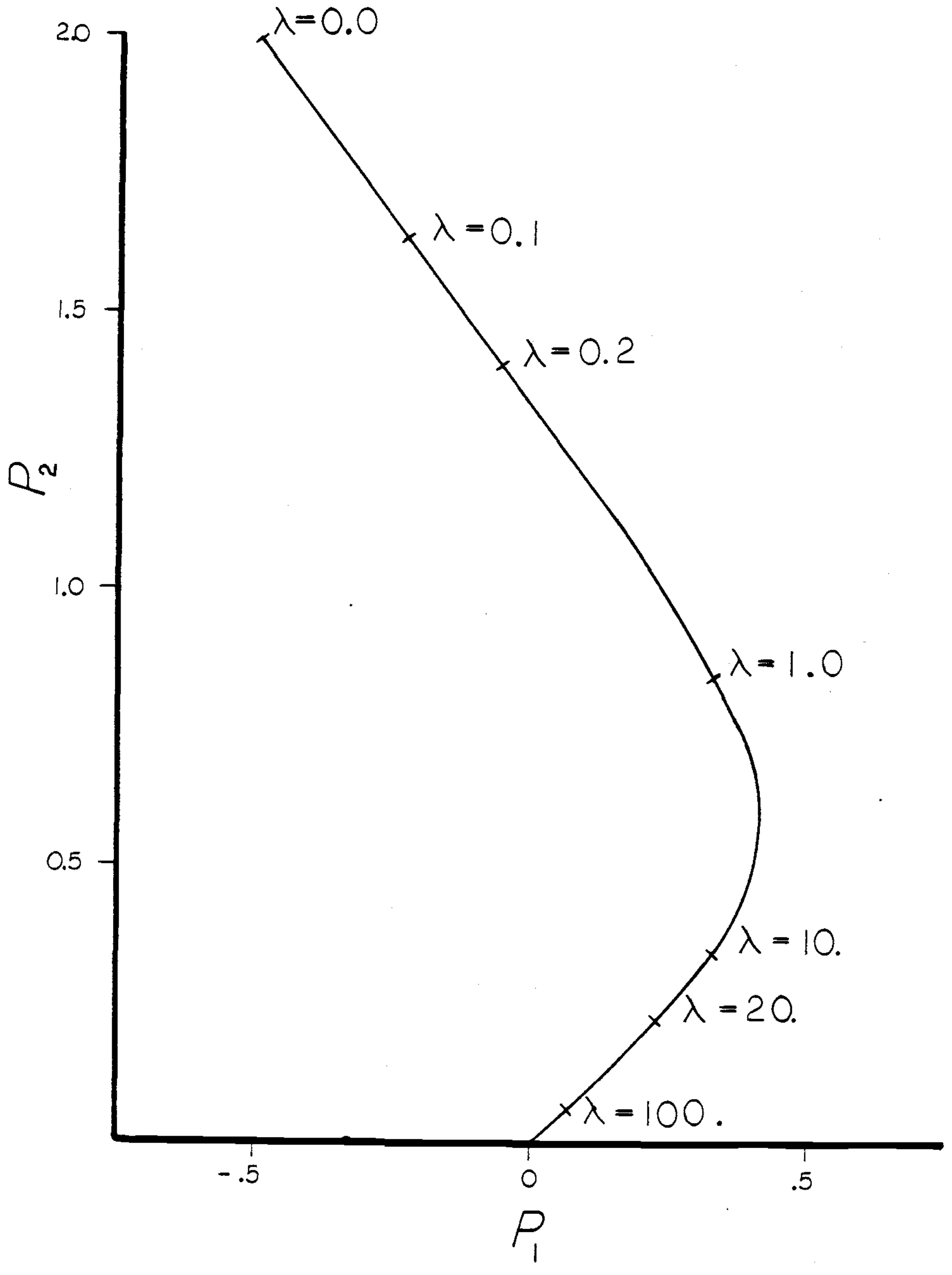


Figure B-1. Typical 2-Dimensional P(λ) Curve in the L-M Method

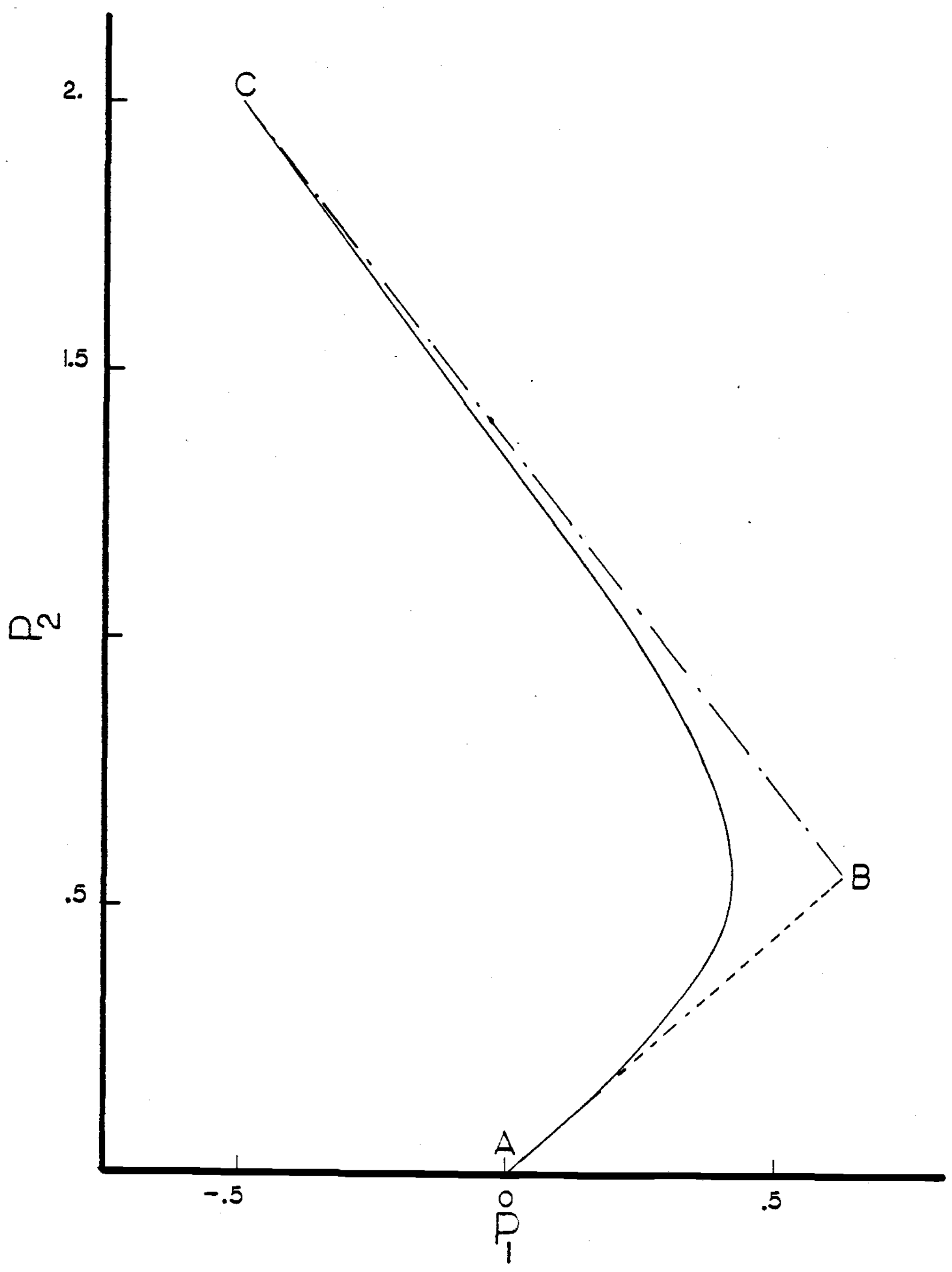


Figure B-2. Typical Dogleg Approximation to the P(λ) Curve where

$$\mu = \frac{\| G \|^2}{\| JG \|^2} \tag{13}$$

$$\mu = \frac{(8^2 + 7^2)}{(30^2 + 23^2)} = 0.079076$$

$$\overline{AB} = 0.79076 \begin{bmatrix} 8 \\ 7 \end{bmatrix}$$

For $\lambda = 0$, $P(\lambda)$ is just the correction step given by N-R algorithm so that

$$\lambda = 0; \ \vec{BC} = P(\lambda)$$

$$P^N = P(\lambda) = -J^{-1}F = [-0.5;2]^T$$

Now the broken line ABC (dogleg) is completely described and, as seen in figure B-2, it well represents the L-M curve. The advantage of using Powell's dogleg curve to approximate the L-M curve is that only one linear system was solved, the N-R correction step.

In order to solve the system of equations (11), it would be interesting if after each iteration we could reduce the value of $\| F \|$, in other words, we want equation (3) to be satisfied every iteration. Let us consider a trust region Δ which represents the distance between X_i and X^* (the solution vector). One of the good characteristics of Newton's method is that it converges fast nearby the solution, and one good characteristic of the L-M method is that for a sufficiently large λ equation (5) is satisfied. The parameter Δ will be used to determine the correction step through the following algorithm.

Use:

$$P^N \text{ if } \Delta \geq \| P^N \| \qquad \text{(N-R step)}$$

$$P^S \text{ if } \Delta \leq \| P^S \| \qquad \text{(Steepest Descent Step)}$$

If $\| P^N \| < \Delta < \| P^S \|$ we will use the intersection of the broken line (dogleg) with the circle of radius Δ.

$$P = \alpha P^N + (1 - \alpha) P^S$$

where

$$\alpha = \frac{\Delta^2 - \| P^S \|}{(P^N - P^S)^T P^S + [((P^N)^T P^S - \Delta^2)^2 + (\| P^N \|^2 - \Delta^2)(\Delta^2 - \| P^S \|^2))^{\frac{1}{2}}]}$$

In this case we will keep the good convergence properties of the L-M algorithm. To illustrate the procedure let us use again the system of equations (11) and assume that $\Delta = \sqrt{2}$.

$$\| P^N \| = 2.0615$$

$$\| P^S \| = 0.8406$$

so

$$\| P^N \| \quad \Delta \quad \| P^S \|$$

and

$$P^{NS} = (-0.04085;1.413624)^T$$

The next step in the algorithm is to update Δ. We want Δ as large as possible because we want to decrease $\| F \|$ in every iteration, without taking too small steps. If Δ is too small the number of iterations required is prohibitive.

Chen and Stardtherr suggest the following procedure. If the Jacobian is new, use:

$$\Delta = \| P^i \| \max (0.1,\lambda)$$

where

$$\lambda = \frac{-b}{2a}$$

and

$$a = S^* - S - 2(F^i)^T J^i P^i$$

$$b = 2(F^i)^T J^i P^i$$

$$S^* = \| F(X^i + P^i) \|^2$$

$$S = \| F(X^i \|^2$$

Otherwise, evaluate d_m,

$$dm = S - S^* - 0.1(S - \| q\|^2)$$

$$q = F(X^i) + J^i P^i$$

if $d_m < 0$ set

$$\Delta = 0.5 \| P^i \|$$

if $d_m \geq 0$

$$\lambda^2 = 1 + \frac{d_m}{\sigma_P + (\sigma_P{}^2 + d_m \sigma_S)^{\frac{1}{2}}}$$

$$\sigma_P = \sum_{J=1}^{n} | f_J(X^i + P^i)[(f_j(X^i + P^i)) - q_S] |$$

$$\sigma_S = \sum_{J=1}^{n} [f_j(X^i + P^i) - q_j]^2$$

The new Δ is defined as follows:

- set TFLAG = 1 if the Jacobian is new or whenever Δ is reduced.
- calculate λ and find MV = min(2,λ,TFLAG)
- reset TFLAG = λ/MV
- set Δ = MV$\| P^i \|$

Other features of Chen's and Startherr algorithm are:

- Check for slow convergence or nonconvergence. This check is done because the algorithm may converge to a local minimum and no progress is done towards the solution. If the check shows slow convergence of nonconvergence the program stops and a different set of initial guesses is required.

- The update of the Jacobian through secant formulas "degrades" as the number of iterations increases. When the Jacobian is not making good progress, it is reevaluated through finite differences.

What was exposed so far are the basic ideas behind Chen and Stadtherr's modification of Powell's dogleg method. The following references will cover in more details the dogleg method and its modifications: Powell (1970), Broyden (1970), Chen and Stardtherr (1981).

If anyone wants to work out the system of equations (7), table A-2 shows some results.

Table B-2. Results for the System of Equations (12)

ITERATION	1	2	3
X_1	1.0	0.9591	1.0849
X_2	1.0	2.4136	1.8301
F_1	-3.0	1.7455	-0.4737
F_2	-1.0	0.3319	-0.0001
Δ	$\sqrt{2}$	1.1007	
P_1	-0.04095	0.1258	
P_2	1.4136	-0.5835	
$\|P\|$	2.0	0.3563	
$\|F\|$	3.16	1.7768	0.2244
α	0.5946	--	

APPENDIX C

Subroutine Listing

EQSS

LIBRARY

```
C-------------------------------------------------------------------
C
C
C
C
      SUBROUTINE SREAD(NSTR,NEQ,NC)
      COMMON /S1/ NS,NS1(30),FLOW(30) MN,CNAME(20),COMP(30,20)
      CHARACTER*5 NAME
```

```
      COMMON /SIM1/ NAME(20)
      COMMON /SIM2/ IFK(30),ICK(30,20),IEK(20,20),IIOP(20,7)
      COMMON /SIM3/ ID(20),ICC(30),IEC(20),INOP(20),CX(30,20)
C
      READ(9,1) A
    1 FORMAT(A3)
      DO 2 I=1,NSTR
    2 READ(9,10) L,IFK(L),(ICK(L,J),J=1,20)
      READ(9,1) A
      DO 3 I=1,NEQ
    3 READ(9,20) L,NAME(L),(IEK(L,J),J=1,20)
   10 FORMAT(1X,I2,3X,I2,1X,20(I2))
   20 FORMAT(I2,1X,A5,1X,20(I2))
C
      DO 30 I=1,NSTR
      IF(IFK(I).EQ.1) CX(I,20)=FLOW(I)
      DO 30 L=1,NC
      IF(ICK(I,L).EQ.1) CX(I,L)=COMP(I,L)
   30 CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE IDEN
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      CHARACTER*5 NA
      COMMON /SIM1/ NA(20)
      COMMON /SIM2/ IFK(30),ICK(30,20),IEK(20,20),ITOP(20,7)
      COMMON /SIM3/ ID(20),ICC(30),IEC(20),INOP(20),CX(30,20)
C
      DO 1 I=1,NE
      IF(NA(I).EQ.'SPLIT') THEN
      ID(I)=1
      GO TO 1
      ELSE
      IF(NA(I).EQ.'MIXER') THEN
      IO(I)=2
      GO TC 1
      ELSE
      IF(NA(I).EQ.'REACT'.OR.NA(I).EQ.'REAC') THEN
      ID(I)=3
      GO TO 1
      ELSE
      IFINA(I).EQ.'SEPAR') THEN
      ID(I)=4
      GO TO 1
      ELSE
      IF(NA(I).EQ.'FLASH') THEN
      ID(I)=5
      GD TO 1
      ELSE
      IF(NA(I).EQ.'USER1'.OR.NA(I).EQ.'USER') THEN
      ID(I)=5
      GC TO 1
      ELSE
      ENDIF
      ENDIF
      ENDIF
      ENDIF
      ENDIF
      ENOIF
      WRITE(6,10) I,NA(I)
   10 FORMAT(/,5X,"CHECK SPELLING FOR MODULE(",I2,")=",A5,/)
      STOP
    1 CONTINUE
C
      DO 20 I=1,NS
      ICC(I)=0
      IF(IFK(I).EQ.1) FLOW(I)=CX(I,20)
       DO 21 L=1,NC
      IF(ICK(I,L).EQ.1) THEN
      ICC(I)=ICC(I)+1
      COMP(I,L)=CX(I,L)
      INOP(ICC(I))=L
      GO TO 21
      ELSE
      ENDIF
   21 CONTINUE
      IF(ICC(I).EQ.0) GO TO 20
      DO 22 L=1,ICC(I)
   22 ICK(I,L)=INOP(L)
   20 CONTINUE
C
      DO 30 I=1,NE
      IEC(I)=0
      IF(ID(I) EQ 2) GO TO 30
      L1=1
      L2=NC
      IF(ID(I) GE 4) GO TO 35
      L1=2
      L2=IEQP(I,8)+1
      IF(ID(I) EQ 1) GO TO 35
      L1=20
      L2=20
   35 CONTINUE
      DO 31 L=L1,L2
      IF(IEK(I,L) EQ 0) THEN
      IEC(I)=IEC(I)+1
      INOP(IEC(I))=L
      GO TO 31
      ELSE
      ENDIF
   31 CONTINUE
      IF(IEC(I) EQ 0) GO TO 30
      DO 32 L=1,IEC(I)
   32 IEK(I,L)=INOP(L)
   30 CONTINUE
C
      DO 40 K=1,NE
      GO TO (50,60 70 80 80,90),ID(K)
C
   50 CONTINUE
      INOP(K)=IEQP(K 8)+1
      DO 110 I=1,INOP(K)
  110 ITOP(K,I)=IEQP(K,I)
      GO TO 40
C
   60 CONTINUE
      INOP(K)=IEQP(K 8)+1
      DO 120 I=2,INOP(K)
  120 ITOP(K,I)=IEQP(K,I-1)
      ITOP(K,1)=IEQP(K,7)
      GO TO 40
C
   70 CONTINUE
      INOP(K)=2
      DO 130 I=1,2
  130 ITOP(K,I)=IEQP(K,I)
      GO TO 40
C
C
   80 CONTINUE
      INOP(K)=3
   95 CONTINUE
      DO 140 I=1,INOP(K)
  140 ITOP(K,I)=IEQP(K I)
      GO TO 40
C
   90 CONTINUE
      INOP(K)=IEQP(K,8)
      IF(EQP(K,20) NE 1) GO TO 41
      DO 150 II=1 INOP(K)
  150 ITOP(K,II)=IEQP(K,II)
      GO TO 40
   41 IF(EQP(K,20) EQ 2) GO TO 80
      INOP(K)=2
      IF(EQP(K,20) EQ 3) GO TO 95
      WRITE(6,65) K,EQP(K,20)
   65 FORMAT(/,5X,"ERROR INPUT PARAMETER"
     2          /,5X,"EQP(",I2,",20) =",I3,
     3          /,5X,"SHOULD BE 1,2, OR 3 !!!",/)
      STOP
   40 CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE SIMMAN(NSIG,MS)
C
C           THIS SUBROUTINE PREPARES THE SYSTEM OF
C           NONLINEAR EQUATIONS FOR SOLUTION
C
      REAL X(200) F(200) WK(6400)
      EXTERNAL FCN
      COMMON /S1/ NSTR NS1(30),FLOW(30),MN,CNAME(20) COMP(30,20)
      COMMON /SIM2/ IFK(30),ICK(30 20),IEK(20,20),ITOP(20,7)
      COMMON /SIM3/ ID(20),ICC(30) IEC(20),INOP(20),CX(30,20)
C
C
      SH=0
      DO 100 I=1,NSTR
      IF(IFK(I) EQ 1) FLOW(I)=CX(I,20)
  100 IF(FLOW(I) GT SH) SH=FLOW(I)
      DO 110 I=1,NSTR
  110 FLOW(I)=FLOW(I)/SH
C
      DO 30 I=1,NSTR
   30 IF(IFK(I) EQ 1) CX(I,20)=CX(I,20)/SH
C
C
      N=200
      CALL IDEN
      CALL VARIN(X,II)
      CALL FUVAL(F,IT1,N)
      IF(IT EQ IT1) GO TO 120
      WRITE(6,200) IT,IT1
      STOP
  120 N=IT-1
      ITMAX=300
      WRITE(6,201) N
      WRITE(6,204) SH
      CALL WRITES
      CALL WRITEE
      CALL MPOLM(FCN,X,F N,WK,ITMAX,NSIG,MS)
      WRITE(6,203) ITMAX
  203 FORMAT(/,5X,"NUMBER OF ITERATIONS =" I4 /
     1          ,5X,"----------------------------",/)
      DO 300 I=1,NSTR
  300 FLOW(I)=FLOW(I)*SH
      WRITE(6,202)
      CALL WRITES
      CALL WRITEE
  200 FORMAT(/,5X,"ERROR ON INPUT",/,5X,"NUMBER OF VARIABLES=" I3
     1          /,5X,"NUMBER OF EQUATIONS=",I3,/)
  201 FORMAT(/,5X,"NUMBER OF VARIABLES = ",I3,
     1          /,5X,"THE SYSTEM TO BE SOLVED WILL BE PRINTED NOW"
     2          /,5X,"-------------------------------------------",/)
  202 FORMAT(/,5X,"THE SOLUTION TO THE PROBLEM IS   ",
     1          /,5X,"-------------------------------",/)
  204 FORMAT(/,5X,"ALL FLOW RATES HAVE A SCALLING "
     1          /,5X,"FACTOR = ",F10.4,/,
     2          /,5X,"-------------------------------",/)
      RETURN
      END
C
C
C
C
      SUBROUTINE FCN(X,F,N)
C
      REAL X(N),F(N)
      CALL VAROUT(X,IT,N)
      CALL FUVAL(F,IT,N)
      RETURN
      END
C
C
C
C
      SUBROUTINE FUVAL(F,IT,N)
      COMMON /S1/ NS NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE NEQP(20),EQNAME(20),EQP(20 20) IEQP(20,8)
      COMMON /SIM2/ IFK(30),ICK(30,20),IEK(20,20) ITOP(20,7)
      COMMON /SIM3/ ID(20),ICC(30),IEC(20),INOP(20),CX(30,20)
      REAL F(N)
      IT=1
C
      DO 1 K=1,NE
C
      GO TO (10,20,30,40,50,60),ID(K)
   10 CALL SSPLIT(K,N F,IT)
      GO TO 1
   20 CALL SMIXER(K N,F,IT)
      GO TO 1
   30 CALL SREAC(K N F,IT)
      GO TO 1
   40 CALL SSEPAR(K N F IT)
      GO TO 1
   50 CALL SFLASH(K N F IT)
      GO TO 1
   60 CALL USERI(K N F IT)
    1 CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE VAROUT(X,IT,N)
      COMMON /S1/ NS NSTR(30),FLOW(30),NC CNAME(20) COMP(30 20)
      COMMON /E1/ NE NEQP(20),EQNAME(20),EQP(20 20) IEQP(20 8)
      COMMON /SIM2/ IFK(30),ICK(30 20) IEK(20 20) IIOP(20 7)
      COMMON /SIM3/ ID(20),ICC(30),IEC(20),INOP(20) CX(30,20)
      REAL X(N)
      INTEGER ITER(30)
C
      IT=1
      DO 5 I=1,NS
      ITER(I)=0
    5 IF(IFK(I) EQ 1 AND ICC(I) EQ NC) ITER(I)=1
C
      OO 1 K=1,NE
      OO 100 I=1,INOP(K)
      L1=ITOP(K,I)
      IF(ITER(L1) EQ 1) GO TO 100
      ITER(L1)=1
      IF(IFK(L1) EQ 1) GO TO 105
      FLOW(L1)=X(IT)
      IT=IT+1
  105 CONTINUE
      DO 111 J=1,NC
      L=ICK(L1,J)
      IF(L EQ J) GO TO 111
      COMP(L1,J)=X(IT)
      IT=IT+1
  111 CONTINUE
  100 CONTINUE
C
      IF(IEC(K) EQ 0) GO TO 1
      DO 200 I=1,IEC(K)
      L1=IEK(K,I)
      EQP(K,L1)=X(IT)
  200 IT=IT+1
    1 CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE VARIN(X,IT)
      COMMON /S1/ NS NSTR(30),FLOW(30),NC CNAME(20) COMP(30 20)
      COMMON /E1/ NE NEQP(20),EQNAME(20),EQP(20,20) IEQP(20 8)
      COMMON /SIM2/ IFK(30),ICK(30,20),IEK(20 20) IIOP(20,7)
      COMMON /SIM3/ ID(20) ICC(30) IEC(20),INOP(20) CX(30,20)
      REAL X(*)
      INTEGER ITER(30)
C
      IT=1
      OO 5 I=1,NS
      ITER(I)=0
    5 IF(IFK(I) EQ 1 AND ICC(I) EQ NC) ITER(I)=1
C
      OO 1 K=1,NE
      OO 100 I=1,INOP(K)
      L1=ITOP(K,I)
      IF(ITER(L1) EQ 1) GO TO 100
      ITER(L1)=1
      IF(IFK(L1) EQ 1) GO TO 105
      X(IT)=FLOW(L1)
      CX(L1,20)=IT
      IT=IT+1
  105 CONTINUE
      DO 110 J=1,NC
      L=ICK(L1,J)
      IF(L EQ J) GO TO 110
      X(IT)=COMP(L1,J)
      CX(L1,J)=IT
      IT=IT+1
  110 CONTINUE
```

```
  100 CONTINUE
C
      IF(IEC(K) EQ 0) GO TO 1
C
       ICK(K,20)=IT
      DO 200 I=1,IEC(K)
      L1=IEK(K I)
      X(IT)=EQP(K,L1)
  200 IT=IT+1
    1 CONTINUE
      RETURN
      ENO
C
C
C
      SUBROUTINE SFLASH(NE,N,F,IT)
      COMMON /S1/ NS,NSTR(30),FLOW(30) NC,CNAME(20) COMP(30 20)
      COMMON /E1/ N1,NEQP(20),EQNAME(20),EQP(20,20) IEQP(20 8)
      REAL F(N)
      SUM2=0
      SUM3=0
      L1=IEQP(NE,1)
      L2=IEQP(NE,2)
      L3=IEQP(NE,3)
      DO 1 I=1,NC
      FL1=FLOW(L1)*COMP(L1,I)
      FL2=FLOW(L2)*COMP(L2,I)
      FL3=FLOW(L3)*COMP(L3 I)
      F(IT)=FL1-FL2-FL3
    1 IT=IT+1
      OO 2 I=1,NC
      SUM2=SUM2+COMP(L2,I)
      SUM3=SUM3+COMP(L3,I)
      F(IT)=COMP(L2,I)-EQP(NE,I)*COMP(L3,I)
    2 IT=IT+1
      F(IT)=1 -SUM2
      F(IT+1)=1 -SUM3
      IT=IT+2
      RETURN
      ENO
C
C
C
C
      SUBROUTINE SMIXER(NE,N,F,IT)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ N1,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20 8)
      REAL F(N)
      L8=IEQP(NE,8)
      L7=IEQP(NE,7)
      SUM1=0
      DO 1 I=1,NC
      SUM1=SUM1+COMP(L7,I)
      SUM=FLOW(L7)*COMP(L7,I)
      DO 2 L=1,L8
      L1=IEQP(NE,L)
    2 SUM=SUM-FLOW(L1)*COMP(L1,I)
      F(IT)=SUM
    1 IT=IT+1
      F(IT)=1 -SUM1
      IT=IT+1
      RETURN
      END
C
C
C
C
      SUBROUTINE SREAC(NE N,F,IT)
      COMMON /S1/ NS,NSTR(30),FLOW(30) NC,CNAME(20) COMP(30 20)
      COMMON /E1/ N1,NEQP(20),EQNAME(20),EQP(20,20) IEQP(20 8)
      REAL F(N)
      L8=IEQP(NE,8)
      L1=IEQP(NE,1)
      L2=IEQP(NE,2)
      SUM=0
      R=FLOW(L1)*COMP(L1,L8)*EQP(NE,20)/(-EQP(NE,L8))
      DO 1 I=1,NC
      SUM=SUM+COMP(L2 I)
      F1=FLOW(L1)*COMP(L1,I)
      F2=FLOW(L2)*COMP(L2,I)
      F3=F2-F1
      F(IT)=F3-EQP(NE,I)*R
    1 IT=IT+1
      F(IT)=1 -SUM
      F(IT)=1 -SUM2
      F(IT+1)=1 -SUM3
      IF(L4 NE 0) THEN
      F(IT+2)=1-SUM4
      IT=IT+3
      ELSE
      IT=IT+2
      ENDIF
      RETURN
      END
      SUBROUTINE SSPLIT(NE N,F,IT)
      COMMON /S1/ NS NSTR(30),FLOW(30),NC CNAME(20),COMP(30 20)
      COMMON /E1/ N1,NEQP(20),EQNAME(20),EQP(20 20) IEQP(20 8)
      COMMON /SIM3/ ID(20),ICC(30),IEC(20),INOP(20),CX(30 20)
      REAL F(N),SUM(8)
      OO 1 I=1,8
    1 SUM(I)=0
      L1=IEQP(NE 1)
      L8=IEQP(NE,8)
      SUM(1)=FLOW(L1)
      OO 2 I=2,L8+1
      SUM(8)=SUM(8)+EQP(NE I)
      L3=IEQP(NE,I)
      F2=FLOW(L3)
      SUM(1)=SUM(1) F2
      F(IT)=SUM(1)
    2 CONTINUE
      IT=IT+1
      DO 3 I=2,L8
      L2=IEQP(NE,I)
      F(IT)=FLOW(L2) FLOW(L1)*EQP(NE,I)
    3 IT=IT+1
C
      DO 4 I=2,L8+1
      L2=IEQP(NE,I)
      DO 4 L=1,NC
      F(IT)=COMP(L1 L)-COMP(L2,L)
    4 IT=IT+1
C
      IF(IEC(NE) EQ 0) GO TO 5
      F(IT)=1 -SUM(8)
      IT=IT+1
    5 CONTINUE
      RETURN
      END
      SUBROUTINE READ(NST,NEQ)
      OIMENSION FNAME(8)
      COMMON /S1/ NS NSTR(30),FLOW(30) NC,CNAME(20) COMP(30 20)
      COMMON /E1/ NE NEQP(20),EQNAME(20),EQP(20,20),IEQP(20 8)
      LL=0
      NS=NST
      NE NEQ
      READ(5 2) (FNAME(I) I=1,8),ID
    2 FORMAT(8A10/I2)
      WRITE(6,3) (FNAME(I),I=1,8)
    3 FORMAT(1H1//3X 8A10)
   50 LL=LL+1
      IC=10*(LL-1)+1
      IE=10*LL
      READ(5,30) NY (CNAME(J),J=IC,IE)
   30 FORMAT(I2,8X 10(A5 2X))
      OO 5 I=1,NS
    5 READ(5,10) NSTR(I),FLOW(I) (COMP(I,J),J=IC IE)
   10 FORMAT(I2,F8 2 10F7 4)
      READ(5,15) ID
   15 FORMAT(I2)
      IF(NY GT 10 AND LL EQ 1) GO TO 50
      OO 20 I=1,NE
   20 READ(5,25) NEQP(I) EQNAME(I),(EQP(I,J),J=1 20) (IEQP(I J) J=1 8)
   25 FORMAT(I2,1X,A10,/ 10G8 3,/,10G8 3,/,8I8)
      NC=NY
      RETURN
      END
      SUBROUTINE FLASH(N)
      DIMENSION EK(20) X1(20) X2(20) X3(20)
      COMMON /S1/ NS NSTR(30) FLOW(30),NC,CNAME(20) COMP(30 20)
      COMMON /E1/ NE NEQP(20),EQNAME(20),EQP(20 20) IEQP(20 8)
      NT=200
      A0=1 0
      EPS=1 0E-05
      N1=IEQP(N,1)
      FN1=FLOW(N1)
      FEED=FN1
      CALL CHECKS(N1)
```

```
      IT=IT+1
      RETURN
      END
C
C
C
C
      SUBROUTINE SSEPAR(NE,N,F,IT)
      COMMON /S1/ NS NSTR(30) FLOW(30) NC CNAME(20),COMP(30 20)
      COMMON /E1/ N1,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
      REAL F(N)
      SUM2=0
      SUM3=0
      L1=IEQP(NE,1)
      L2=IEQP(NE,2)
      L3=IEQP(NE,3)
      OO 1 I=1,NC
      SUM2=SUM2+COMP(L2,I)
      SUM3=SUM3+COMP(L3,I)
      F1=FLOW(L1)*COMP(L1,I)
      F2=FLOW(L2)*COMP(L2,I)
      F3=FLOW(L3)*COMP(L3,I)
      F(IT+NC)=F1*EQP(NE I)-F2
      F(IT)=F1-F2-F3
    1 IT=IT+1
      IT=IT+NC
      F(IT)=1 -SUM2
      F(IT+1)=1 -SUM3
      IT=IT+2
      RETURN
      END
C
C
C
C
      SUBROUTINE SSPLIT(NE,N F,IT)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30 20)
      COMMON /E1/ N1,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20 8)
      COMMON /SIM3/ ID(20),ICC(30),IEC(20),INOP(20) CX(30 20)
      REAL F(N),SUM(8)
      OO 2 I=1,8
    2 SUM(I)=0
      L1=IEQP(NE 8)
      L2=IEQP(NE,1)
      OO 1 I=2 L1+1
      SUM(1)=SUM(1)+EQP(NE,I)
      L3=IEQP(NE,I)
      F1=EQP(NE,I)*FLOW(L2)
      F2=FLOW(L3)
      OO 1 L=1,NC
      SUM(I)=SUM(I)+COMP(L3 L)
      F(IT)=F1*COMP(L2,L)-F2*COMP(L3,L)
    1 IT=IT+1
      IF(IEC(NE) EQ 0) GO TO 3
      F(IT)=1 -SUM(1)
      IT=IT+1
    3 CONTINUE
      DO 4 I=2,L1+1
      F(IT)=1 -SUM(I)
    4 'T=IT+1
      RETURN
      END
C
C
C
C
      SUBROUTINE JACOBI(X,F,N)
      EXTERNAL USER1
      INTEGER IP(30) IH(30)
      REAL X(N),F(N) FN(200)
      COMMON /S1/ NS NSTR(30),FLOW(30) NC CNAME(20) COMP(30 20)
      COMMON /E1/ NE,NEQP(20) EQNAME(20) EQP(20,20) IEQP(20 8)
      COMMON /SIM2/ IFK(30),ICK(30,20) IEK(20 20) IIOP(20 7)
      COMMON /SIM3/ ID(20) ICC(30) IEC(20),INOP(20) CX(30 20)
      COMMON /SP2/ IZ(200),B(200,70),Q(200),IC(200 70)
      IT=1
      DO 2 I=1,N
    2 IZ(I)=1
      DO 1 I=1,NE
      GO TO(10,20 30 40,50 60),ID(I)
   10 CALL JSPL(I IT)
      GO TO 1
   20 CALL JMB(I,IT)
      IS=IEQP(I 7)
```

MPDLM

VERSION 2

```
C
C
C------------------------------------------------------------------------
C
C
      SUBROUTINE MPDLM(FCN,X,F,N,B,ITMAX,IDGT,MS)
C
C       A SUBROUTINE TO FIND THE SOLUTION OF " N " NONLINEAR
C       EQUATIONS OF THE FORM
C
C       F(X1,X2,    ,XN)=0
C
C       THE SUBROUTINE USES A MODIFICATION OF POWELL'S DOGLEG
C       METHOD AS PROPOSED BY CHEN AND STADTHERR IN " COMP
C       CHEM  ENG ,5,143(1981) "
C       THERE WERE A FEW MODIFICATIONS IN THE METHOD PROPOSED
C       BY CHEN AND STADTHERR THE MOST IMPORTANT OF ALL
C       BEING THE UPDATE OF THE JACOBIAN BY SCHUBERT'S METHOD
C       INSTEAD OF BROYDEN'S METHOD
C
C       PARAMETERS
C
C       FCN      A SUBROUTINE USED TO EVALUATE FUNCTIONS VALUE  THE
C                SUBROUTINE MUST BE DECLARE IN A EXTERNAL STATEMANT
C                IN THE CALLING PROGRAM  THE SUBROUTINE HAS TO BE IN
C                THE FORM   SUBROUTINE FCN(X F N)
C
C       X        A N VECTOR OF INITIAL GUESSES ON  INPUT WHICH ARE
C                SPECIFIED BY THE USER  ON OUTPUT, X CARRIES THE BEST
C                ESTIMATE OF THE SOLUTION
C
C       F        A N-VECTOR WHICH " PASSES " THE VALUE OF EACH FUNC
C                TION EVALUATED WITH THE X VECTOR
C
C       N        NUMBER OF EQUATIONS BEING SOLVED (LIMITED TO 200)
C
C       WK       A WORK VECTOR OF DIMENSION 32*N
C
C       ITMAX    MAXIMUM NUMBER OF ITERATIONS ALLOWED
C
C       NSIG     NUMBER OF DIGITS OF ACCURACY
C
C       MS       PARAMETER USED TO SPECIFY THE OPTION OF AUTO SCALING
C                MS=0 NO SCALLING IS PERFORMED
C                MS=2 SCALLING (WE RECOMMEND THE USE OF AUTO SCALLING)
C                MS=3 THE JACOBIAN WILL BE EVALUATED THROUGH
C                AN EXTERNAL SUBROUTINE = JACOBI(X,F N B)
C                THE USER MUST PROVIDE THE SUBROUTINE  THE SUB
C                ROUTINE MUST USE THE COMMON BLOCK /ASNO/
C                COMMON BLOCK /ASNO/ SHOULD CONTAIN THE NUMBER OF ELEMENTS
C                IN EACH ROW OF B (VECTOR IZ(I))  ARRAY B HAS THE JACOBIAN IN A
C                COMPRESSED FORM WHERE ONLY NON-ZERO ELEMENTS ARE
C                STORED  ARRAY IC(I,L) CONTAINS THE COLLUMN OF THE
C                LTH ELEMENT OF B
C
C
C             COMMON BLOCK /SETA/ IS USED TO "PASS" THE PARAMETER IR
C             IR I AN ESTIMATIVE OF HOW FAR FROM THE ACTUAL SOLUTION
C             THE INITIAL GUESSES ARE  IR IS GIVEN AS A PERCENTAGE  EX
C             IR=70 MEANS THAT THE ACTUAL SOLUTION IS IN A INTERVAL OF
C             X+0 7*X   X-0 7*X     DEFAULT VALUE IS 50%
C
C
      COMMON /ASNO/ IC(200,32),INZ(200)
      REAL X(N),F(N) B(N,32),DF(200),PK(200) PS(200)
      REAL G(200) FN(200),CG(9)
      COMMON /SP2/ IZ(200) DB(200,70) Q(200) ICC(200 70)
      REAL LAMB
      COMMON /IP1/ IP(200)
      COMMON /SETA/ IR
      IF(IR LT 0 OR IR GT 100) IR=50
C
C          INITIALIZE PARAMETERS
C
C
      IF(MS NE 1) THEN
      DO 1 I=1,N
    1 DF(I)=1
      ELSE
      ENDIF
C
      DGT=1
      DO 2 I=1,IDGT
    2 DGT=DGT/10
C
      DGT1=SQRT(DGT)
C
C          HJ IS THE MACHINE EPSLON
C
      HJ=6 E-8
      PA1=HJ*SQRT(128 )
      PA2=1 /128
C
      IM=MAX(10,(N+4))
      ICON=0
C
      DO 3 I=1,9
    3 CG(I)=1
      IDEL=0
C
C          EVLUATION OF THE JACOBIAN
C
C
      JJJ=0
      IF(MS GE 3) THEN
      JJJ=1
      IF(MS EQ 4) JJJ=2
      MS=2
      ELSE
      END IF
C
  100 CONTINUE
      CALL FCN(X,F,N)
      DO 4 I=1,N
      IF(ABS(F(I)) LT 1E-14) F(I)=0
      INZ(I)=0
      IZ(I)=0
      DO 4 L=1,70
      ICC(I,L)=0
      DB(I,L)=0
      IF(L GT 32) GO TO 4
      IC(I,L)=0
      B(I,L)=0
    4 CONTINUE
C
      IF(JJJ GE 1) THEN
      CALL JACOBI(X,F,N)
      DO 101 I=1,N
      INZ(I)=IZ(I)
      DO 101 L=1,IZ(I)
      IC(I,L)=ICC(I,L)
      B(I,L)=DB(I,L)
  101 CONTINUE
      GO TO 102
      ELSE
      ENDIF
C
      DO 10 I=1,N
      HMU=ABS(X(I))
      PA3=MAX(HMU,PA2)
      PA4=PA1*PA3
      TEMP=X(I)
      XD=X(I)+PA4
      PA4=XD-TEMP
      X(I)=XD
      CALL FCN(X,FN,N)
      X(I)=TEMP
      DO 11 L=1,N
      IF(ABS(FN(L)).LT.1E-14) FN(L)=0
      CDF=FN(L)-F(L)
      IF(ABS(CDF).LT.1E-14) CDF=0
      CJC=CDF/PA4
      CJ1=ABS(CJC)
      IF(CJ1.LT.1.E-13) GO TO 11
      IF(ABS(1.-CJ1).LT.1.E-4) CJC=CJC/CJ1
      IZ(L)=IZ(L)+1
      INZ(L)=INZ(L)+1
      ICC(L,IZ(L))=I
      IC(L,INZ(L))=I
      B(L,INZ(L))=CJC
   11 CONTINUE
   10 CONTINUE
C
  102 CONTINUE
C
C
      II=0
C
      CALL ARR1(N)
C
      IFLAG=0
      JFLAG=1
      TAU=1
C
C                     SCALING OF THE FUNCTIONS
C
      IF(MS.EQ.2) THEN
      DO 15 I=1,N
      HOLD=0.0
      DO 16 L=1,INZ(I)
   16 IF(ABS(B(I,L)).GT.HOLD) HOLD=ABS(B(I,L))
      DF(I)=1./HOLD
   15 CONTINUE
      ELSE
      ENDIF
      DO 20 I=1,N
      G(I)=0
      F(I)=F(I)*DF(I)
      IZ(I)=INZ(IP(I))
      DO 20 L=1,INZ(I)
   20 B(I,L)=B(I,L)*DF(I)
      IS=2
      LB=N
      PNJ=EUCN(F,N)
C
C            CALCULATE INITIAL STEP BOUND
C
      IF(ICON.EQ.0) THEN
      HOLD=EUCN(X,N)
      HOLD=HOLD*TR/100
      DELTA=MIN(HOLD,10.)
      ELSE
      DELTA=SEARCH
      ENDIF
C
      DO 9 I=1,N
      DO 9 L=1,INZ(IP(I))
      ICC(I,L)=IC(IP(I),L)
    9 DB(I,L)=B(IP(I),L)
C
C
  150 CONTINUE
      DO 103 I=1,N
      WRITE(7,105) INZ(I)
      WRITE(7,105)(IC(I,L),L=1,INZ(I))
      WRITE(7,104)(B(I,L),L=1,INZ(I))
  103 CONTINUE
      WRITE(7,104) ( F(I),I=1,N)
      IF(JJJ.EQ.2) STOP
  104 FORMAT(10G12.6)
  105 FORMAT(10I12)
C
      DELHOL=DELTA
C
C            SOLVE LINEAR SYSTEM
C
      DO 21 I=1,N
   21 Q(I)=-F(IP(I))
C
C
C
C
      CALL SPAMA2(N,IS,LB,PK)
C
C
C
C
      IF(IS.EQ.-1) THEN
      PRINT*,'  THE JACOBIAN HAS A ZERO IN THE DIAGONAL'
      PRINT*,'  CHANGE THE ORDER OF THE EQUATIONS'
      PRINT*,'    LB = ',LB
      STOP
      ELSE
      END IF
      IS=3
C
C             CALCULATE CORRECTION STEP (PK)
C
      DO 7 I=1,N
      IF(ABS(PK(I)).LT.1E-14) PK(I)=0.0
    7 Q(I)=PK(I)
  160 PC1=EUCN(PK,N)
C
      WRITE(7,106) ICON
      WRITE(7,104) (Q(L),L=1,N)
      WRITE(7,107) PC1,DELTA
  106 FORMAT(5X,"-----",I2,"----- Q SOLUTION")
  107 FORMAT(5X,"PC1=",G10.4,"DELTA=",G10.4)
C
      IRV=0
      IF(PC1.LE.DELTA) THEN
      IRV=1
      GO TO 29
      ELSE
      ENDIF
C
      DO 5 I=1,N
    5 G(I)=0
C
      DO 22 I=1,N
      DO 22 L=1,INZ(I)
   22 G(IC(I,L))=G(IC(I,L))-B(I,L)*F(I)
C
      PC2=EUCN(G,N)
      PC3=PC2*PC2
      DO 23 I=1,N
      PS(I)=0
      DO 23 L=1,INZ(I)
   23 PS(I)=PS(I)+B(I,L)*G(IC(I,L))
C
      PC4=EUCN(PS,N)
      PC5=PC4*PC4
      PC6=PC3/PC5
      DO 24 I=1,N
   24 PS(I)=G(I)*PC6
      PD1=EUCN(PS,N)
      IF(PD1.LT.DELTA) GO TO 28
      PD2=DELTA/PC2
      DO 25 I=1,N
   25 PK(I)=PD2*G(I)
      GO TO 29
   28 CONTINUE
C
C            EVALUATION OF ALFA
C
      PE1=PC1*PC1
      PE2=PD1*PD1
      PE3=0
      PE4=0
      DO 26 I=1 N
      PE3=PE3+(PK(I) PS(I))*PS(I)
   26 PE4=PE4+PK(I)*PS(I)
C
      PE5=DELTA*DELTA
      PE6=(PE1-PE5)*(PE5 PE2)+(PE4-PE5)*(PE4-PE5)
      PE7=SQRT(PE6)
      ALFA=(PE5-PE2)/(PE3+PE7)
      PF1=(1 -ALFA)
      DO 27 I=1,N
   27 PK(I)=ALFA*PK(I)+PF1*PS(I)
   29 CONTINUE
      PC1=EUCN(PK N)
C
C           EVALUATION OF F(XK+PK)
C
      DO 30 I=1,N
   30 X(I)=X(I)+PK(I)
      DGTX=EUCN(X N)
      HOLD=MAX(1  DGTX)
      DGTX=DGT*HOLD
C
      CALL FCN(X,FN N)
C
      PG1=EUCN(FN N)
      IF(PG1 LT DGT1 AND PC1 LT DGTX) THEN
      ITMAX=ICON
      RETURN
      ELSE
      ENDIF
C
      DO 31 I=1,N
      IF(ABS(FN(I)) LT 1E-14) FN(I)=0 0
   31 FN(I)=FN(I)*DF(I)
      PG2=EUCN(FN,N)
      PG3=EUCN(F N)
      SK1=PG2*PG2
      SK=PG3*PG3
      IF(SK1 LT (SK*0 999)) THEN
      II=II-1
      IF(II LT 0) II=0
      ELSE
      II=II+1
      ENDIF
      IF(IM LT II) THEN
      PRINT*,'  CONVERGENCE IS TOO SLOW '
      PRINT*,'  CHANGE INITIAL GUESSES '' 
      SSS=SSS/PP
      STOP
      ELSE
      ENDIF
C
C           CHECK WHETHER A NEW EALUATION
C           OF THE JACOBIAN IS NEEDED
C           IF IFLAG IS = TO 1 **AND**
C           F(XK) AS BEEN REDUCED BY A
C           FACTOR OF TWO A NEW JACOBIAN
C           IS EVALUATED (RENEWED)
C
C
      IFLAG=0
      DO 40 I=1,8
      HOLD=CG(I+1)
   40 CG(I)=HOLD
      CG(9)=SK
C
      PNJ1=PG2
      IF(PG2 LT 1E-14) PNJ1=1
      IF((PNJ/PNJ1) GE 2 ) THEN
      IF(ICON LT 10) THEN
      IF(II GT 3) GO TO 100
      ELSE
      R1=SK1/CG(5)
      R2=SQRT(CG(5)/CG(1))
      IF(R1 GT R2) GO TO 100
      ENDIF
      ELSE
      ENDIF
C
      IF(SK1 LT SK) GO TO 47
      DO 41 I=1 N
   41 X(I)=X(I) PK(I)
   47 CONTINUE
C
C
C        UPOATE DELTA
C
C
      IF(JFLAG EQ 1 AND SK1 GE SK) THEN
      PH2=0
      DO 42 I=1,N
   42 G(I)=0
      DO 43 I=1,N
      DO 43 L=1,INZ(I)
   43 G(IC(I,L))=G(IC(I L))+B(I,L)*F(I)
      DO 6 I=1,N
    6 PH2=PH2+G(I)*PK(I)
      LAMB=PH2/(SK+2 *PH2-SK1)
      PH3=MAX(0 1,LAMB)
      DELTA=PC1*PH3
C
      IF(SK1 LT SK AND IDEL EQ 0) THEN
      SEARCH=DELTA
      IDEL=1
      ELSE
      ENDIF
C
      IF(DELTA LT DELHOL) TAU=1
      IF(PG2 GT (1 5*PG3)) GO TO 160
      GO TO 200
      ELSE
      DO 44 I=1,N
      PH5=0
      DO 45 L=1,INZ(I)
   45 PH5=PH5+B(I L)*PK(IC(I L))
   44 G(I)=F(I)+PH5
      PH6=EUCN(G,N)
      PH4=PH6*PH6
      DM=SK-SK1-0 1*(SK-PH4)
      IF(DM LT 0) THEN
      DELTA=PC1/2
      TAU=1
      ELSE
      PTP=0
      PTS=0
      DO 46 I=1,N
      HOLD=FN(I)-G(I)
      PTS=HOLD*HOLD+PTS
      PTP1=ABS(FN(I)*HOLD)
   46 PTP=PTP+PTP1
      PJ1=PTP+SQRT(PTP*PTP+DM*PTS)
      LAMB=SQRT(1 +DM/PJ1)
      AMU=MIN(2 ,LAMB,TAU)
      TAU=LAMB/AMU
      DELTA=AMU*PC1
      IF(DELTA LT DELHOL) TAU=1
      ENDIF
      ENDIF
C
  200 CONTINUE
      JFLAG=0
      WRITE(7,108) PC1,DELTA,SK,SK1
  108 FORMAT(5X,"PC1=",G15 6,"DELTA=",G15 6,"SK=" G15 6 "SK1=" G15 6)
      IF(ICON GT ITMAX) GO TO 300
      ICON=ICON+1
C
C
C
      IF(SK1 LT SK AND IDEL EQ 0) THEN
      SEARCH=DELTA
      IDEL=1
      ELSE
      ENDIF
C
C           THE JACOBIAN WILL BE UPOATE BY
C           SCHUBERT'S ALGORITHM
C
      DO 50 I=1,N
      DO 51 I1=1 N
   51 G(I1)=0
C
      IF(ABS(Q(I)) LT HJ*HJ) THEN
      SS1=1
      ELSE
      SS1=PK(I)/Q(I)
      END IF
C
      DXT=0
      DO 52 I1=1 INZ(I)
      G(I1)=Q(IC(I I1))
   52 DXT=DXT+G(I1)*G(I1)
      DXT=DXT*SS1
C
      DFT=FN(I)-(1  SS1)*F(I)
      IF(ABS(DXT) GE HJ*HJ) THEN
      DMU=DFT/DXT
      ELSE
      DMU=1
      END IF
C
      DO 54 L=1,INZ(I)
   54 B(I,L)=B(I,L)+DMU*G(L)
   50 CONTINUE
C
      DO 715 I=1 N
      HOLD=0
      DO 720 L=1 INZ(I)
      SS1=ABS(B(I,L))
  720 IF(SS1 GT HOLD) HOLD=SS1
      DF(I)=DF(I)/HOLD
      FN(I)=FN(I)/HOLD
      DO 715 L=1 INZ(I)
  715 B(I,L)=B(I L)/HOLD
      IS=2
      DO 55 I=1 N
      F(I)=FN(I)
      IZ(I)=INZ(IP(I))
      DO 55 L=1,INZ(IP(I))
      ICC(I,L)=IC(IP(I) L)
   55 OB(I,L)=B(IP(I) L)
      GO TO 150
  300 CONTINUE
      PRINT*,'  NO CONVERGENCE IN ' ITMAX ' ITERATIONS'
      PRINT*,'  CHANGE INITIAL GUESSES OR USE  ANOTHER SUBROUTINE '
      STOP
      END
C
C
C
C
C
C          FUNCTION EUCN EVALUATES THE EUCLIDIAN NORM
C          OF A VECTOR OF DIMENSION " N "
C
C
      FUNCTION EUCN(Y J)
      REAL Y(J)
      SS=0
      DO 1 I=1,J
      SS=SS+Y(I)*Y(I)
    1 CONTINUE
      EUCN=SQRT(SS)
      RETURN
      END
C
C
C
C
C
      SUBROUTINE SPAMA2(N,IS LB X)
C
C      SUBROUTINE SPAMA2 SOLVES A SYSTEM OF " N " LINEAR EQUATIONS
C      THIS SUBROUTINE USES SPARSE MATRIX TECHNICS  THE RIGHT HAND
C      SIDE OF THE SYSTEM IS STORED IN VECTOR B   INZ(N) AS THE NUM
C      BER OF ELEMENTS IN EACH ROW OF THE SYSTEM  A(I L) CONTAINS
C      THE ELEMENTS OF THE MATRIX STORED IN A COMPRESSED FORM
C      THE VECTOR IC(I L) CONTAINS THE COLUMN OF EACH ELEMENT
C      IN MATRIX A
C
C      N        NUMBER OF EQUATIONS BEING SOLVED
C
C      LB       ON OUTPUT CONTAINS THE DIMENSION OF THE VECTOR AB
C
C      IS       OPTION OF SOLUTON  FOR IS=1 THE METHOD USED FOR
C               SOLUTION IS " RELATIVE TOLERANCE "   FOR IS=2
C               THE METHOD IS " PIVOTAL CONDENSATION "   THE OP
C               TION IS=3 IS USED WHEN A SYSTEM IS SOLVED SEVERAL
C               TIMES AND THE ONLY THE RIGHT HAND SIDE OF THE
C               SYSTEM IS CHANGED
```

```
C
C
      COMMON /ARR/ IY(200)
      COMMON /SP2/ INZ(200),A(200,70),B(200) IC(200,70)
      INTEGER IX(200),IZ(200)
      REAL X(200)
      COMMON /G2/ AB(3000) IBO(200),IBM(3000),IB(200)
      EPS=1 E-12
      IF=N
C
C
C
      IF(IS EQ 3) GO TO 999
      DO 1 I=1,N
      IB(I)=I
      IY(I)=I
      IBO(I)=0
    1 IZ(I)=I
      LB=0
C
C
      NZM=1
C
C           EVALUATION OF THE TOLERANCE OF THE SYSTEM
C
  205 AM=A(1,1)
      DO 206 I=1,N
      NZ=INZ(I)
      DO 206 K=1,NZ
      IF(ABS(AM) GE ABS(A(I,K))) GO TO 206
      AM=A(I,K)
  206 CONTINUE
      TOL=EPS*ABS(AM)
C
      IF(IS EQ 2) GO TO 2
      N1=N-1
      DO 10 I2=1 N1
      I=IZ(I2)
      L=1
      IX(1)=0
      AM=0
      N2=I+IF
      IF(N2 GT N) N2=N
      DO 11 J1=I2,N2
      J=IZ(J1)
      IF(IC(J,1) NE I2) GO TO 11
      L=L+1
      IX(1)=IX(1)+1
      IX(L)=J
      IF(ABS(A(J 1)) LE ABS(AM)) GO TO 11
      AM=A(J,1)
      NJ=J
   11 CONTINUE
      IF(ABS(AM) GT TOL) GO TO 12
      GO TO 101
    2 N1=N-1
      DO 20 I3=1,N1
      I=IZ(I3)
      L=1
      IX(1)=0
      N2=I+IF
      IF(N2 GT N) N2=N
      LLL=N
C
      DO 21 J1=I3,N2
      J=IZ(J1)
      IF(IC(J,1) NE I3) GO TO 21
      IF(INZ(J) LT LLL) LLL=J
      L=L+1
      IX(1)=IX(1)+1
      IX(L)=J
      IY(L)=J
   21 CONTINUE
      NJ=IX(2)
C
      IF(ABS(A(NJ 1)) GT TOL AND NJ LE LLL) GO TO 12
C
C
      KL=IX(1)+1
      DO 85 LJ1=2 KL
      LJ3=N
      LJL=LJ1
      HJ3=0
      LJH=LJ1
      DO 86 LJ2=LJ1,KL
      IF(INZ(IX(LJ2)) LT LJ3) THEN
      LJ3=INZ(IX(LJ2))
      LJL=LJ2
      ELSE
      ENDIF
      IF(ABS(A(IY(LJ2),1)) GT HJ3) THEN
      HJ3=ABS(A(IY(LJ2),1))
      LJH=LJ2
      ELSE
      ENDIF
   86 CONTINUE
      IAUX=IX(LJ1)
      IX(LJ1)=IX(LJL)
      IX(LJL)=IAUX
      IAY=IY(LJ1)
      IY(LJ1)=IY(LJH)
      IY(LJH)=IAY
   85 CONTINUE
C
      DO 87 LJ1=2,KL
      LJ3=40001
      LJL=LJ1
      DO 89 LJ2=LJ1,KL
      JMU=INZ(IX(LJ2))*INZ(IY(LJ2))
      IF(JMU LT LJ3) THEN
      LJ3=JMU
      LJL=LJ2
      ELSE
      ENDIF
   89 CONTINUE
      IAUX=IX(LJ1)
      IAUY=IY(LJ1)
      IX(LJ1)=IX(LJL)
      IY(LJ1)=IX(LJL)
      IX(LJL)=IAUX
      IY(LJL)=IAUY
   87 CONTINUE
C
      NZ=IX(1)+1
      DO 23 K=2,NZ
      NJ=IX(K)
      IF(ABS(A(NJ,1)) GT TOL) GO TO 15
   23 CONTINUE
      NJ=IX(2)
  101 IS=1
C
      IF(IX(1) EQ 0) GO TO 150
   12 IF(NJ EQ I) GO TO 14
C
C           CHANGE ROWS
C
   15 NZ=INZ(I)
      IF(INZ(NJ) GT NZ) NZ=INZ(NJ)
      DO 13 K=1,NZ
      AUX=A(I,K)
      A(I,K)=A(NJ,K)
      A(NJ,K)=AUX
      IAUX=IC(I,K)
      IC(I,K)=IC(NJ,K)
      IC(NJ,K)=IAUX
   13 CONTINUE
      IAUX=INZ(I)
      INZ(I)=INZ(NJ)
      INZ(NJ)=IAUX
      IB(I)=NJ
      AUX=B(I)
      B(I)=B(NJ)
      B(NJ)=AUX
   14 A1=A(I,1)
      A(I,1)=0
      LB=LB+1
C
C
      IF(LB GT 3000) THEN
      WRITE(6,88)
      STOP
      ELSE
      ENDIF
C
   88 FORMAT(//,10X,"SUBROUTINE SPAMAT CAN NOT SOLVE THIS PROBLEM"
     1        /,10X,"CHANGE MAIN PROGRAM PLACING THE FOLLOWING    "
     2       //,10X,"   COMMON /IMSL1/ IFLAG   ",4(/,15X,"     ")
     3        /,10X,"   IFLAG=1    "//,10X,"***** IMPORTANT *****"
     4        /,10X," BEFORE THE PROGRAM IS RUN ENTER      "
     5        /,10X," ATTACH IMSL/UN=LIBRARY "/,10X," LIBRARY IMSL "//)
C
C
C
      AB(LB)=A1
      IBO(I)=IBO(I)+1
      IBM(LB)=I
      B(I)=B(I)/A1
      NZ=INZ(I)
      DO 24 K=1,NZ
   24 A(I,K)=A(I,K)/A1
C
C           ELIMINATION OF ONE ROW
C
      NM=IX(1)+1
      IF(NM GE 3) GO TO 25
      IF(IS EQ 2) GO TO 20
      GO TO 10
   25 DO 40 L=3,NM
      I1=IX(L)
      A2=A(I1,1)
      A(I1,1)=0
      NZ=INZ(I)
      IF(NZ GT 1) GO TO 57
C
C           CHANGE ROWS
C
      IF(INZ(I1) EQ 1) GO TO 56
      K3=INZ(I1)-1
      DO 303 K4=1,K3
      IC(I1,K4)=IC(I1,K4+1)
      A(I1,K4)=A(I1,K4+1)
  303 CONTINUE
      INZ(I1)=INZ(I1)-1
      GO TO 56
   57 NN=IC(I,NZ)
      NZ1=INZ(I1)
      IF(IC(I1,NZ1) GT NN) NN=IC(I1,NZ1)
      I4=I2
      IF(IS EQ 2) I4=I3
      DO 50 K=I4,NN
   50 X(K)=0
      IF(NZ1 EQ 1) GO TO 55
      DO 51 K1=2,NZ1
      K=IC(I1,K1)
      X(K)=A(I1,K1)
   51 CONTINUE
C
C
   55 CONTINUE
      IF(NZ GT 70) THEN
      WRITE(6,88)
      STOP
      ELSE
      ENDIF
C
C
      DO 52 K1=2,NZ
      K=IC(I,K1)
      X(K)=X(K)-A2*A(I,K1)
   52 CONTINUE
      K1=0
      INZ(I1)=INZ(I1)-1
      I5=I2
      IF(IS EQ 2) I5=I3
      DO 53 K=I5,NN
      IF(X(K) EQ 0) GO TO 53
      K1=K1+1
CC
      IF(K1 GT 70) THEN
      WRITE(6,88)
      STOP
      ELSE
      ENDIF
C
      A(I1,K1)=X(K)
      IC(I1,K1)=K
   53 CONTINUE
      INZ(I1)=K1
   56 B(I1)=B(I1)-A2*B(I)
      IF(INZ(I1) GT NZM) NZM=INZ(I1)
      LB=LB+1
C
```

```
      IF(LB GT 3000) THEN
      WRITE(6 88)
      STOP
      ELSE
      ENDIF
C
      AB(LB)=A2
      IBO(I)=IBO(I)+1
      IBM(LB)=I1
   40 CONTINUE
      IF(IS EQ 1) GO TO 10
   20 CONTINUE
      GO TO 33
   10 CONTINUE
   33 IF(IS NE 3) GO TO 901
C
C
C
C
C
  999 LC1=1
      N1=N-1
      DO 100 I=1 N1
      LC2=LC1+IBO(I) 1
      J1=IBM(LC1)
      J=IB(J1)
      AUX=B(I)
      B(I)=B(J)
      B(J)=AUX
      B(I)=B(I)/AB(LC1)
  110 LC1=LC1+1
      IF(LC1 GT LC2) GO TO 100
      I1=IBM(LC1)
      A2=AB(LC1)
      B(I1)=B(I1) A2*B(I)
      GO TO 110
  100 CONTINUE
C
C
C
C
C
  901 J=IZ(N)
      NZ=INZ(J)
      DO 30 K=1,NZ
      IF(IC(J,K) NE 0) GO TO 32
   30 CONTINUE
      GO TO 150
   32 B(J)=B(J)/A(J,K)
      DO 90 J=2,N
      IBB=N-J+1
      I1=IZ(IBB)
      NZ=INZ(I1)
      IF(IC(I1,NZ) EQ 0) GO TO 90
      DO 91 K=1,NZ
      IA=IC(I1,K)
      J1=IZ(IA)
      B(I1)=B(I1)-A(I1,K)*B(J1)
   91 CONTINUE
   90 CONTINUE
      DO 94 I=1,N
      K=IZ(I)
   94 X(I)=B(K)
  150 RETURN
      END
C
C
C
C
      SUBROUTINE ARR1(N)
C
C      THIS SUBROUTINE ARRANGES THE LINEAR SYSTEM SOLVED BY
C      SPAMA  THE ARRANGEMENT PERFORMED LEAVES THE SYSTEM
C      WHITH THE TOP ROWS HAVING THE LOWEST POSSIBLE NUMBER
C      OF ELEMENTS
C
      COMMON /SP2/ INZ(200) A(200,70) Q(200) IC(200,70)
      COMMON /ARR/ IST(200)
      COMMON /IP1/ IP(200)
      DO 30 I=1,N
      IST(I)=1
   30 IP(I)=I
C
      DO 10 M=1,N
      J=0
      DO 1 I=M,N
      DO 11 L=1,INZ(I)
      IF(IC(I,L) EQ M) THEN
      J=J+1
      IST(J)=I
      GO TO 1
      ELSE
      END IF
   11 CONTINUE
    1 CONTINUE
C
   12 IGS=1
      K=N
      DO 101 I=1,J
  101 IF(INZ(IST(I)) LE K) K=INZ(IST(I))
      DO 2 I=1,J
      LR=IST(I)
      IF(INZ(LR) GT K) GO TO 2
      IF(IGS GT 1) GO TO 55
      IGS=IGS+1
      KR=LR
      GO TO 2
   55 IF(IC(LR,K) GT IC(KR K)) THEN
      GO TO 2
      ELSE
      KR=LR
      ENDIF
    2 CONTINUE
   29 KK=KR
      IAUX=IP(M)
      IP(M)=IP(KK)
      IP(KK)=IAUX
      IAUX=INZ(M)
      INZ(M)=INZ(KK)
      INZ(KK)=IAUX
      LL=MAX(INZ(KK) INZ(M))
      DO 4 I=1,LL
      IAUX=IC(M,I)
      IC(M,I)=IC(KK I)
    4 IC(KK,I)=IAUX
   10 CONTINUE
C
      RETURN
      END
C
C
C-----------------------------------------------------------------------  C
C
C
```

SIMO LIBRARY

```
C
C
C     THIS COMPUTER PACKAGE CONTAINS A SELECTION OF NONLINEAR
C     EQUATION SOLUTION TECHNIQUES WITH AUTOMATIC OUTPUT OPTION
C     AND A COLLECTION OF EQUIPMENT MODULES WHICH PERFORMS
C     THE MATERIAL BALANCE CALCULATIONS FOR FLASH, SEPARATOR
C     REACTOR  SPLITTER, AND MIXER  THE MODULES ARE INTERLINKED
C     THROUGH A COMMON BLOCK STRUCTURE WHICH CARRIES THE INPUT/
C     OUTPUT STREAM VARIABLES  A CONVERGENCE MODULE IS ALSO
C     INCLUDED WHICH USES A MODIFICATION OF NEWTON'S METHOD
C     TECHNIQUE
C
       SUBROUTINE SECNEW(X,NT,EPS,SUB)
      REAL G(10)
      DO 1 I=1,10
    1 G(I)=I
      NTT=0
C
C            CALCULATE DERIVATIVE
C
    5 CALL SUB(X,F)
      FODER=F
      X1=ABS(X)
      H=MAX(X1,0 0078)
      HJ=EPS*10 *H
      XN=X+HJ
      CALL SUB(XN,FN)
      DER=(FN-F)/HJ
      ICON=0
      XN=X-F/DER
   10 CALL SUB(XN,FN)
      IF(ABS(FN) LE EPS) THEN
      X=XN
      NT=NTT+1
      RETURN
      ELSE
      ENDIF
      W2=FN*FN
      W3=F*F
      IF(W2 LT W3* 999) THEN
      ICON=ICON-1
      IF(ICON LT 0) ICON=0
      ELSE
      ICON=ICON+1
      ENDIF
      W1=ABS(FODER/FN)
      NTT=NTT+1
      IF(NTT GT NT) THEN
      WRITE(6,20) NT,X,F
   20 FORMAT(/,10X,"NO COVERGENCE ACHIEVED IN ",I4 "  ITERETIONS "
     1          /,10X,"LAST VALUE OF X=",F10 4
     2          /,10X,"LAST VALUE OF F =",F10 4,/)
      STOP
      ELSE
      ENDIF
      DO 2 I=1,9
    2 G(I)=G(I+1)
      G(10)=FN
      IFLAG=0
      IF(NTT GT 9) THEN
      VAL1=ABS(G(10)/G(6))
      VAL2=ABS(G(6)/G(1))
      IF(VAL1 GT VAL2) IFLAG=1
      ELSE
      ENDIF
      IF(ICON GT 3) IFLAG=1
      IF(IFLAG EQ 1 AND ABS(W1) GE 2) GO TO 5
      DER=DER-(FN*DER)/F
      X=XN
      XN=XN-FN/DER
      F=FN
      GO TO 10
      END
C
C
      SUBROUTINE NEWTON(X NT,EPS SFNC K)
C
C     A SUBROUTINE TO FIND THE ROOT OF A NONLINEAR FUNCTION
C     F(X)=0 USING THE NEWTON'S METHOD
C
C     X         INITIAL GUESS OF THE ROOT ON INPUT  BUT THE BEST
C               ESTIMATE OF THE ROOT ON OUTPUT
C     NT        TOTAL NUMBER OF ITERATIONS ALLOWED
C     EPS       RELATIVE ERROR CRITERION FOR CONVERGENCE
C               FOR 4-DIGIT ACCURACY SPECIFY EPS=0 0001
C     SFNC      NAME OF THE SUBROUTINE THAT CALCULATES THE FUNCTION
C               VALUE "F" AND THE DERIVATIVE "FD" AT X  USER MUST
C               PROVIDE SUBROUTINE SNFC(X,F,FD) AND PUT THE ACTUAL
C               NAME OF THE SUBROUTINE IN EXTERNAL STAMENT WHICH MUST
C               BE LOCATED IN THE PROGRAM THAT CALLS NEWTON
C     K         A USER SPECIFIED PARAMETER TO CONTROL THE PRINTING OF
C               ITERATION RESULTS  EVERY K TH  ITERATION IS PRINTED
C               NO PRINTING FROM THE SUBROUTINE NEWTON IF K=0
C
C
      IF(K LT 1) GO TO 50
      WRITE(6,20)
   20 FORMAT(//,5X,"ITERATION RESULTS FOR NEWTON'S METHOD  " /
     1 4X,"ITER",8X,"X",13X,"F",13X,"FD")
      CALL SFNC(X,F,FD)
      WRITE(6,25) 0,X,F,FD
   25 FORMAT(2X,I5,3E14 4)
      GO TO 60
   50 CALL SFNC(X,F FD)
   60 J=0
      DO 100 I=1,NT
      J=J+1
      XN=X-F/FD
      XDEN=X
      IF(ABS(X) LT 1 0E-10) XDEN=SIGN(1 0E-10,X)
      E=ABS((XN-X)/XDEN)
      IF(E LT EPS) GO TO 120
      X=XN
      CALL SFNC(X F FD)
      IF(K LT 1) GO TO 100
      IF(J LT K) GO TO 100
      WRITE(6,25) I,X F FD
      J=0
  100 CONTINUE
      WRITE(6,110) NT X,F
  110 FORMAT(//,2X,"NO CONVERGENCE IN",I5,3X,"ITERATIONS",
     1 /5X,"X =",E14 4 5X,"F =",E14 4)
      STOP
  120 X=XN
      IF(K LT 1) RETURN
      WRITE(6 25) I X F,FD
      RETURN
      END
C
C
C
      SUBROUTINE INTHLV(XL,XR,X,N,FNC,K)
C
C     A SUBROUTINE TO FIND THE ROOT OF A NONLINEAR EQUATION F(X)=0
C     USING THE INTERVAL-HALVING (HALF INTERVAL) TECHNIQUE
C
C     XL        USER SPECIFIED LEFT HAND BOUND ON THE ROOT
C     XR        USER SPECIFIED RIGHT HAND BOUND ON THE ROOT
C               ON RETURN BOTH XL AND XR ARE REPLACED WITH THE FINAL
C               BOUNDS ON THE ROOT  THIS REFLECTS FINAL ACCURACY
C     X         BEST ESTIMATE OF THE ROOT ON OUTPUT
C     N         NUMBER OF ITERATIONS  SINCE THIS TECHNIQUE IS
C               GUARANTEED TO FIND THE SINGLE ROOT IN (XL XR) WITH A
C               CERTAIN DEGREE OF ACCURACY WHICH DEPENDS ON THE NUMBER
C               OF ITERATIONS, THE USER IS ASKED TO ESTIMATE N
C               BEFORE CALLING THE SUBROUTINE
C     FNC       THE NAME OF THE FUNCTION THAT CALCULATES F(X)
C               USER MUST PROVIDE FUNCTION FNC(X)
C               ACTUAL NAME OF FNC MUST BE DEFINED IN EXTERNAL
C     K         A USER SPECIFIED PARAMETER TO CONTROL THE PRINTING OF
C               ITERATION RESULTS  EVERY K TH ITERATION IS PRINTED
C               NO PRINTING FOR K=0
C
C
      FL=FNC(XL)
      FR=FNC(XR)
      IF(FL*FR LT 0 0) GO TO 20
      WRITE(6,10) XL FL XR,FR
   10 FORMAT(//,2X,"       XL         FNC(XL)      XR          FNC(XR)",/
     1 2X,4E10 3/5X "ERROR IN INPUT TO INTERVAL HALVING")
      STOP
C
   20 IF(K LT 1) GO TO 50
      WRITE(6,22)
   22 FORMAT(//5X,"ITERATION RESULTS FOR INTERVAL HALVING METHOD  "
     1 4X,"ITER",7X "XL",12X "FL" 12X,"XR",12X,"FR")
      WRITE(6,25) 0 XL,FL XR FR
   25 FORMAT(2X,I5,4E14 4)
```

```
C
   50 J=0
      DO 100 I=1 N
      J=J+1
      X=(XL+XR)/2
      FX=FNC(X)
      IF(FX*FR GT 0 0) GO TO 35
      FL=FX
      XL=X
      GO TO 40
   35 FR=FX
      XR=X
   40 IF(K LT 1) GO TO 100
      IF(J LT K) GO TO 100
      WRITE(6 25) I XL FL,XR FR
      J=0
  100 CONTINUE
      X=(XL+XR)/2
      RETURN
      END
C
C
C
      SUBROUTINE SUCSUB(X NT,EPS,FNC K)
C
C     A SUBROUTINE TO FIND THE SOLUTION OF A NONLINEAR EQUATION
C     IN THE FORM X=F(X) USING THE SUCCESSIVE SUBSTITUTION METHOD
C
C     X        INITIAL GUESS OF THE SOLUTION ON INPUT, BUT THE
C              BEST ESTIMATE OF THE SOLUTION ON OUTPUT
C     NT       NAXIMUM NUMBER OF ITERATIONS ALLOWED
C     EPS      RELATIVE ERROR BOUND AS STOPPING CRITERION
C     FNC      NAME OF THE FUNCTION THAT EVALUATES F(X)
C              USER MUST PROVIDE FUNCTION FNC(X) AND PUT THE
C              ACTUAL NAME OF FNC IN EXTERNAL STATEMENT
C     K        A USER SPECIFIED PARAMETER TO CONTROL THE PRINTING OF
C              ITERATIONS RESULTS  EVERY K TH ITERATION IS PRINTED
C              NO PRINTING FOR K=0
C
C
      IF(K LT 1) GO TO 50
      WRITE(6,20)
   20 FORMAT(//5X "ITERATION RESULTS FOR SUCCESSIVE SUBSTITUTION " /
     1 4X "ITER" 8X "X")
      WRITE(6 25) 0 X
   25 FORMAT(2X I5 E14 4)
C
   50 J=0
      DO 100 I=1 NT
      J=J+1
      XN=FNC(X)
      XDEN=X
      IF(ABS(X) LT 1 0E-10) XDEN=SIGN(1 0E-10 X)
      IF(ABS((XN-X)/XDEN) LT EPS) GO TO 120
      X=XN
      IF(K LT 1) GO TO 100
      IF(J LT K) GO TO 100
      J=0
      WRITE(6 25) I X
  100 CONTINUE
      WRITE(6 110) NT X XN
  110 FORMAT(//2X "NO CONVERGENCE IN" I5 3X "ITERATIONS"
     1 5X "X =" E14 4 5X "XN =" E14 4)
      STOP
  120 X=XN
      IF(K LT 1) RETURN
      WRITE(6,25) I X
      RETURN
      END
C
C
C
      SUBROUTINE WEGSTN(X NT,EPS FNC K)
C
C     A SUBROUTINE TO FIND THE SOLUTION OF A NONLINEAR FUNCTION
C     X=F(X) USING THE METHOU OF WEGSTEIN
C
C     X        INITIAL GUESS OF THE SOLUTION ON INPUT  BUT THE
C              BEST ESTIMATE OF THE SOLUTION ON OUTPUT
C     NT       MAXIMUM NUMBER OF ITERATIONS ALLOWED
C     EPS      RELATIVE ERROR AS THE STOPPING CRITERION
C     FNC      THE NAME OF THE FUNCTION THAT EVALUATES F(X)
C              USER MUST PROVIDE FUNCTION FNC(X) AND PUT THE
C              ACTUAL NAME OF THE FUNCTION IN EXTERNAL STATEMENT
C     K        A USER SPECIFIED PARAMETER TO CONTROL THE PRINTING OF
C              ITERATION RESULTS  EVERY K TH ITERATION IS PRINTED
C              NO PRINTING FOR K=0
C
C
      IF(K LT 1) GO TO 50
      WRITE(6,20)
   20 FORMAT(//5X "ITERATION RESULTS FOR WEGSTEIN'S METHOD  "
     1 5X "ITER" 8X "X")
      WRITE(6,25) 0 X
   25 FORMAT(2X,I5,E14 4)
C
   50 J=0
      X1=X
      F1=FNC(X1)
      X2=F1
      IF(K LT 1) GO TO 60
      WRITE(6,25) 1,X2
   60 DO 100 I=2,NT
      J=J+1
      F2=FNC(X2)
      X2M1=X2-X1
      IF(ABS(X2M1) LT 1 0E-10) X2M1=SIGN(1 0E 10,X2M1)
      S=(F2-F1)/X2M1
      IF(ABS(S-1 ) LT 0 000001) S=S+SIGN(0 00001 (S-1 ))
      T=1 /(1 -S)
      IF(ABS(T) GT 10 ) T=SIGN(10 ,T)
      XN=(1 -T)*X2+T*F2
      X1=X2
      F1=F2
      X2=XN
      XDEN=X1
      IF(ABS(X1) LT 1 0E-10) XDEN=SIGN(1 0E-10 X1)
      IF(ABS((X2-X1)/XDEN) LT EPS) GO TO 120
      IF(K LT 1) GO TO 100
      IF(J LT K) GO TO 100
      J=0
      WRITE(6,25) I X2
  100 CONTINUE
      WRITE(6 110) NT X1,X2
  110 FORMAT(//2X "NO CONVERGENCE IN" I5,3X "ITERATIONS" /
     1 5X "X1 =" E14 4,5X "X2 =" E14 4)
      STOP
  120 X=XN
      IF(K LT 1) RETURN
      WRITE(6,25) I X2
      RETURN
      END
C
C
C
      SUBROUTINE WEGSMD(N X NT EPS SUB K)
C
C     A SUBRDUTINE TO FIND THE SDLUTION OF N NONLINEAR EQUATIONS
C     OF THE FORM X1=F1(X1     XN),       XN=FN(X1,      XN)
C     USING THE MULTIDIMENSIONAL METHOD OF WEGSTEIN
C
C     N        NUMBER OF EQUATIONS TO BE SOLVED (LIMITED TO 40)
C     X        N-VECTOR OF INITIAL GUESSES ON INPUT  WHICH ARE SPECIFIED
C              BY THE USER  ON OUTPUT  X CARRIES THE BEST ESTIMATE OF
C              THE SOLUTION VECTOR
C     NT       MAXIMUM NUMBER OF ITERATIONS ALLOWED
C     EPS      RELATIVE ERROR SPECIFIED BY USER AS STOPPING CRITERION
C     SUB      A SUBROUTINE USED TO EVALUATE FUNCTION VALUES
C              SUBROUTINE MUST BE IN THE FORM SUB(M X F) WHERE BOTH
C              X AND F ARE M-OIMENSIONAL VECTORS  WHILE X=INDEPENDENT
C              VARIABLES AND F=FUNCTION VALUES
C              IN "SUB" VECTORS X ANF F MUST HAVE VARIABLE OIMENSIONS (M)
C              "SUB" MUST BE PROVIOED BY THE USER AND THE ACTUAL NAME
C              MUST BE OECLARED EXTERNAL
C     K        A USER SPECIFIED PARAMETER TO CONTROL THE PRINTING
C              OF ITERATION RESULTS  EVERY K TH ITERATION IS
C              PRINTED  NO PRINTING FOR K=0
C
C
C              COMMON /MWEG/ IS USED WHEN WE WANT 'NSEQ' SUCCESSIVE
C              SUBSTITUTION ITERATIONS  BETWEEN EACH WEGSTAIN ITERATION
C
      OIMENSION X(N) X2(60) F2(60) X1(60) F1(60)
      COMMON /MWEG/ NSEQ
C
      IF(NSEQ LT 1 OR NSEQ GT 1E14) NSEQ=0
C
      KL=0
```

```
C
C
      IF(K LT 1) GO TO 50
      WRITE(6,20)
   20 FORMAT(//5X,"ITERATION RESULTS FOR MULTIDIMENSIONAL WEGSTEIN  ",/)
      WRITE(6,25) 0,(X(I),I=1,N)
   25 FORMAT(2X,I5,5E13 4/7X,5E13 4)
C
   50 J=0
      DO 51 I=1,N
   51 X1(I)=X(I)
      CALL SUB(N,X1,F1)
      DO 52 I=1,N
   52 X2(I)=F1(I)
      IF(K LT 1) GO TO 60
      WRITE(6,25) 1,(X2(I),I=1,N)
C
      KL=0
C
   60 DO 100 IK=2 NT
      J=J+1
C
      CALL SUB(N,X2,F2)
C
      KL=KL+1
      IF(KL GT NSEQ) THEN
      KL=0
      DO 80 I=1,N
      XD=X2(I)-X1(I)
      IF(ABS(XD) GT 1 0E-08) GO TO 65
      X1(I)=X2(I)
      GO TO 80
   65 S=(F2(I)-F1(I))/XD
      IF (ABS(S-1 ) LT 0 00001) S=S+SIGN(0 00001 (S 1 ))
      T=1 /(1 -S)
      IF(ABS(T) GT 10 ) T=SIGN(10 ,T)
      X1(I)=X2(I)
      X2(I)=(1 -T)*X2(I)+ T*F2(I)
   80 F1(I)=F2(I)
      ELSE
      DO 55 I=1,N
      X1(I)=X2(I)
      X2(I)=F2(I)
   55 F1(I)=F2(I)
      ENDIF
C
      DO 85 I=1,N
      XDEN=X1(I)
      IF(ABS(XDEN) LT 1 0E-10) XDEN=1 0E-10
      IF(ABS((X2(I)-X1(I))/XDEN) GT EPS) GO TO 90
   85 CONTINUE
      CALL SUB(N,X2,F2)
      DO 180 I=1,N
      IF(ABS(X2(I)-F2(I)) GT EPS*ABS(F2(I))) GO TO 190
  180 CONTINUE
      GO TO 86
  190 DO 195 I=1,N
  195 X2(I)=F2(I)
      GO TO 90
   86 DO 88 I=1,N
   88 X(I)=X2(I)
      GO TO 120
   90 CONTINUE
      IF(K LT 1) GO TO 100
      IF(J LT K) GO TO 100
      J=0
      WRITE(6,25) IK,(X2(I),I=1,N)
  100 CONTINUE
      LL=MIN(N,10)
      WRITE(6,110) NT,(X2(I),I=1,LL)
  110 FORMAT(//2X,"MULTIDIMENSIONAL WEGSTEIN'S METHOD DOES NOT CONVERGE"
     1       /2X,"TOTAL ITERATIONS =",I3 5X,"LAST POINT  "
     2       /5E14 4/5E14 4)
C
      IF(LL LE 10) STOP
      WRITE(6,26) (X2(I) I=LL+1,N)
   26 FORMAT(/5E14 4,/5E14 4,//)
C
      STOP
  120 IF(K LT 1) RETURN
      WRITE(6,25) IK,(X(I) I=1,N)
      RETURN
      END
C
```

```
C
C
      SUBROUTINE READ(NST,NEQ)
C
C     A SUBROUTINE TO READ THE "NST" STREAM VARIABLES  AND THE "NEQ"
C     EQUIPMENT PARAMETERS  THE VARIABLES ARE DEFINED AS FOLLOWS
C
C     NST    NUMBER OF STREAMS  A MAXIMUM OF 30 IS ALLOWED  INFORMATION
C            FOR ALL STREAMS MUST BE ENTERED IN A SEQUENTIAL ORDER AS
C            DESCRIBED BELOW  STREAM NUMBERS MUST BE ENTERED FOR ALL
C            BUT THE FLOWRATES AND THE COMPOSITIONS (MOLE FRACTIONS)
C            CAN BE LEFT BLANK
C            A TOTAL OF 20 COMPONENTS ARE ALLOWED IN THE SYSTEM
C
C     NEQ :  NUMBER OF UNITS (EQUIPMENT)  INFORMATION MUST BE ENTERED
C            IN A SEQUENTIAL ORDER AFTER STREAM VARIABLES ARE SPECIFIED
C            ACCORDING TO THE FORMAT GIVEN BELOW
C            A MAXIMUM OF 20 UNITS ARE ALLOWED IN THE SYSTEM
C
C     FIRST LINE OF DATA FILE IS USED FOR PROBLEM IDENTIFICATION
C     AS A TITLE FOR THE OUTPUT  THE SECOND LINE IS NOT USED  BUT
C     IT CAN BE UTILIZED BY PUTTING INTEGER NUMBERS TO GUIDE THE
C     ENTERING OF DATA  THIRD LINE IS USED TO ENTER COMPONENT
C     IDENTIFICATIONS  VARIABLES ENTERED IN THIS LINE ARE  NC  CNAME(1)
C      , CNAME(NC) WHERE NC=NUMBER OF COMPONENTS, AND CNAME(J)=NAME OF
C     J TH COMPONENT  THE FORMAT USED IS I2,8X,10(A5,2X)  THUS  COMPONENT
C     NAMES ARE LIMITED TO 5 ALPHANUMERIC CHARACTERS  SIMILAR TO THE ABOVE
C     COLUMN IDENTIFICATION LINE, THE FIRST LINE AFTER STREAM VARIABLES
C     (OR THE FIRST LINE BEFORE EQUIPMENT PARAMETERS) IS USED
C     TO GUIDE DATA BY ENTERING INTEGER NUMBERS
C
C
C     STREAM VARIABLES ARE ENTERED IN THE FOLLOWING ORDER
C
C     LINE 1    NSTR(1), FLOW(1), COMP(1,1), COMP(1,2),
C                   , COMP(1,18), COMP(1,19)
C     LINE 2    NSTR(2), FLOW(2), COMP(2,1), COMP(2,2)
C                   , COMP(2,18), COMP(2,19)
C     ETC
C     WHERE
C
C     NSTR(I)       STREAM NUMBER  NSTR(1)=1  NSTR(2)=2  NSTR(3)=3  ETC
C                   THIS HELPS TO ENTER DATA IN A SEQUENTIAL ORDER
C                   AND TO IDENTIFY STREAM INFORMATION DURING CALCULATIONS
C     FLOW(I)       MOLAR FLOWRATE OF STREAM I  THIS PACKAGE OF PROGRAMS
C                   DOES NOT CHECK UNITS  THEREFORE, UNITS MUST BE KEPT
C                   CONSISTENT IN EACH PROBLEM
C     COMP(I,J)     MOLE FRACTION OF COMPONENT J IN STREAM I
C                   AT INPUT, ONLY THE KNOWN STREAM COMPOSITIONS
C                   (ALL TWENTY) ARE SPECIFIED  AT OUTPUT  ALL THE
C                   UNSPECIFIED STREAM VARIABLES ARE CALCULATED
C
C     FORMATTING FOR EACH LINE IS   (I2,F8 2,10F7 4)
C
C     EQUIPMENT PARAMETERS
C
C     THESE ARE ENTERED AFTER THE STREAM VARIABLES IN A SEQUENTIAL
C     ORDER AS FOLLOWS  (ONE LINE IS SKIPPED FOR COLUMN IDENTIFICATION)
C
C     LINE 1    NEQP(1), EQNAME(1)
C     LINE 2  . EQP(1,1), EQP(1,2),       . EQP(1,10)
C     LINE 3  . EQP(1,11), EQP(1,12),       , EQP(1,20)
C     LINE 4    IEQP(1,1), IEQP(1,2)        . IEQP(1,8)
C     LINE 5    NEQP(2), EQNAME(2)
C     LINE 6  . EQP(2,1), EQP(2,2),       . EQP(2,10)
C     LINE 7    EQP(2,11)  EQP(2,12).       . EQP(2,20)
C     LINE 8    IEQP(2,1)  IEQP(2,2).       , IEQP(2,8)
C     ETC
C     WHERE
C
C     NEQP(I)       EQUIPMENT NUMBER  NEQP(1)=1  NEQP(2)=2  ETC
C                   ONLY TEN (10) ARE ALLOWED
C     EQNAME(I)     NAME OF THE I TH EQUIPMENT  MODULES AVAILABLE ARE
C                   FLASH, SEPARATOR  REACTOR, SPLITTER, MIXER
C     EQP(I,J)      J TH PARAMETER OF THE I TH EQUIPMENT
C                   NOTE THAT EQP(I,J) IS REAL
C     IEQP(I,J)     J TH PARAMETER (INTEGER) OF THE ITH EQUIPMENT
C
C     SPECIFICATION OF PARAMETERS FOR EACH MODULE
C
C     FLASH          EQP(I J)=KJ WHERE KJ=X2J/X3J IS THE VAPOR-LIQUID
C                   MOLE FRACTION RATIO FOR THE OUTPUT STREAMS OF A
C                   FLASH EVAPORATOR  UP TO 19 KJ'S CAN BE SPECIFIED
C                   DUE TO COMPONENT  LIMITATION
```

```
C                    IEQP(I J)=STREAM NUMBER OF FEED INTO THE FLASH UNIT
C                   IF J=1  BUT STREAM NUMBER FOR VAPOR IF J=2 AND
C                   STREAM NUMBER FOR LIQUID IF J=3   I INDICATES THE
C                   EQUIPMENT NUMBER
C
C     SEPARATOR      EQP(I J)= BETAJ FOR EQUIPMENT I  WHERE
C                   BETAJ=X2J*N2/(X1J*N1) IS THE SEPARATION FACTOR FOR
C                   THE J TH COMPONENT
C                    IEQP(I J)=STREAM NUMBERS (J=1 FOR N1  J=2 FOR N2 AND
C                   J=3 FOR N3) FOR THE I TH EQUIPMENT
C
C     REACTOR        EQP(I,J)=NEUJ FOR J=1,2,    ,19  HERE  NEUJ IS THE
C                   STOICHIOMETRIC COEFFICIENT OF THE J TH COMPONENT IN
C                   REACTION  NEUJ IS POSITIVE FOR PRODUCTS  NEGATIVE
C                   FOR REACTANTS AND ZERO FOR INERTS
C                   EQP(I,20)= GAMMAK WHERE GAMMAK IS THE SPECIFIED
C                   CONVERSION FOR THE K TH COMPONENT WHICH MUST BE
C                   PRESENT IN THE REACTANT STREAM, GAMMAK IS IN (0 1)
C                    IEQP(I,J)=NUMER OF THE REACTANT (J=1) OR THE
C                   PRODUCT (J=2) STREAM  J=3 TO J=7 ARE NOT USED
C                   IEQP(I 8)=K TO INDICATE THE COMPONENT NUMBER FOR
C                   THE SPECIFIED CONVERSION
C
C     SPLITTER       EQP(I J)=DELTAJ WHERE DELTAJ IS THE SPLIT RATIO
C                   FOR STREAM NUMBERS IEQP(I,J) FOR J=2 3,    6,7
C                   NOTE THAT A GIVEN STREAM CAN SPLIT UP TO SIX ONLY
C                   SO THE LINE WITH EQP(J,11) >> EQP(J,20) MUST BE EMPTY
C                   OR WITH ZEROS
C                    IEQP(I J)=STREAM NUMBERS  J=1 FOR INPUT STREAM
C                   J=2 FOR 1ST OUTPUT STREAM, J=3 FOR 2ND OUTPUT STREAM,
C                        J=7 FOR  6TH OUTPUT STREAM  AND J=8 FOR THE
C                   NUMBER OF OUTPUT (SPLITS) STREAMS
C
C     MIXER          EQP(I J)  NOT USED
C                    LINES FOR EQP(J,1) >> EQP(J 20) MUST BE EMPTY
C                    OR WITH ZEROS
C                    IEQP(I J)=STREAM NUMBERS, J=1 FOR THE 1ST INPUT STREAM
C                    J=2 FOR THE 2ND INPUT STREAM,      J=6 FOR THE
C                   6TH INPUT STREAM,J=7 FOR THE STREAM NUMBER OF THE
C                   OUTPUT AND J=8 FOR THE NUMBER OF STREAMS TO BE MIXED
C                   NOTE THAT UP TO SIX STREAMS ARE ALLOWED TO BE MIXED
C
C     THE FORMATTING FOR THE TWO LINE SEQUENCES IS AS FOLLOWS
C     (I2,1X,A10,10G8 3,/,10G8 3,/,8I8)
C
C
C     ALL OF THE STREAM VARIABLES AND THE EQUIPMENT PARAMETERS ARE
C     STORED IN COMMON BLOCKS AND ARE AVAILABLE TO ALL OF THE
C     MODULES  TAPE5 IS USED TO READ DATA AND TAPE6 IS
C     USED TO WRITE THE RESULTS
C     THE COMMON BLOCKS USED ARE
C         COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20) COMP(30,20)
C         COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20) IEQP(20,8)
C
C
      DIMENSION FNAME(8)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30 20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20 20),IEQP(20 8)
C
      LL=0
      NS=NST
      NE=NEQ
      READ(5,2) (FNAME(I),I=1,8),ID
    2 FORMAT(8A10/I2)
      WRITE(6,3) (FNAME(I),I=1,8)
    3 FORMAT(1H1//3X,8A10)
   50 LL=LL+1
      IC=10*(LL-1)+1
      IE=10*LL
      READ(5,30) NY,(CNAME(J) J=IC IE)
   30 FORMAT(I2,8X,10(A5,2X))
      DO 5 I=1,NS
    5 READ(5,10) NSTR(I),FLOW(I),(COMP(I,J),J=IC IE)
   10 FORMAT(I2,F8 2,10F7 4)
      READ(5,15) IO
   15 FORMAT(I2)
      IF(NY GT 10 AND LL EQ 1) GO TO 50
      DO 20 I=1,NE
   20 READ(5,25) NEQP(I) EQNAME(I),(EQP(I J) J=1 20) (IEQP(I J) J=1 8)
   25 FORMAT(I2,1X,A10,/,10G8 3,/,10G8 3,/,8I8)
      NC=NY
      RETURN
      END
C
```

```
C
      SUBROUTINE FLASH(N)
C
C     A SUBROUTINE TO CALCULATE THE MATERIAL BALANCES
C     AROUND A FLASH EVAPORATOR WITH EQUIPMENT UNIT NUMBER =N
C     STREAM VARIABLES OF THE FEED AND THE EQUILIBRIUM
C     CONSTANTS MUST ALL BE SPECIFIED OR CALCULATED BEFORE
C     CALLING THIS SUBROUTINE
C
C     EK        19-VECTOR OF EQUILIBRIUM RELATIONS AT THE
C               SPECIFIC TEMPERATURE AND PRESSURE (SPECIFIED)
C     X1        MOLE FRACTIONS IN FEED (SPECIFIED)
C     X2        MOLE FRACTIONS IN VAPOR OUTPUT (CALCULATED)
C     X3        MOLE FRACTIONS IN LIQUID OUTPUT (CALCULATED)
C     FEED      MOLAR FEED FLOWRATE (SPECIFIED)
C     VAPOR     MOLAR VAPOR FLOWRATE (CALCULATED)
C     FLIQ      MOLAR LIQUID FLOWRATE (CALCULATED)
C
      DIMENSION EK(20),X1(20),X2(20),X3(20)
      COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20 8)
C
      NT=200
      A0=1 0
      EPS=1.0E-05
      N1=IEQP(N,1)
      FN1=FLOW(N1)
      FEED=FN1
      CALL CHECKS(N1)
      DO 10 J=1,NC
      EK(J)=EQP(N,J)
   10 X1(J)=COMP(N1,J)
      N2=IEQP(N,2)
      N3=IEQP(N,3)
      A=A0
      DO 100 K=1,NT
      F=0
      FD=0
      DO 50 I=1,NC
      IF(X1(I) LT 1 0E-10) GO TO 50
      B=1 -1 /EK(I)
      F=F+X1(I)/(1 -A*B)
      FD=FD+X1(I)*B/((1 -A*B)**2)
   50 CONTINUE
      F=F-1
      AN=A-F/FD
      IF(ABS(AN-A) LT EPS) GO TO 200
      A=AN
  100 CONTINUE
      WRITE(6,110) NT,A,F,FD
  110 FORMAT(//2X,"NEWTON'S METHOD IN FLASH CALCULATIONS DOES NOT",/
     1         2X,"CONVERGE IN",I5,"   ITERATIONS",/,5X "A =",E10 3
     2         5X,"F =",E10 3,5X "FD =",E10 3/)
      PRINT*, 'PARAMETERS AT THE TIME OF THE ERROR '
      CALL WRITES
      CALL WRITEE
      STOP
  200 A=AN
      IF(A GT 2 *EPS AND A LT 1 0) GO TO 205
      WRITE(6,203) N A
  203 FORMAT(//2X,"WRONG ROOT IN FLASH" I3," CALCULATIONS"/
     1    2X,"ALFA =",E12 3,3X,"UNFEASIBLE SOLUTION FOR THIS INPUT SET")
      PRINT*, 'PARAMETERS AT THE TIME OF THE ERROR '
      CALL WRITES
      CALL WRITEE
      STOP
  205 DO 210 I=1,NC
      X3(I)=X1(I)/(A+(1 -A)*EK(I))
      X2(I)=EK(I)*X3(I)
      COMP(N2,I)=X2(I)
  210 COMP(N3,I)=X3(I)
      FLIQ=A*FEED
      VAPOR=FEED-FLIQ
      FLOW(N2)=VAPOR
      FLOW(N3)=FLIQ
C
      RETURN
      END
C
C
C
      SUBROUTINE SEPAR(N)
C
C     A SUBROUTINE TO CALCULATE THE MATERIAL BALANCES AROUND A
```

```
C         SEPARATOR WITH EQUIPMENT NUMBER =N  STREAM VARIABLES FOR THE
C         INPUT STREAM AND THE SEPARATION FACTORS (BETAJ'S) MUST ALL
C         BE SPECIFIED OR CALCULATED FOR UNIT N BEFORE CALLING THIS
C         SUBROUTINE
C
          COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
          COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
          REAL F(4,10)
C
          N1=IEQP(N,1)
          CALL CHECKS(N1)
          IF(IEQP(N,8) GT 2) GO TO 100
          S=0
          DO 30 J=1,NC
       30 S=S+(1 0-EQP(N,J))*COMP(N1,J)
          FN1=FLOW(N1)
          FN3=FN1*S
          FN2=FN1-FN3
          N2=IEQP(N,2)
          N3=IEQP(N,3)
          FLOW(N2)=FN2
          FLOW(N3)=FN3
          DO 40 J=1,NC
          COMP(N2,J)=EQP(N,J)*COMP(N1,J)*FN1/FN2
       40 COMP(N3,J)=(1 -EQP(N,J))*COMP(N1,J)*FN1/FN3
          RETURN
      100 CONTINUE
C
          DO 200 I=1,NC
          F(1,I)=FLOW(N1)*COMP(N1,I)
      200 F(4,I)=F(1,I)
C
          DO 205 I=2,3
          K=0
          IF(I EQ 3) K=10
          DO 205 L=1,NC
          F(I,L)=F(1,L)*EQP(N,K+L)
      205 F(4,L)=F(4,L)-F(I,L)
C
          DO 210 I=2,4
          K=IEQP(N,I)
          FLOW(K)=0
          DO 210 L=1,NC
      210 FLOW(K)=FLOW(K)+F(I,L)
C
          DO 215 I=2,4
          K=IEQP(N,I)
          DO 215 L=1,NC
      215 COMP(K,L)=F(I,L)/FLOW(K)
C
          RETURN
          END
C
C
C
          SUBROUTINE REAC(N)
C
C         A SUBROUTINE TO CALCULATE THE MATERIAL BALANCES AROUND A
C         REACTOR WITH EQUIPMENT UNIT NUMBER =N
C         STREAM VARIABLES FOR THE INPUT STREAM, THE STOICHIOMETRIC
C         COEFFICIENTS AND THE CONVERSION FRACTION MUST ALL BE
C         SPECIFIED OR CALCULATED BEFORE CALLING THIS SUBROUTINE
C
          COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
          COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
C
          N1=IEQP(N,1)
          CALL CHECKS(N1)
          GAMMAK=EQP(N,20)
          K=IEQP(N,8)
          SK=EQP(N,K)
          FN1=FLOW(N1)
          R=-GAMMAK*COMP(N1,K)*FN1/SK
          IF(R GT 0 ) GO TO 24
          WRITE(6,46) N,R,SK,GAMMAK
       46 FORMAT(//2X,"CONVERSION SPECIFICATIONS AND/OR INPUT PARAMETERS"/
         1 2X,"ARE WRONG FOR REACTOR " I2,/2X,"REACTION RATE =" E14 4 /
         2 2X,"STOICHIMETRIC COEFFICIENT FOR CONVERSION =" F7 2/
         3 2X,"CONVERSION =",F8 3)
          PRINT*, 'PARAMETERS AT THE TIME OF THE ERROR '
          CALL WRITES
          CALL WRITEE
          STOP
       24 S=0
```

```
          DO 10 J=1,NC
       10 S=S+EQP(N,J)
          FN2=FN1+S*R
          N2=IEQP(N,2)
          FLOW(N2)=FN2
          DO 20 J=1,NC
       20 COMP(N2,J)=(COMP(N1,J)*FN1+EQP(N,J)*R)/FN2
          RETURN
          END
C
C
C
          SUBROUTINE SPLIT(N)
C
C         A SUBROUTINE TO CALCULATE THE MATERIAL BALANCES AROUND A
C         SPLITTER WITH EQUIPMENT UNIT NUMBER =N  INPUT STREAM VARIABLES
C         AND THE SPLITTING FRACTIONS MUST ALL BE SPECIFIED OR CALCULATED
C         BEFORE CALLING THIS SUBROUTINE
C
          COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
          COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
C
          N1=IEQP(N,1)
          CALL CHECKS(N1)
          FN1=FLOW(N1)
          M=IEQP(N,8)
          IF(M GT 1 AND M LT 19) GO TO 5
          WRITE(6,7) N,M
        7 FORMAT(///5X,"IEQP(N,8) IN ERROR FOR N=",I3/
         1 5X,"VALUE SPECIFIED IS =",I4)
          PRINT*, 'PARAMETERS AT THE TIME OF THE ERROR '
          CALL WRITES
          CALL WRITEE
          STOP
        5 M1=M+1
          S=0
          DO 20 J=2,M1
          NJ=IEQP(N,J)
          FLOW(NJ)=EQP(N,J)*FN1
          S=S+FLOW(NJ)
          DO 10 J1=1,NC
       10 COMP(NJ,J1)=COMP(N1,J1)
       20 CONTINUE
          IF(ABS(S-FN1) LT 1 0E-08) GO TO 30
          WRITE(6,25) (EQP(N,J),J=2,M1),S,N
       25 FORMAT(///2X,"SPLIT FRACTIONS DO NOT ADD UP TO 1 "/
         1       2X,7F10 4/5X,"FOR EQUIPMENT =",I4)
          PRINT*, 'PARAMETERS AT THE TIME OF THE ERROR '
          CALL WRITES
          CALL WRITEE
          STOP
       30 RETURN
          END
C
C
C
          SUBROUTINE MIXER(N)
C
C         A SUBROUTINE TO CALCULATE THE MATERIAL BALANCES AROUND A
C         MIXER WITH EQUIPMENT UNIT NUMBER =N  INPUT STREAM VARIABLES
C         MUST ALL BE SPECIFIED OR CALCULATED BEFORE CALLING THIS SUBROUTINE
C
          COMMON /S1/ NS,NSTR(30),FLOW(30),NC,CNAME(20),COMP(30,20)
          COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
C
          M=IEQP(N,8)
          IF(M GT 1 AND M LT 7) GO TO 5
          WRITE(6,7) N,M
        7 FORMAT(///5X,"IEQP(N,8) IN ERROR FOR N=",I3/
         1 5X,"VALUE SPECIFIED IS =",I4)
          STOP
        5 DO 10 I=1,M
          N1=IEQP(N,I)
       10 CALL CHECKS(N1)
          S1=0
          DO 20 I=1,M
          FN=FLOW(IEQP(N,I))
       20 S1=S1+FN
          FLOW(IEQP(N,7))=S1
          DO 30 J=1,NC
          S2=0
          DO 25 K=1,M
          NK=IEQP(N,K)
          FK=FLOW(NK)
```

```
   25 S2=S2+COMP(NK J)*FK
   30 COMP(IEQP(N,7),J)-S2/S1
      RETURN
      END
C
C
C
      SUBROUTINE CHECKS(N1)
C
C     A SUBROUTINE TO CHECK THE CONSISTENCY OF STREAM VARIABLES
C     OF STREAM N1  THE SUM OF MOLE FRACTIONS ARE COMPARED TO 1
C
      COMMON /S1/ NS NSTR(30),FLOW(30),NC,CNAME(20) COMP(30 20)
      COMMON /E1/ NE NEQP(20),EQNAME(20),EQP(20 20),IEQP(20 8)
C
      LL=0
      EPS=1 0E-02
      FN1=FLOW(N1)
      S=0
      DO 10 J=1,NC
      S=S+COMP(N1,J)
   10 CONTINUE
      IF(ABS(S-1 ) LT EPS AND FN1 GT EPS) GO TO 30
      WRITE(6,12)
   12 FORMAT(///2X," STREAM VARIABLES ARE UNSPECIFIED OR INCONSISTENT")
   50 LL=LL+1
      IC=10*(LL-1)+1
      IE=LL*10
      WRITE(6,15) (CNAME(J),J=IC,IE)
   15 FORMAT(//1X,"STREAM VARIABLES  "/1X,"NSTR",2X "FLOW" 10(2X A5))
      WRITE(6,20) N1,FN1,(COMP(N1 J),J=IC,IE)
   20 FORMAT(2X,I2 F8 2,10F7 4)
      IF(NC GT 10 AND LL EQ 1) GO TO 50
      STOP
   30 RETURN
      END
C
C
C
      SUBROUTINE WRITEE
C
C     A SUBROUTINE TO WRITE THE EQUIPMENT PARAMETERS
C
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20) IEQP(20 8)
C
      WRITE(6,10)
   10 FORMAT(//2X,"EQUIPMENT PARAMETERS ",4X,
     1        " (2 X EQP(I,J))/IEQP(I,J)")
      DO 20 I=1,NE
   20 WRITE(6,30) NEQP(I),EQNAME(I),(EQP(I J) J=1 20) (IEQP(I J) J=1 8)
   30 FORMAT(/,14X,"----------     MODULE # (",I2 ") - ",A10,
     1        " ----------",/,10F8 3,/,10F8 3,/ 8(I5 3X))
      WRITE(6,33)
   33 FORMAT(/)
      RETURN
      END
C
C
C
      SUBROUTINE WRITES
C
C     A SUBROUTINE TO WRITE THE STREAM VARIABLES
C
      COMMON /S1/ NS NSTR(30),FLOW(30),NC,CNAME(20) COMP(30 20)
C
      LL=0
   50 LL=LL+1
      IC=(LL-1)*10+1
      IE=LL*10
      IF(NC LE IE) IE=NC
      WRITE(6,10) (CNAME(J),J=IC IE)
   10 FORMAT(//1X,"STREAM VARIABLES  "/1X,"NSTR" 2X "FLOW" 10(2X A5))
      DO 20 I=1,NS
   20 WRITE(6,30) NSTR(I),FLOW(I),(COMP(I,J) J=IC IE)
   30 FORMAT(I2,F8 2 10F7 4)
      IF(NC GT 10 AND LL EQ 1) GO TO 50
      RETURN
      END
C
C
C
C
      SUBROUTINE WRITEX(L1,L2)
C
C     A SUBROUTINE TO WRITE THE EQUIPMENT PARAMETERS
C
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20,8)
C
      WRITE(6,10)
   10 FORMAT(//2X,"EQUIPMENT PARAMETERS ",4X,
     1        " (2 X EQP(I,J))/IEQP(I,J)")
      DO 20 I=L1,L2
   20 WRITE(6,30) NEQP(I),EQNAME(I),(EQP(I,J),J=1,20),(IEQP(I J) J=1,8)
   30 FORMAT(/,14X,"----------     MODULE # (",I2,") = ",A10,
     1        " ----------",/,1X,10E10 3,/,1X,10E10 3,/ 8(I7 3X))
      WRITE(6,33)
   33 FORMAT(/)
      RETURN
      END
C
C
C
C
C
      SUBROUTINE SIMSO(I1,I2,I3,IP,NSIG)
C
C      A SUBROUTINE TO SOLVE MASS BALANCES USING THE SIMULTANEOUS
C      MODULAR APPROACH  IF THE PROBLEM HAS CONSTRAINS  SUBROUTINE
C      SPEC MUST BE USED WITH SUBROUTINE SIMSO
C
C      I1        TOTAL NUMBER OF STREAMS TORN
C
C      I2(I)     NUMBER OF EACH STREAM TORN (I=1,I1)
C
C      I3        TOTAL NUMBER OF CONSTRAINS
C
C      IP        PRINTING PARAMETER  FOR IP=1 NO PRINTING OF STREAM
C                OR EQUIPMENT PARAMETERS BEFORE SOLUTION  FOR IP=2
C                ONLY STREAM VARIABLES ARE PRINTED  FOR IP=3
C                EQIPMENT  AND STREAM PARAMETERS ARE PRINTED
C
C      NSIG      DESIREO ACCURACY OF SOLUTION
C
      EXTERNAL FCN
      COMMON /S1/ NS NSTR(30),FLOW(30),NC,CNAME(20),COMP(30 20)
      COMMON /E1/ NE,NEQP(20),EQNAME(20),EQP(20,20),IEQP(20 8)
      COMMON /TAA1/ N1,N2(20),N3
      COMMON /TBB2/ N4(20),N5(20),N6(20),N7(20),N8(20) N9(20) F1(20)
      COMMON /ITT/ LIT,INT
      DIMENSION X(50),F(50),I2(I1),WK(1400)
      INT=2
      MS=2
      ITMAX=300
      N1=I1
      N3=I3
      N=NC*I1+N3
      DO 1 I=1,N1
      N2(I)=I2(I)
      DO 1 L=1,NC
    1 X((I-1)*NC+L)=FLOW(N2(I))*COMP(N2(I),L)
      IF(N3 LE 0) GO TO 3
      K=0
      DO 2 I=1+NC*I1,N
      K=K+1
      X(I)=FLOW(N4(K))
      IF(N5(K) EQ 0) GO TO 2
      X(I)=EQP(N4(K),N5(K))
    2 CONTINUE
    3 CONTINUE
C
      CALL THREE(X,F,N,IP)
C
      CALL MPOLM(FCN X,F,N,WK,ITMAX,NSIG,MS)
C
      WRITE(6,200)
      WRITE(6,300) ITMAX,INT
      WRITE(6,200)
  200 FORMAT(1X,/,"--------------------------------------"
     1        "---------------------------------------------"/)
  300 FORMAT(/,/,10X,"NUMBER OF SIMULTANEOUS ITERATIONS =" I4 /,
     1        10X,"NUMBER OF SEQUENTIAL ITERATIONS   =" I4 /)
      RETURN
      END
C
C
C
C
      SUBROUTINE SPEC(N3 NAME,NUE,NUP NACO,NUS NCO VAL)
```

```
C
C
C        A SUBROUTINE TO SPECIFY ALL THE CONSTRAINS IMPOSED TO THE
C        PROBLEM
C
C        N3           TOTAL NUMBER OF CONSTRAINS (MAX OF 19)
C
C        I=1,N3
C
C        NAME(I)      NAME OF THE MODULES WHICH HAVE EQUIPMENT PARAMETERS
C                     MANIPULATED  IF A FLOW RATE IS MANIPULATED
C                     SET NAME(I)='FLOW'
C
C        NUE(I)       NUMBER OF  MODULE OR FLOW RATE
C
C        NUP(I)       NUMBER OF THE EQUIPMENT PARAMETER MANIPULETED  FOR
C                     EQP(L,J), NUP(I)=J , IF A FLOW RATE IS MANIPULATED
C                     SET NUP(I)=0 (THIS IS VERY IMPORTANT)
C
C
C        NACO(I)      NAME OF THE CONSTRAINS BEING IMPOSED  (THERE ARE ONLY
C                     TWO POSSIBILITIES 'FLOW' FOR FLOW RATE OR 'COMP' FOR
C                     COMPOSITION
C
C        NUS(I)       NUMBER OF THE STREAM WHICH HAS THE CONSTRAIN 'NACO(I)'
C
C        NCO(I)'      NUMBER OF THE COMPONENT BEING SPECIFIED  IF THE SPECIFI-
C                     CATION IS FLOW RATE ,SET NCO(I)=1
C
C        VAL(I)       NUMERICAL VALUE OF THE CONSTRAIN
C
      INTEGER NUE(N3),NUP(N3),NUS(N3),NCO(N3)
      REAL VAL(N3)
      CHARACTER*4 NAME(N3),NACO(N3)
      CHARACTER*10 NPR(20)
      COMMON /TBB2/ N4(20),N5(20),N6(20),N7(20),N8(20),N9(20),F1(20)
C
      DO 1 I=1,N3
      N4(I)=NUE(I)
      N8(I)=NUS(I)
      F1(I)=VAL(I)
C
      IF(NAME(I) EQ 'SPLI') THEN
      N6(I)=1
      N5(I)=NUP(I)
      NPR(I)='SPLITTIER'
      GO TO 1
      ELSE
C
      IF(NAME(I) EQ 'REAC') THEN
      N6(I)=3
      N5(I)=NUP(I)
      NPR(I)='REACTOR'
      GO TO 1
      ELSE
C
      IF(NAME(I) EQ 'SEPA') THEN
      N6(I)=2
      N5(I)=NUP(I)
      NPR(I)='SEPARATOR'
      GO TO 1
      ELSE
      IF(NAME(I) EQ 'FLOW') THEN
      N6(I)=4
      N5(I)=NUP(I)
      NPR(I)='FLOW RATE'
      GO TO 1
      ELSE
      IF(NAME(I) EQ 'USE1') THEN
      N6(I)=1
      N5(I)=NUP(I)
      NPR(I)='USER 1'
      GO TO 1
      ELSE
C
      IF(NAME(I) EQ 'USE2') THEN
      N6(I)=2
      N5(I)=NUP(I)
      NPR(I)='USER 2'
      GO TO 1
      ELSE
C
      WRITE(6,100)NAME(I),I
      ENDIF
      ENDIF
      ENDIF
      ENDIF
      ENDIF
      ENDIF
    1 CONTINUE
C
  100 FORMAT(/,10X,"CHECK SPELLING FOR ",A4,"(",I2,")",/)
C
      DO 2 I=1,N3
C
      IF(NACO(I) EQ 'FLOW') THEN
      N7(I)=1
      N9(I)=1000
      GO TO 2
      ELSE
C
      IF(NACO(I) EQ 'COMP') THEN
      N7(I)=2
      N9(I)=NCO(I)
      GO TO 2
      ELSE
C
      IF(NACO(I) EQ 'USER') THEN
      N7(I)=3
      N9(I)=1000
      NACO(I)='NONE'
      GO TO 2
      ELSE
C
      WRITE(6,100)NACO(I),I
C
      ENDIF
      ENDIF
      ENDIF
    2 CONTINUE
C
      WRITE(6,200)
  200 FORMAT(1X,"-------------------------------------"
     1            "--------------------------------------------------",/)
      WRITE(6,50)
      DO 3 I=1,N3
      IF(N9(I) EQ 1000) GO TO 4
      WRITE(6,10) NPR(I),NUE(I),NACO(I),N8(I),N9(I),VAL(I)
      GO TO 3
C
    4 IF(N7(I) EQ 3) THEN
      WRITE(6,30) NPR(I),NACO(I)
   30 FORMAT(8X,A10,2X,"###",6X,A4)
      GO TO 3
      ELSE
      WRITE(6,20) NPR(I),NUE(I),NACO(I),N8(I),VAL(I)
      ENDIF
    3 CONTINUE
   10 FORMAT(/8X,A10,2X,I3,6X,A4,"(",I2,",",I2,")= ",F10.4)
   20 FORMAT(/8X,A10,2X,I3,6X,A4,"(",I2,")    = ",F10.4)
   50 FORMAT(/2X,"MODULE MANIPULATED  #        CONSTRAIN ")
      RETURN
      END
C
C
C
C
C
      SUBROUTINE FCN(X,F,N)
C
C       SUBROUTINE FCN CONTAINS THE SYSTEM OF NONLINEAR
C       EQUATIONS BEING SOLVED  THIS SUBROUTINE IS USED
C       BY SUBROUTINE MPDLM
C
      DIMENSION X(N),F(N)
      COMMON /TAA1/ N1,N2(20),N3
      COMMON /TBB2/ N4(20),N5(20),N6(20),N7(20),N8(20),N9(20),F1(20)
      COMMON /ITT/ LIT,INT
      INT=INT+1
      IT=1
C
C
      IF(N3 LE 0) GO TO 10
      CALL TEAR1(X,F,IT,N)
      CALL CONT1(X,F,IT,N)
C
      DO 20 I=1,N
   20 IF(N7(I) EQ 3) IT=IT-1
```

```
C
      CALL FSIS(X F IT N)
      IT=1
      CALL TEARO(X.F.IT.N)
      CALL CONTO(X.F.IT.N)
      LIT=IT
      RETURN
   10 CONTINUE
      IT=1
      CALL TEARI(X F.IT.N)
      LIT=IT
      CALL FSIS(X.F.IT.N)
      IT=1
      CALL TEARO(X.F.IT.N)
      RETURN
      END
C
C       SUBROUTINE TEARI AND CONTI ARE USED TO ASSIGN THE VALUES
C       CALCULATED BY MPOLM TO THE PROCESS VARIABLES
C
C       SUBROUTINE TEARO AND CONTO ARE USED TO CALCULATE
C       FUNCTION VALUES
C
C
C
C
C
C
        SUBROUTINE TEARI(X.F.IT.N)
        DIMENSION X(N).F(N)
      COMMON /TAA1/ N1.N2(20).N3
      COMMON /S1/ NS.NSTR(30).FLOW(30) NC.CNAME(20) COMP(30 20)
C
      JJ=IT
      LL=IT
      DO 1 I=1.N1
      FLOW(N2(I))=0
      DO 2 L=1.NC
      FLOW(N2(I))=FLOW(N2(I))+(X(LL))
    2 LL=LL+1
      DO 3 K=1.NC
      COMP(N2(I).K)=(X(JJ))/FLOW(N2(I))
    3 JJ=JJ+1
    1 IT=IT+NC
      RETURN
      END
C
C
C
C
      SUBROUTINE TEARO(X.F.IT.N)
      DIMENSION X(N).F(N).FM(50)
      COMMON /S1/ NS.NSTR(30).FLOW(30) NC.CNAME(20) COMP(30 20)
      COMMON /TAA1/ N1.N2(20).N3
C
      DO 1 I=1.N1
      J=N2(I)
      DO 2 L=1.NC
    2 FM(L)=FLOW(J)*COMP(J.L)
      DO 3 L=1.NC
      F(IT)=X(IT)-FM(L)
      IT=IT+1
    3 CONTINUE
    1 CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE CONTI(X F.IT N)
      COMMON /E1/ NE NEQP(20) EQNAME(20).EQP(20.20) IEQP(20 8)
      COMMON /S1/ NS.NSTR(30) FLOW(30).NC.CNAME(20).COMP(30 20)
      COMMON /TAA1/ N1.N2(20).N3
      COMMON /TBB2/ N4(20).N5(20).N6(20).N7(20) N8(20) N9(20) F1(20)
      DIMENSION X(N) F(N)
C
      DO 1 I=1.N3
      SP=0
C
      IF(N6(I) EQ 1) GO TO 10
      IF(N6(I) EQ 4) GO TO 5
      EQP(N4(I).N5(I))=X(IT)
      IT=IT+1
      GO TO 1
    5 FLOW(N4(I))=X(IT)
      IT=IT+1
      GO TO 1
C
   10 DO 2 J=2.IEQP(N4(I).8)
      EQP(N4(I).J)=X(IT)
      SP=SP+X(IT)
    2 IT=IT+1
      EQP(N4(I).IEQP(N4(I).8)+1)=1 -SP
    1 CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE CONTO(X.F.IT.N)
      COMMON /S1/ NS.NSTR(30).FLOW(30).NC CNAME(20).COMP(30 20)
      COMMON /TAA1/ N1.N2(20).N3
      COMMON /TBB2/ N4(20).N5(20).N6(20) N7(20).N8(20) N9(20) F1(20)
      DIMENSION X(N).F(N)
C
      DO 1 I=1.N3
      GO TO(10.20.1).N7(I)
C
   10 F(IT)=FLOW(N8(I))-F1(I)
      IT=IT+1
      GO TO 1
C
   20 F(IT)=COMP(N8(I) N9(I))-F1(I)
      IT=IT+1
    1 CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE THREE(X F.N.IP)
C
      COMMON /S1/ NS.NSTR(30).FLOW(30) NC.CNAME(20).COMP(30 20)
      DIMENSION X(N).F(N)
      COMMON /TAA1/ N1.N2(20).N3
      COMMON /ITT/ LIT.INT
C
      CALL FCN(X.F.N)
      IT=LIT
      DO 10 I=1.2
   10 CALL FSIS(X.F.IT.N)
C
      IT=1
      DO 20 I=1.N1
      L=N2(I)
      DO 30 K=1.NC
      X(IT)=FLOW(L)*COMP(L.K)
   30 IT=IT+1
C
   20 CONTINUE
      GO TO (70.50.60).IP
   50 CALL WRITES
      GO TO 80
   60 CALL WRITES
      CALL WRITEE
   80 WRITE(6.200)
      WRITE(6.100)
      READ*.NNN
  200 FORMAT(1X./."----------------------------------------"
     1          "--------------------------------------------------"./)
      IF(NNN EQ 10) STOP
  100 FORMAT(/.5X."RESULTS AFTER 3 SEQUENTIAL ITERATIONS "
     */.5X."IF YOU WANT TO STOP THE PROGRAM AND CHANGE  "
     */.5X."INITIAL GUESSES ENTER 10 AS ? APPEARS ",/)
   70 RETURN
      END
C
```

MPDLM
VERSION 1

```
C
C
C-----------------------------------------------------------------------C
C
C
      SUBROUTINE MPDLM(FCN,X,F,N,B,ITMAX,IDGT,MS)
      REAL X(N),F(N),B(N,N),DF(150),PK(150),PS(150)
      REAL G(150),FN(150),CG(9)
      REAL Q(150),DIA(150),XB(6000)
      REAL LAMB
      INTEGER IP(150)
C
      COMMON /SETA/ IRI
      IF(IRI.LT.0.OR.IRI.GT.1000) THEN
      IRI=50
      OVAL=1000
      ELSE
      OVAL=10
      ENDIF
C
      IF(MS.NE.1) THEN
      DO 1 I=1,N
    1 DF(I)=1
      ELSE
      ENDIF
C
      DGT=1
      DO 2 I=1,IDGT
    2 DGT=DGT/10
C
C             HJ IS THE MACHINE EPSLON
C
      HJ=6.E-8
      PA1=HJ*SQRT(128.)
      PA2=1./128
C
      IM=MAX(10,(N+4))
      ICON=0
C
      DO 3 I=1,9
    3 CG(I)=1
      IDEL=0
C
C             EVLUATION OF THE JACOBIAN
C
  100 CONTINUE
      CALL FCN(X,F,N)
C
      DO 6 I=1,N
      IF(ABS(F(I)).LT.1E-14) F(I)=0.0
    6 CONTINUE
C
      DO 10 I=1,N
      HMU=ABS(X(I))
      PA3=MAX(HMU,PA2)
      PA4=PA1*PA3
      TEMP=X(I)
      XD=X(I)+PA4
      PA4=XD-TEMP
      X(I)=XD
      CALL FCN(X,FN,N)
      X(I)=TEMP
      DO 11 L=1,N
      IF(ABS(FN(L)).LT.1E-14) FN(L)=0
   11 B(L,I)=(FN(L)-F(L))/PA4
   10 CONTINUE
C
      II=0
      IFLAG=0
      JFLAG=1
      TAU=1
C
C
C             CALCULATE INITIAL STEP BOUND
C
      IF(ICON.EQ.0) THEN
      PB1=EUCN(X,N)
      PB1=PB1*IRI/100
C
      PB2=MIN(PB1,OVAL)
      IF(N.GT.10) PB2=PB1
      DELTA=PB2
      ELSE
      DELTA=SEARCH
      ENDIF
C
      IF(MS.EQ.2) THEN
      DO 15 I=1,N
      HOLD=0.0
      DO 16 L=1,N
   16 IF(ABS(B(I,L)).GT.HOLD) HOLD=ABS(B(I,L))
      DF(I)=1./HOLD
   15 CONTINUE
      ELSE
      ENDIF
C
C
C             CALCULATE SEARCH STEP
C
      DO 20 I=1,N
      F(I)=F(I)*DF(I)
      DO 20 L=1,N
   20 B(I,L)=B(I,L)*DF(I)
      PNJ=EUCN(F,N)
C
      WRITE(9,900)((B(I,L),L=1,N),I=1,N)
      WRITE(9,900)(DIA(I),I=1,N)
      WRITE(9,900)(X(I),I=1,N)
      WRITE(9,900)(F(I),I=1,N)
      WRITE(9,900)(DF(I),I=1,N)
      WRITE(9,900)SK,SKI,DELTA
      WRITE(9,901)
      WRITE(8,901)
  901 FORMAT(10X,"NEW JACOBIAN")
C
C             DECOMPOSE JACOBIAN INTO L*U FACTORS
C
      CALL FACLU(B,IP,N)
C
      DO 450 I=1,N
      DIA(I)=B(I,I)
  450 B(I,I)=1
C
  150 CONTINUE
      DELHOL=DELTA
C
C             SOLVE LINEAR SYSTEM
C
      G(1)=-F(IP(1))
      DO 400 I=2,N
      FAC=0
      DO 401 J=1,I-1
  401  FAC=FAC+B(I,J)*G(J)
  400 G(I)=-F(IP(I))-FAC
      PK(N)=G(N)/DIA(N)
      DO 405 I=N-1,1,-1
      FAC=0
      DO 406 J=I+1,N
  406 FAC=FAC+B(I,J)*PK(J)
  405 PK(I)=(G(I)-FAC)/DIA(I)
C               CALCULATE CORRECTION STEP (PK)
C
  160 PCI=EUCN(PK,N)
C
      IRV=0
      IF(DELTA.GE.PCI) THEN
      IRV=1
      GO TO 29
      ELSE
      ENDIF
C
C       OBTAIN --- G
C
```

```
      DO 410 I=1 N
  410 Q(I)=-F(IP(I))
      DO 411 I=1 N-1
      HOLD=0
      DO 412 L=I+1 N
  412 HOLD=HOLD+B(L I)*Q(L)
  411 Q(I)=Q(I)+HOLD
      DO 415 I=1 N
  415 G(I)=DIA(I)*Q(I)
      DO 416 I=1 N-1
      DO 416 L=I+1 N
  416 G(L)=G(L)+B(I L)*Q(I)
C
      PC2=EUCN(G N)
      PC3=PC2*PC2
C
      DO 420 I=1 N
  420 Q(I)=DIA(I)*G(I)
      DO 421 I=1 N-1
      DO 421 L=I+1 N
  421 Q(I)=Q(I)+B(I L)*G(L)
      DO 422 I=1 N
  422 PS(I)=Q(I)
      DO 423 I=2 N
      DO 423 L=1 I-1
  423 PS(I)=PS(I)+B(I L)*Q(L)
C
      PC4=EUCN(PS N)
      PC5=PC4*PC4
      PC6=PC3/PC5
      DO 24 I=1 N
   24 PS(I)=G(I)*PC6
      PD1=EUCN(PS N)
      IF(PD1 LT DELTA) GO TO 28
      PD2=DELTA/PC2
      DO 25 I=1 N
   25 PK(I)=PD2*G(I)
      GO TO 29
   28 CONTINUE
C
C           EVALUATION OF ALFA
C
      PE1=PC1*PC1
      PE2=PD1*PD1
      PE3=0
      PE4=0
      DO 26 I=1 N
      PE3=PE3+(PK(I)-PS(I))*PS(I)
   26 PE4=PE4+PK(I)*PS(I)
C
      PE5=DELTA*DELTA
      PE6=(PE1-PE5)*(PE5-PE2)+(PE4-PE5)*(PE4-PE5)
      PE7=SQRT(PE6)
      ALFA=(PE5-PE2)/(PE3+PE7)
      PF1=(1 -ALFA)
      DO 27 I=1 N
   27 PK(I)=ALFA*PK(I)+PF1*PS(I)
   29 CONTINUE
      PC1=EUCN(PK N)
C
C           EVALUATION OF F(XK+PK)
C
      DO 30 I=1 N
      X(I)=X(I)+PK(I)
   30 CONTINUE
      WRITE(7 900)(X(I) I=1 N)
      SCE=EUCN(X N)
      DGTF=SQRT(DGT)
      HOLD=MAX(1  SCE)
      DGTC=DGT*HOLD
C
      CALL FCN(X FN N)
      PG1=EUCN(FN N)
C
      IF(PG1 LT DGTF AND PC1 LT DGTC) THEN
      ITMAX=ICON
      RETURN
      ELSE
      ENDIF
C
      DO 31 I=1 N
   31 FN(I)=FN(I)*DF(I)
C
C
```

```
      PG2=EUCN(FN N)
      PG3=EUCN(F N)
      SK1=PG2*PG2
      SK=PG3*PG3
      IF(SK1 LT (SK*0 999)) THEN
      II=II-1
      IF(II LT 0) II=0
      ELSE
      II=II+1
      ENDIF
      IF(IM LT II) THEN
      PRINT*,'  CONVERGENCE IS TOO SLOW '
      PRINT*,'  CHANGE INITIAL GUESSES '
      SSS=SSS/PP
      STOP
      ELSE
      ENDIF
C
C           CHECK WHETHER A NEW EALUATION
C           OF THE JACOBIAN IS NEEDED
C
      IFLAG=0
      DO 40 I=1 8
      HOLD=CG(I+1)
   40 CG(I)=HOLD
      CG(9)=SK
C
      IF(ICON LT 10) THEN
      IF(II GT 3) IFLAG=1
      ELSE
      R1=SK1/CG(5)
      R2=SQRT(CG(5)/CG(1))
      IF(R1 GT R2 OR II GT 3) IFLAG=1
      ENDIF
C
C            IF IFLAG IS = TO 1 **AND**
C           F(XK) AS BEEN REDUCED BY A
C           FACTOR OF TWO A NEW JACOBIAN
C           IS EVALUATED (RENEWED)
C
      PG21=PG2
      P1=ABS(PG2-PNJ)
      IF(ABS(P1/PNJ) LT HJ) PG21=PNJ
      IF(IFLAG EQ 1 AND (PNJ/PG21) GT 2) GO TO 100
C
      IF(SK1 LT SK) GO TO 47
      DO 41 I=1 N
   41 X(I)=X(I)-PK(I)
   47 CONTINUE
C
C
C        UPDATE DELTA
C
      IF(IRV EQ 1) THEN
      DO 428 I=1 N
  428 G(I)=-F(I)
      ELSE
      DO 430 I=1 N 1
      Q(I)=DIA(I)*PK(I)
      DO 430 L=I+1 N
  430 Q(I)=Q(I)+B(I L)*PK(L)
      Q(N)=DIA(N)*PK(N)
      DO 435 I=2 N
      HOLD=Q(I)
      DO 436 L=1 I-1
  436 HOLD=HOLD+B(I L)*Q(I)
  435 G(IP(I))=HOLD
      G(IP(1))=Q(1)
      ENDIF
C
      IF(JFLAG EQ 1 AND SK1 GE SK) THEN
CC
      BL=0
      DO 42 I=1 N
   42 BL=BL+F(I)*G(I)
      BL=2 *BL
      CL=SK
      AL=SK1-BL-CL
      LAMB=-BL/(2 *AL)
      PH3=MAX(0 1 LAMB)
      DELTA=PC1*PH3
C
      IF(SK1 LT SK AND IDEL EQ 0) THEN
      SEARCH=DELTA
      IDEL=1
      ELSE
      ENDIF
C
      IF(DELTA LT DELHOL) TAU=1
      IF(PG2 GT (1 5*PG3)) GO TO 160
      GO TO 200
      ELSE
C
       IF(IRV EQ 1) THEN
       PH4=0
       ELSE
      DO 44 I=1,N
   44 Q(I)=F(I)+G(I)
      PH4=EUCN(Q,N)
      PH4=PH4*PH4
      ENDIF
C
      DM=SK-SK1-0 1*(SK-PH4)
      IF(DM LT 0) THEN
      DELTA=PC1/2
      TAU=1
      ELSE
      PTP=0
      PTS=0
C
      IF(IRV EQ 1) THEN
      PTP=PG2*PG2
      PTS=PTP
      ELSE
      DO 46 I=1,N
      HOLD=FN(I)-Q(I)
      PTP=PTP+ABS(FN(I)*HOLD)
   46 PTS=PTS+HOLD*HOLD
      ENDIF
C
      PJ1=PTP+SQRT(PTP*PTP+DM*PTS)
      LAMB=SQRT(1 +DM/PJ1)
      AMU=MIN(2  LAMB TAU)
      TAU=LAMB/AMU
      DELTA=AMU*PC1
      IF(DELTA LT DELHOL) TAU=1
      ENDIF
      ENDIF
C
  200 CONTINUE
      JFLAG=0
      IF(ICON GT ITMAX) GO TO 300
      ICON=ICON+1
C
C
      WRITE(8,900)((B(I,L) L=1,N),I=1,N)
      WRITE(8,900)(DIA(I),I=1,N)
      WRITE(8,900)(X(I),I=1,N)
      WRITE(8,900)(F(I),I=1,N)
      WRITE(8,900)(DF(I),I=1 N)
      WRITE(8,900)SK,SK1,DELTA
  900 FORMAT(10(1X,G12 6))
C
      IF(SK1 LT SK AND IDEL EQ 0) THEN
      SEARCH=DELTA
      IDEL=1
      ELSE
      ENDIF
C
C            THE JACOBIAN WILL BE UPDATE BY
C            BROYDEN'S ALGORITHM
C
      DX1=0
      DO 50 I=1,N
   50 DXI=DXI+PK(I)*PK(I)
C
      WRITE(9,900)(PK(I) I=1,N)
      IF(IRV EQ 1) THEN
      DO 51 I=1,N
   51 Q(I)=FN(I)
      ELSE
      DO 52 I=1,N
   52 Q(I)=FN(I)-F(I) G(I)
      END IF
      WRITE(9,900)(Q(I) I=1,N)
C
      DO 53 I=1,N 1
      PK(I)=PK(I)/DXI
      G(I)=Q(IP(I))
      DO 53 L=I+1 N
   53 B(I,L)=B(I L)/DIA(I)
C
      G(N)=Q(IP(N))
      PK(N)=PK(N)/DX1
      WRITE(9,900)(PK(I),I=1,N)
      WRITE(9,900)(F(I),I=1,N)
      WRITE(9,900)(FN(I),I=1,N)
      I=1
      XB(1)=1
      DO 55 L=2,(N*N/4+1)
   55 XB(L)=0
C
   80 CONTINUE
C
      PTS=0
      K=1
      KI=N-I+1
      DO 60 M=1,I
      PTS=PTS+XB(K)*G(M)
   60 K=K+KI
C
      DIA(I)=DIA(I)+PK(I)*PTS
      IF(I EQ N) GO TO 85
      OO 62 M=1,N
   62 Q(M)=PK(I)*G(M)
C
      M=1
      DO 70 L=1,I
      F(L)=XB(M)
   70 M=M+KI
      M=1
      DO 63 L=1,I
      DO 63 J=1,N-I
      XB(M)=XB(M+L)-B(I+J,I)*F(L)
   63 M=M+1
C
C
      XB(M)=1
      DO 71 L=M+1 M+KI
   71 XB(L)=0
      HOLD=PK(1)
      OO 64 J=1,N-I
   64 PK(J)=PK(J+1)-B(I,I+J)*HOLD
C
      KI=KI-1
      DO 65 J=1,N-I
      B(I,I+J)=B(I,I+J)+PK(J)*PTS/DIA(I)
      HOLD=0
      K=J
      DO 66 M=1,I
      HOLD=HOLD+XB(K)*Q(M)
   66 K=K+KI
      HOLD=HOLD+Q(I+J)
   65 B(I+J,I)=B(I+J,I)+HOLD/DIA(I)
      HOLD=PTS/DIA(I)
      DO 67 M=1,N
   67 G(M)=G(M)-HOLD*Q(M)
      I=I+1
      GO TO 80
   85 CONTINUE
      WRITE(9,900)((B(I,L),L=1,N),I=1,N)
      WRITE(9,900)(DIA(I) I=1,N)
C
      DO 86 I=1,N-1
      F(I)=FN(I)
      DO 86 L=I+1,N
   86 B(I,L)=B(I,L)*DIA(I)
      F(N)=FN(N)
C
      GO TO 150
C
  300 CONTINUE
      PRINT*,'  NO CONVERGENCE IN ' ITMAX ' ITERATIONS'
      PRINT*,'  CHANGE INITIAL GUESSES OR USE  ANOTHER SUBROUTINE
      STOP
      END
C
C
C
C
C
C           FUNCTION EUCN EVALUATES THE EUCLIDIAN NORM
```

```
C             OF A VECTOR OF DIMENSION " N "
C
C
      FUNCTION EUCN(Y,J)
      REAL Y(J)
      SS=0
      DO 1 I=1,J
      SS=SS+Y(I)*Y(I)
    1 CONTINUE
      EUCN=SQRT(SS)
      RETURN
      END
C
C
C
C
      SUBROUTINE FACLU(A,IP,N)
C
C        THIS SUBROUTINE FACTORS MATRIX A INTO A PRODUCT
C        OF A LOWER TRIANGULAR MATRIX L AND AN UPPER TRI-
C        ANGULAR U. L HAS UNIT DIAGONAL WHICH IS NOT STORED
C
C        FROM    IBM PROGRAMMER'S MANUAL
C                SYSTEM/360 SCIENTIFIC SUBROUTINE PACKAGE
C                (360A-CM-03X) VERSION III  (1968)
C
C        USAGE
C                A IS THE MATRIX WHICH ONE WANTS TO FACTORISE
C                THE MATRIX IS STORED COLUMNWISE
C                IP CONTAINS A PERMUTATION VECTOR ON OUTPUT
C                PER IS USED FOR INTERNAL COMPUTATIONS
C
      REAL A(N*N),PER(150)
      INTEGER IP(N)
C
C             COMPUTATION OF WEIGHTS FOR EQUILIBRATION
C
      DO 20 I=1,N
      IP(I)=I
      IJ=I
      X=ABS(A(IJ))
      DO 10 J=2,N
      IJ=IJ+N
   10 IF(ABS(A(IJ)).GE.X) X=ABS(A(IJ))
      IF(X.LE.0) THEN
      PRINT*,'  A ROW IN THE INPUT MATRIX IS NULL'
      ELSE
      ENDIF
   20 PER(I)=1./X
      IO=0
      DO 100 I=1,N
      IM1=I-1
      IP1=I+1
      IPIVOT=I
      X=0
C
C             COMPUTATION OF THE ITH COLUMN OF L
C
      DO 50 K=I,N
      KI=IO+K
      DP=A(KI)
      IF(I-1) 110,40,25
   25 KJ=K
      DO 30 J=1,IM1
      IJ=IO+J
      DP=DP-A(KJ)*A(IJ)
   30 KJ=KJ+N
      A(KI)=DP
C
C             SEARCH FOR EQUILIBRATED PIVOT
C
   40 IF((X-ABS(DP)*PER(K)).GE.0.) GO TO 50
      IPIVOT=K
      X=ABS(DP)*PER(K)
   50 CONTINUE
      IF(X.LE.0) GO TO 110
C
C             PERMUTATION OF OF ROWS IF REQUIRED
C
   55 IF(IPIVOT-I) 110,70,57
   57 KI=IPIVOT
      IJ=I
C
      IAUX=IP(IJ)
      IP(IJ)=IP(KI)
      IP(KI)=IAUX
C
      DO 60 J=1,N
      X=A(IJ)
      A(IJ)=A(KI)
      A(KI)=X
      KI=KI+N
   60 IJ=IJ+N
      PER(IPIVOT)=PER(I)
   70 PER(I)=IPIVOT
      IF((I-N).GE.0) GO TO 100
      IJ=IO+I
      X=A(IJ)
C
C
C
      KO=IO+N
      DO 90 K=IP1,N
      KI=IO+K
      A(KI)=A(KI)/X
      IF(I-1) 110,90,75
   75 IJ=I
      KI=KO+I
      DP=A(KI)
      DO 80 J=1,IM1
      KJ=KO+J
      DP=DP-A(IJ)*A(KJ)
   80 IJ=IJ+N
      A(KI)=DP
   90 KO=KO+N
  100 IO=IO+N
      RETURN
  110 WRITE(6,120)
  120 FORMAT(//,5X," FACLU CAN NOT SOLVE THIS PROBLEM",
     1         /,5X," CHECK INPUT PARAMETERS OR USE ANOTHER"
     2         /,5X," SUBROUTINE ",//)
      SS=SS/Q
      RETURN
      END
```