MEMORANDUM
RM-4345-PR
JANUARY 1965



# THE NUMERICAL SOLUTION OF THE CHEMICAL EQUILIBRIUM PROBLEM

R. J. Clasen

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-4345-PR
JANUARY 1965

# THE NUMERICAL SOLUTION OF THE CHEMICAL EQUILIBRIUM PROBLEM

R. J. Clasen

This research is sponsored by the United States Air Force under Project RAND—Contract No. AF 19(638)-700 monitored by the Directorate of Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.



The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA • 90406

## PREFACE

This Memorandum is one in a continuing series of RAND publications dealing with theoretical computational questions arising from the RAND program of research in biology and physiology. The Memorandum contributes to our ability to apply computer technology to the analysis of complex chemical systems by considering the "chemical equilibrium problem," the problem of determining the distribution of chemical species that minimizes the free energy of a system while conserving the mass of each of the chemical elements.

Solutions to the chemical equilibrium problem published up to this time [4,5] apply to those problems for which an estimate of the solution exists. This Memorandum considers a problem for which no estimated solution exists and solves that problem with the maximum precision now available.

The mathematical aspects of this Memorandum should also be of interest in other fields where computational analyses of complex chemical systems are under consideration, e.g., in studies of rocket propulsion systems, planetary atmospheres, re-entry problems, etc.

## SUMMARY

In physical chemistry, the "chemical equilibrium problem" is the problem of determining the distribution of chemical species that minimizes the free energy of a system while conserving the mass of each of the chemical elements. The reactions occurring within the chemical system may be quite complex. However, in a great number of cases, the mathematical statement of the problem can be simplified to a particular mathematical form [7,8] involving the minimization of a nonlinear objective function over a set of linear constraints.

This Memorandum presents the numerical solution of the chemical equilibrium problem by describing methods for starting the solution when an initial estimate is not available, and for improving an initial estimate to make it feasible. It presents a first-order method and a second-order method for solving the chemical equilibrium problem in the context of the linear-logarithmic programming problem [4] and provides convergence criteria for the majority of problems of this type that are likely to be attempted.

## FOREWORD

In deciding between the languages of mathematics and physical chemistry, we have chosen in this Memorandum to use that of mathematics. The disadvantage of this choice is that the physical chemist may experience some difficulty in immediately identifying certain concepts. The advantage is that mathematical language divorces the methods from the physical assumptions involved in constructing a mathematical model of a physical system.* The mathematical methods are, hence, free to transcend their specific chemical applications.

The methods given here do not solve every problem that is specified in the given mathematical form. The solution of a problem in which some phase vanishes (a degenerate problem) requires further work. Some work has been done on particular degenerate systems [13], but the accurate numerical solution of a large general system of this type has yet to be accomplished. Until recently, a skilled physical chemist could intuitively eliminate the degeneracies of his model and

---

*The reader is referred to other works for the procedure of constructing the mathematical models of biochemical systems [9-12].

obviate the need for solving a degenerate system. But, as problems grow, eliminating degeneracy becomes increasingly difficult. Frequently, the point at which the problem becomes too large for the physical chemist to decide whether or not to include a phase coincides with the point at which the problem becomes numerically unwieldy. Hopefully, the future will eliminate these difficulties.

Statements about convergence and convergence tests exist, unless otherwise indicated, in the context of finite-accuracy numerics. Statements of this kind do not mean, in the absence of qualification, that no problem exists nor that no machine would serve as a counter example. Rather they are simply descriptions of what was found to occur in actual practice.

No attempt has been made to describe those methods which were tried and found wanting. The methods presented are those which are best for the largest number of cases.

Finally, it should be pointed out that although computing time was a factor, it was considered secondary to accuracy of results.

## ACKNOWLEDGMENTS

The author wishes to thank J. C. DeHaven, E. C. DeLand, F. R. Gilmore, and N. Z. Shapiro for their many constructive comments and suggestions.

CONTENTS

PREFACE ..... iii

SUMMARY ..... v

FOREWORD ..... vii

ACKNOWLEDGMENTS ..... ix

Section
1. INTRODUCTION ..... 1

2. THE INITIAL SOLUTION ..... 5
The Projection Method ..... 5
The Linear Programming Method ..... 9

3. THE LINEAR-LOGARITHMIC PROGRAMMING PROBLEM, FIRST-ORDER METHOD ..... 15

4. THE FIRST-ORDER METHOD FOR SOLVING THE CHEMICAL EQUILIBRIUM PROBLEM ..... 21

5. THE LINEAR-LOGARITHMIC PROGRAMMING PROBLEM, SECOND-ORDER METHOD ..... 28

6. THE SECOND-ORDER CHEMICAL EQUILIBRIUM ALGORITHM ..... 33

7. SUMMARY OF THE COMPUTATION PROCEDURE ..... 35

Appendix
A. A FORTRAN-IV PROGRAM FOR SOLVING THE CHEMICAL EQUILIBRIUM PROBLEM ..... 37
General Description ..... 37
Subroutines ..... 41
Listing ..... 45
Calling Sequence for Simplex Subroutine ..... 56

B. MATRIX NOTATION AND FURTHER PROOFS ..... 67

REFERENCES ..... 79

## 1. INTRODUCTION

For the purposes of this Memorandum, the chemical equilibrium problem is merely a name we use for a particular mathematical programming problem, i.e., the problem of minimizing a particular nonlinear function $F(x_1,x_2,\ldots,x_n)$, defined below, while satisfying the linear restraints or constraints

$$\sum_{j=1}^{n} a_{ij}\, x_j = b_i \qquad i=1,2,3,\ldots,m \qquad (1.1)$$

with $x_j \geq 0$ for $j=1,2,\ldots,n$ and $a_{ij}$, $b_i$ given constants. Assuming that the equations of (1.1) are linearly independent, then in order to have a non-trivial problem it can be assumed that $m<n$. The variables $x_1,x_2,\ldots,x_n$ can be considered components of a vector $(x_1,x_2,\ldots,x_n)$. Solving the chemical equilibrium problem then is the problem of determining this vector. The variable $x_j$ will be referred to as the "$j^{th}$ component"; also the numerical value of $x_j$ may be referred to as the "component" rather than using the perhaps linguistically correct but cumbersome term "component value."

The components are partitioned into p non-empty subsets called compartments. Let us denote these compartments by $\langle 1\rangle, \langle 2\rangle, \ldots, \langle p\rangle$. Then if the $j^{th}$ component is in the $k^{th}$ compartment, we will say $j \in \langle k\rangle$, where each component is in exactly one compartment. The number of the compartment that the $j^{th}$ component is in is denoted by $[j]$. Hence $j \in \langle k\rangle$ implies $[j] = k$, and conversely. Each compartment has associated with it a sum defined by

$$S_k = \sum_{j \in \langle k\rangle} x_j \ . \tag{1.2}$$

The component fraction $\hat{x}_j$ is defined by $\hat{x}_j = \frac{x_j}{S_{[j]}}$ whenever $S_{[j]} > 0$.

The objective function to be minimized over (1.1) is

$$F(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} x_j (c_j + \log \hat{x}_j) \tag{1.3}$$

where $c_1, c_2, \ldots, c_n$ are given constants, called objective constants.

When an $x_j$ is zero, $\log \hat{x}_j$ is undefined; but we define 0 log 0 to equal 0 so that we may evaluate F when

some components are zero. A feasible solution to the chemical equilibrium problem is defined to be any set of non-negative components that satisfies (1.1). The problem is said to be feasible if it has feasible solutions. If no feasible solution is arbitrarily large in any component, the feasible problem is said to be bounded feasible; all practical problems with which one might have occasion to deal are bounded feasible.

A solution or optimal solution to a bounded feasible problem is any feasible solution in which $F(x_1,\ldots,x_n)$ attains the minimum value over all feasible solutions. A problem which has optimal solutions in which some component is zero is called degenerate, and a bounded feasible problem in which the components in any optimal solution are all strictly positive is called a non-degenerate problem. It has been shown [1, Theorem 12.1] that a non-degenerate problem has exactly one optimal solution. Hence, we may speak of the solution to the problem. Furthermore, it has also been shown* for the non-degenerate problem that the minimization of F is equivalent to the existence of numbers $\pi_1,\pi_2,\ldots,\pi_m$, called Lagrange multipliers, which satisfy:

---

*Ref. 1, p. 18.

$$\sum_{i=1}^{m} \pi_i a_{ij} = c_j + \log \hat{x}_j \,. \qquad j=1,2,3,\ldots,n \qquad (1.4)$$

In the following sections we derive conditions, analogous to (1.4), which are useful in solving the problem. In Sec. 2 we are interested in finding a solution to (1.1) with all $x_j > 0$. A set of $x_j$ which satisfies these conditions is called a positive feasible solution. If (1.1) is satisfied with $x_j \geq 0$, we have called such a result a feasible solution. The theory of linear programming gives us methods of finding feasible solutions to problems with linear restraints. In Sec. 2, we use a linear programming technique to find a positive feasible solution. In Sec. 4 we show how to modify the initial positive feasible solution to get the solution to the problem.

## 2. THE INITIAL SOLUTION

The algorithms presented in the following sections require an initial positive feasible solution in order that the procedure for solving the problem can be initiated. Frequently, an individual with a problem to solve will be able to give a rather accurate estimate of its optimal solution. This estimate may be the exact solution of another problem which differs from the one being considered in relatively minor ways.

### THE PROJECTION METHOD

Let us suppose that such is the case, and let us denote the estimate of the components by $y_1, y_2, \ldots, y_n$. These values, substituting $y_j$ for $x_j$ in Eq. (1.1), will not generally satisfy (1.1), being somewhat in error. Let us denote these errors by $g_1, g_2, \ldots, g_m$; that is, let

$$g_i = b_i - \sum_{j=1}^{n} a_{ij} y_j . \qquad i=1,2,\ldots,m \qquad (2.1)$$

Then, we wish to find corrections to $y_j$ such that, denoting the corrections by $\theta_j$, we have

$$b_i - \sum_{j=1}^{n} a_{ij}(y_j + \theta_j) = 0 \quad i=1,2,\ldots,m$$

or

$$g_i = \sum_{j=1}^{n} a_{ij}\theta_j \quad . \qquad i=1,2,\ldots,m \qquad (2.2)$$

The $\theta_j$ must also be chosen such that $y_j + \theta_j > 0$, for all $j$. We cannot guarantee this condition, but we can attempt to choose small values for $\theta_j$. One way to do this is to minimize

$$\sum_{j=1}^{n} w_j \theta_j^2$$

subject to (2.2), where $w_j$ is the "weight" or relative importance of minimizing $\theta_j$. This reduces to the problem of finding Lagrange multipliers $\pi_1, \pi_2, \ldots, \pi_m$, such that with

$$L = \frac{1}{2} \sum_{j=1}^{n} w_j \theta_j^2 - \sum_{i=1}^{m} \pi_i \left( \sum_{j=1}^{n} a_{ij}\theta_j - g_i \right) \qquad (2.3)$$

we have

$$\frac{\partial L}{\partial \theta_j} = 0 \quad . \qquad j=1,2,\ldots,n \qquad (2.4)$$

Equation (2.4) becomes

$$w_j\theta_j = \sum_{i=1}^{m} a_{ij}\pi_i \qquad j=1,2,\ldots,n \qquad (2.5)$$

and substituting (2.5) into (2.2) we have

$$g_i = \sum_{\ell=1}^{m}\left[\pi_\ell\left(\sum_{j=1}^{n}\frac{a_{\ell j}a_{ij}}{w_j}\right)\right]. \quad i=1,2,\ldots,m \qquad (2.6)$$

The terms

$$\sum_{j=1}^{n}\frac{a_{\ell j}a_{ij}}{w_j}$$

can be immediately evaluated; let us denote these terms by

$$q_{\ell i} = \sum_{j=1}^{n}\frac{a_{\ell j}a_{ij}}{w_j} \quad . \qquad (2.7)$$

Note that $q_{\ell i} = q_{i\ell}$. Then, (2.6) becomes

$$g_i = \sum_{\ell=1}^{m} q_{\ell i}\pi_\ell \quad . \qquad i=1,2,\ldots,m \qquad (2.8)$$

Equation (2.8) is a set of m simultaneous equations in the m unknowns, $\pi_1, \pi_2, \ldots, \pi_m$. These equations may be solved for $\pi_1, \pi_2, \ldots, \pi_m$, and then these values may be substituted in (2.5) to get $\theta_1, \theta_2, \ldots, \theta_n$. There remains the question of choosing values for the weighting factors $w_j$. In tests of this method, it has been found that using

$$w_j = \frac{1}{y_j}$$

yields satisfactory results. The choice of the weighting factors depends, to some extent, on the available computers. Using these weighting factors, we can summarize the computation of $\theta_j$ in the following three equations:

$$q_{\ell i} = \sum_{j=1}^{n} a_{\ell j} a_{ij} y_j \qquad \begin{matrix} i=1,2,\ldots,m \\ \ell=1,2,\ldots,m \end{matrix} \qquad (2.9)$$

$$\sum_{\ell=1}^{m} q_{\ell i} \pi_\ell = b_i - \sum_{j=1}^{n} a_{ij} y_j \qquad i=1,2,\ldots,m \qquad (2.10)$$

$$\theta_j = y_j \sum_{i=1}^{m} a_{ij} \pi_i \qquad j=1,2,\ldots,n \qquad (2.11)$$

where

$$x_j = y_j + \theta_j \,. \qquad j=1,2,\ldots,n \qquad (2.12)$$

The $x_j$ from (2.12) will satisfy (1.1). However, the $x_j$ need not all be strictly positive. If any $x_j$ is zero or negative, this method of obtaining the initial solution, which we shall call the projection method, has failed. If the projection method fails, or if no initial estimate is provided, then a linear programming method may be used.

## THE LINEAR PROGRAMMING METHOD

The terminology used in linear programming is similar to the terminology used above in describing the chemical equilibrium problem. The statement of a linear programming problem includes a set of linear restraints

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad i=1,2,\ldots,m \qquad (2.13)$$

together with a set of constants $C_1,C_2,C_3,\ldots,C_n$, called costs. A feasible solution to a linear programming problem is any set of non-negative $x_j$ such that (2.13) is satisfied. The costs are used to form the following expression, L, which is called the objective function

$$L = \sum_{j=1}^{n} C_j x_j \quad . \tag{2.14}$$

For every set of feasible $x_j$, we can evaluate L. The set of feasible $x_j$ for which L has the minimum value that it can have with any set of feasible $x_j$, is called a solution of the linear programming problem. A problem which has sets of feasible $x_j$ is called a feasible problem, and a problem in which there are no sets of feasible $x_j$ is called an infeasible problem. An infeasible problem has no solutions, while a feasible problem has at least one solution. In this discussion, we will not be concerned as to whether a problem has more than one solution: we will only be concerned with finding a solution to the problem. Since the means of finding a solution to a linear programming problem has been the subject of many papers and books, we will not give an actual method of solving the linear programming problem here. The reader may refer to Dantzig [2] for a complete discussion of the problem.

The problem of finding a feasible solution to a linear programming problem is itself a linear programming problem--that is, it involves finding a solution to the

problem with all $C_j$ equal to zero. With all $C_j = 0$, L in (2.14) is zero for any set of feasible $x_j$; hence, L is at its minimum value for any set of feasible $x_j$. Since L is at its minimum value for any feasible set of $x_j$, any feasible set of $x_j$ is, by the above definition, a solution to the linear programming problem.

However, we must not only find a feasible solution to the linear programming problem, we must also find a positive feasible solution to the problem. In order to do this, we let

$$x_j = y_j + y_{n+1} . \qquad j=1,2,\ldots,n \qquad (2.15)$$

If we can find non-negative values of $y_1, y_2, \ldots, y_{n+1}$ which satisfy

$$\sum_{j=1}^{n} a_{ij}(y_j + y_{n+1}) = b_i \qquad i=1,2,\ldots,m \qquad (2.16)$$

then $x_j$, as defined by (2.15), will be a feasible solution. If we can somehow assure that $y_{n+1}$ is positive, then all $x_j$ will be positive. Rewriting (2.16), we have

$$\sum_{j=1}^{n} a_{ij}y_j + \left( \sum_{j=1}^{n} a_{ij} \right) y_{n+1} = b_i \quad . \qquad i=1,2,\ldots,m \qquad (2.17)$$

If we now specify $C_1, C_2, \ldots, C_{n+1}$, we have a linear programming problem in n+1 unknowns. In order to guarantee that $y_{n+1}$ is positive, if it is possible for it to be positive, we can <u>maximize</u> $y_{n+1}$. It is easy to see that we can maximize $y_{n+1}$ by setting

$$L = - y_{n+1} \qquad (2.18)$$

which is equivalent to setting $C_1=C_2=C_3=\ldots=C_n=0$, $C_{n+1} = -1$. If the solution to the resulting linear programming problem is feasible and $y_{n+1} > 0$, then we have, by (2.15), a positive feasible solution to the analogous chemical equilibrium problem (1.1). If the linear programming problem is feasible but $y_{n+1} = 0$, then the analogous chemical equilibrium problem is degenerate, since there is no strictly positive solution to the problem. However, this is a rather trivial kind of degeneracy, and its occurrence usually indicates that a mistake was made in setting up the problem. Hence, this linear programming method gives us a way of finding a positive feasible solution to the chemical equilibrium problem if the chemical equilibrium problem is non-degenerate.

The positive feasible solution that we obtain by this method will generally not resemble the final solution of the chemical equilibrium problem. The initial positive feasible solution can be improved by the following technique. Define $b_{m+1}$ to be some multiple, between zero and one, of the value of $y_{n+1}$ that was obtained above. Then, adjoin to the linear restraints (2.17) one more restraint of the form $y_{n+1} = b_{m+1}$. Next, solve the linear programming problem with these restraints and with $C_1=c_1$, $C_2=c_2$, ..., $C_n=c_n$, $C_{n+1}=0$ (recall that the lower-case c's here refer to the c's in the chemical equilibrium problem (1.3)). The solution to this linear programming problem will give a set of components more nearly resembling the solution to the chemical equilibrium problem than did the components calculated from Eqs. (2.17) and (2.18). This new solution, in turn, may be improved by solving another linear programming problem (the details of which can be seen in SUBROUTINE LP in Appendix A) and averaging the new solution with the old solution.

In order to solve an elaborate chemical equilibrium problem it is not sufficient to simply use a method which we can prove converges to the correct solution. Proofs of convergence generally assume infinite computational accuracy, but since we are usually limited in practice to

about eight significant digits, the numerical solution will not always converge. However, it has been observed that the closer we can get to the solution by the initial solution methods described above, the greater will be the probability that the numerical procedure will converge. Furthermore, not only will the probability of convergence be greater, but the number of iterations to get to the solution will be fewer, and hence--when an improved initial solution is used--the computation time will be shorter. Unfortunately, the mathematical methods that are available for analyzing convergence of iterative processes do not, in the case of the chemical equilibrium problem, enable us to prove convergence when we are limited to finite mathematical accuracy. Only experience with a particular method will tell us whether it is a useful numerical procedure to use.

In the next section we consider a somewhat more general problem than the chemical equilibrium problem. This problem is considered first because the numerical results take on an especially simple form when the additional generality is admitted.

## 3. THE LINEAR-LOGARITHMIC PROGRAMMING PROBLEM, FIRST-ORDER METHOD

In this section we consider the problem of minimizing

$$F(x_1,x_2,\ldots,x_N) = \sum_{j=1}^{N} x_j(c_j + d_j \log x_j) \qquad (3.1)$$

while satisfying the linear restraints

$$\sum_{j=1}^{N} a_{ij}x_j = b_i \,. \qquad i=1,2,3,\ldots,M \qquad (3.2)$$

The symbols $a_{ij}$, $b_i$, $c_j$, and $d_j$ denote constants, and $x_1,x_2,\ldots,x_N$ are the unknowns that we seek. We restrict the problem to the case that $d_j \neq 0$ for $j = 1,2,3,\ldots,N$. We note that if $x_j < 0$, the term in (3.1), $x_j(c_j + d_j \log x_j)$, is undefined, whereas if $x_j > 0$ this term is defined. If $x_j = 0$ we define $x_j(c_j + d_j \log x_j) = 0$, since this expression approaches zero as $x_j > 0$ approaches zero. From this discussion, we see that, in order for a solution of Eqs. (3.1) and (3.2) to be defined, we must assume that $x_j \geq 0$ for $j = 1,2,3,\ldots,N$.

We may attempt to solve this problem using Lagrange multipliers.* In this method we let

$$L = F(x_1,x_2,x_3,\ldots,x_N) - \sum_{i=1}^{M}\left[\pi_i\left(\sum_{j=1}^{N} a_{ij}x_j - b_i\right)\right]$$

and then set

$$\frac{\partial L}{\partial x_j} = 0$$

for $j = 1,2,3,\ldots,N$. Performing the partial differentiation, we get

$$c_j + d_j \log x_j + d_j - \sum_{i=1}^{M} \pi_i a_{ij} = 0 , \qquad (3.3)$$

$$j=1,2,3,\ldots,N$$

or, when rearranged,

$$\log x_j = d_j^{-1}\left[\sum_{i=1}^{M} \pi_i a_{ij} - c_j - d_j\right] . \qquad (3.4)$$

$$j=1,2,3,\ldots,N$$

---

*See Kaplan, Ref. 3, p. 128, or Dantzig, Ref. 2, p. 140.

Exponentiating both sides of (3.4), we get

$$x_j = \exp\left[ d_j^{-1} \sum_{i=1}^{M} \pi_i a_{ij} - d_j^{-1} c_j - 1 \right]_{j=1,2,3,\ldots,N} . \qquad (3.5)$$

Note that for (3.5) to be a solution to the problem, we must have all $x_j > 0$. We assume, in the remainder of this section, that the solution does have all $x_j > 0$. Then, the problem reduces to the problem of determining the M $\pi_i$ so that the $x_j$ from (3.5) satisfy (3.2) Equivalently, the M + N equations (3.2) and (3.5) must be satisfied simultaneously by the proper choice of the M + N unknowns, $\pi_1, \pi_2, \ldots, \pi_M$, $x_1, x_2, \ldots, x_N$. We now consider two methods of approximating the solution.

In the first method, we suppose that we have an estimate of the $x_j$ which may or may not satisfy (3.2). We denote this estimate by $y_j$, and, in this method, solve Eqs. (3.2) and (3.4) simultaneously by making a linear approximation to log $x_j$. Since we have the estimate that $x_j$ is near $y_j$, we note that the first-order Taylor expansion of log $x_j$ about $y_j$ is

$$\log x_j = \log y_j + \frac{x_j - y_j}{y_j} + \text{(higher-order terms)} \quad . \quad (3.6)$$

Dropping the higher-order terms, and substituting (3.6) into (3.4) and solving for $x_j$, we have

$$x_j = y_j \left[ d_j^{-1} \sum_{i=1}^{M} \pi_i a_{ij} - d_j^{-1} c_j - \log y_j \right] . \quad (3.7)$$

$$j=1,2,3,\ldots,N$$

Now, if we substitute these $x_j$ into (3.2), we get

$$\sum_{\ell=1}^{M} \left( \sum_{j=1}^{N} d_j^{-1} a_{ij} a_{\ell j} y_j \right) \pi_\ell = b_i + \sum_{j=1}^{N} a_{ij} y_j (\log y_j + d_j^{-1} c_j) \quad .$$

$$i=1,2,3,\ldots,M$$

Denoting

$$r_{i\ell} = \sum_{j=1}^{N} d_j^{-1} a_{ij} a_{\ell j} y_j \qquad \begin{matrix} \ell=1,2,3,\ldots,M \\ i=1,2,3,\ldots,M \end{matrix} \quad (3.8)$$

and

$$s_i = b_i + \sum_{j=1}^{N} a_{ij} y_j (\log y_j + d_j^{-1} c_j) \quad (3.9)$$

$$i=1,2,3,\ldots,M$$

we have

$$\sum_{\ell=1}^{M} r_{i\ell}\pi_{\ell} = s_i \quad . \qquad i=1,2,3,\ldots,M \qquad (3.10)$$

Equation (3.10) is a set of simultaneous equations which can be solved for $\pi_1,\pi_2,\ldots,\pi_M$.

With the above results, we can now define the iterative process for the first method. At each iteration we have a set of values for $x_1,x_2,\ldots,x_N$. At the beginning of the iteration these values are called $y_1,y_2,\ldots,y_N$, and at the end of the iteration the values are $x_1,x_2,\ldots,x_N$. If

$$\frac{x_j-y_j}{y_j}$$

is small for each j, then we say we have converged. The magnitude of "small" depends on the nature of the problem. If

$$\frac{x_j-y_j}{y_j}$$

is not small for some j, then we have not converged and the iteration must be repeated. One iteration consists of the following three steps:

1) Evaluate terms in Eqs. (3.8) and (3.9), these terms depending on $y_1, y_2, \ldots, y_N$;

2) Solve Eq. (3.10) for $\pi_1, \pi_2, \ldots, \pi_M$;

3) Substitute $\pi_1, \pi_2, \ldots, \pi_M$ into (3.7) to get $x_1, x_2, \ldots, x_N$.

For this problem, in this generality, we can say nothing about whether this iterative process converges. In the next section we will show that the chemical equilibrium problem is a special case of this problem, and one for which, with appropriate modification, this method does converge.

## 4. THE FIRST-ORDER METHOD FOR SOLVING THE CHEMICAL EQUILIBRIUM PROBLEM

The chemical equilibrium problem is a special case of the linear-logarithmic programming problem. In order to put Eqs. (3.1) and (3.2) into the form of Eqs. (1.1) and (1.3), we first define

$$N = n+p$$

$$M = m+p$$

where, as stated previously, p is the number of compartments in the problem. Then we define $a_{ij}$, $b_i$, $x_j$, and $c_j$, for $i > m$ and $j > n$, as follows

$$b_i = 0 \qquad i=m+1,m+2,\ldots,M \qquad (4.1)$$

$$c_j = 0 \qquad j=n+1,n+2,\ldots,N \qquad (4.2)$$

$$x_{k+n} = S_k \qquad k=1,2,\ldots,p \qquad (4.3)$$

$$a_{ij} = \begin{cases} 0 & \text{if } i \le m,\ j > n \\ 1 & \text{if } i > m,\ j \le n, \text{ and } [j] = i-m \\ 0 & \text{if } i > m,\ j \le n, \text{ and } [j] \neq i-m \\ -1 & \text{if } i > m,\ j > n, \text{ and } i-m = j-n \\ 0 & \text{if } i > m,\ j > n, \text{ and } i-m \neq j-n\ . \end{cases} \tag{4.4}$$

For all j, we define

$$d_j = \begin{cases} +1 & \text{if } j \le n \\ -1 & \text{if } j > n\ . \end{cases} \tag{4.5}$$

With these definitions, it has been shown [4] that the two problems are identical. Next, we let

$$x_j = y_j + \theta_j \tag{4.6}$$

$$\pi_i = \begin{cases} \pi'_i & i \le m \\ \pi'_i + \log S_{i-m} + 1\ . & i > m \end{cases}$$

Substituting Eqs. (4.1) through (4.6) into (3.7) through (3.10) and simplifying, we have

$$\theta_j = y_j \left[ \sum_{i=1}^{m} a_{ij}\pi'_i - c_j - \log \hat{y}_j + \pi'_{[j]+m} \right] \qquad (4.7)$$

$$j=1,2,\ldots,n$$

$$r_{i\ell} = \begin{cases} \sum_{j=1}^{n} a_{ij}a_{\ell j}y_j & \ell \le m,\ i \le m \\ \sum_{j \in \langle i-m \rangle} a_{\ell j}y_j & \ell \le m,\ i > m \\ \sum_{j \in \langle \ell-m \rangle} a_{ij}y_j & \ell > m,\ i \le m \\ 0 & \ell > m,\ i > m \end{cases} \qquad (4.8)$$

$$s'_i = \begin{cases} b_i + \sum_{j=1}^{n} a_{ij}y_j(c_j + \log \hat{y}_j - 1) & i \le m \\ \sum_{j \in \langle i-m \rangle} y_j(c_j + \log \hat{y}_j) & i > m \end{cases} \qquad (4.9)$$

$$\sum_{\ell=1}^{M} r_{i\ell}\pi'_i = s'_i \,. \qquad i=1,2,\ldots,M \qquad (4.10)$$

The directional derivative of F in the direction $(\theta_1,\theta_2,\ldots,\theta_n)$ is given by [1, Theorem 8.11] to be

$$\sum_{j=1}^{n} \theta_j (c_j + \log \hat{y}_j) \quad . \tag{4.11}$$

But, if we compute $\sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j}$ where by (3.7)

$$\theta_{k+n} = S_k \left[ \pi_{m+k} - \log S_k - 1 \right] = S_k \, \pi'_{m+k} \qquad k=1,2,\ldots,p \tag{4.12}$$

we show, in Appendix B, that

$$\sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j} = - \sum_{j=1}^{n} \theta_j (c_j + \log \hat{y}_j) + \sum_{i=1}^{m} \pi_i \left( b_i - \sum_{j=1}^{n} a_{ij} y_j \right) . \tag{4.13}$$

Thus, if we assume that $(y_1, y_2, \ldots, y_n)$ is feasible, we get the interesting result that the directional derivative of F in the direction $(\theta_1, \theta_2, \ldots, \theta_n)$ is

$$\sum_{j=1}^{n} \theta_j (c_j + \log \hat{y}_j) = - \sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j} \leq 0 \quad . \tag{4.14}$$

However, it is also shown in Appendix B that the equality on the right side of (4.14) holds if and only if the values for $y_j$ are optimal. We further note that if $(y_1, y_2, \ldots, y_n)$ is feasible, then

$$\sum_{j=1}^{n} a_{ij}\theta_j = 0$$

for i = 1,2,...,m. Hence, if $(y_1,y_2,\ldots,y_n)$ is feasible, then $(y_1+\lambda\theta_1, y_2+\lambda\theta_2, \ldots, y_n+\lambda\theta_n)$ will be feasible for any $\lambda$ for which each $y_j + \lambda\theta_j$ is positive.

We now state the first-order chemical equilibrium algorithm:

1) Calculate $(\theta_1,\theta_2,\ldots,\theta_n)$ using Eqs. (4.7) through (4.10).

2) Calculate the directional derivative of F in the direction $(\theta_1,\theta_2,\ldots,\theta_n)$ as given by Eq. (4.11); if this quantity is not negative, we are done.

3) Calculate

$$\epsilon = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(\frac{\theta_j}{y_j}\right)^2}$$

$\epsilon$ is a number that represents the root-mean-square error in $(y_1,y_2,\ldots,y_n)$. If $\epsilon$ is less than some given number (say, 0.001), we are done.

4) Calculate the ratio $-y_j/\theta_j$ for every j for which $\theta_j < 0$. Let $\lambda_1$ be the minimum of all such ratios and let $\lambda = \min(1, \beta\lambda_1)$, where $\beta$ is a number less than 1 but close to 1 (say, 0.99). We now perform the following steps until the test at c) below is satisfied:

   a) Let $z_j = y_j + \lambda\theta_j$;

   b) Compute the directional derivative of F at $z_j$ in the direction $(\theta_1, \theta_2, \ldots, \theta_n)$: $f(\lambda) = \theta_j(c_j + \log \hat{z}_j)$;

   c) If $f(\lambda) \leq 0$, go directly to step 5);

   d) Replace $\lambda$ by $\gamma\lambda$, where $0 < \gamma < 1$, e.g., $\gamma = \frac{1}{2}\sqrt{2}$.

5) Finally, replace $y_j$ by $y_j + \lambda\theta_j$ for $j = 1,2,\ldots,n$.

Steps 1-5 are repeated until either the test in step 2 or the test in step 3 is satisfied.

If this process terminates, the solution will be optimal within the specified limits of accuracy. It may happen that the process does not terminate. Since the objective function F is convex* and assuming infinite computational accuracy, non-termination can occur only because the values chosen for $\lambda$ become smaller on every

---

*Ref. 1, Theorem 8.13; Ref. 5.

iteration. This will occur only if some $y_j$ is approaching zero, and hence $(y_1, y_2, \ldots, y_n)$ is approaching a point at which, if it were the optimal solution, the problem would be degenerate. It is possible for this to happen for a non-degenerate problem for which the initial solution chosen was too far from the optimal solution. Convergence can be guaranteed by imposing the condition that the value of F at the initial solution be less than the value of F at any feasible, degenerate point. However, it is not practical to impose this condition on the initial solution since it may be very difficult to find such a point. In practice, it has been found that round-off errors cause more difficulty than the possible selection of a poor initial solution.

## 5. THE LINEAR-LOGARITHMIC PROGRAMMING PROBLEM, SECOND-ORDER METHOD

In the first-order method, presented in Sec. 3, the iterative process was initiated with an estimate of the value of $x_1, x_2, \ldots, x_N$. In the second-order method, we assume that the problem is as defined by Eqs. (3.1) and (3.2), but that we have initial estimates for the values of $\pi_1, \pi_2, \ldots, \pi_M$. Let us denote these estimates by $\lambda_1, \lambda_2, \ldots, \lambda_M$. The $x_j$ can then be evaluated by Eq. (3.5), substituting $\lambda_i$ for $\pi_i$. These $x_j$, however, probably will not satisfy Eq. (3.2). The problem of the second-order method is to find numbers $\Delta\lambda_1, \Delta\lambda_2, \ldots, \Delta\lambda_M$, such that

$$\pi_i = \lambda_i + \Delta\lambda_i \qquad i=1,2,\ldots,M \qquad (5.1)$$

when substituted into (3.5) will give $x_j$ that satisfy (3.2).

In order to accomplish this, we first use the $x_j$ calculated from Eq. (3.5) to get

$$g_i = b_i - \sum_{j=1}^{N} a_{ij} x_j \qquad i=1,2,\ldots,M \qquad (5.2)$$

where $g_i$ represents the amount that equation i is in error. Next, we evaluate

$$\frac{\partial g_i}{\partial \lambda_\ell}$$

by

$$\frac{\partial g_i}{\partial \lambda_\ell} = \frac{\partial}{\partial \lambda_\ell}\left[b_i - \sum_{j=1}^{N} a_{ij}x_j\right] = -\sum_{j=1}^{N} a_{ij}\frac{\partial x_j}{\partial \lambda_\ell}$$

$$= -\sum_{j=1}^{N} a_{ij}\frac{\partial}{\partial \lambda_\ell}\left[\exp\left(d_j^{-1}\sum_{h=1}^{M}\lambda_h a_{hj} - d_j^{-1}c_j - 1\right)\right]$$

$$= -\sum_{j=1}^{N} a_{ij}d_j^{-1}x_j a_{\ell j} = -r_{\ell i} \qquad (5.3)$$

where $r_{\ell i}$ is given by Eq. (3.8). If we make a very small change, $d\lambda_1$, $d\lambda_2$,..., in $\lambda_1,\lambda_2,\ldots$, the change in $g_1,g_2,\ldots$, is given by $dg_1,dg_2,\ldots$, where

$$dg_i = +\sum_{\ell=1}^{M}\frac{\partial g_i}{\partial \lambda_\ell}d\lambda_\ell \qquad i=1,2,\ldots,M$$

or

$$dg_i = -\sum_{\ell=1}^{M} r_{\ell i} d\lambda_\ell \,. \qquad i=1,2,\ldots,M \qquad (5.4)$$

We would want $dg_i$ to be equal to $-g_i$ as computed by Eq. (5.2). If we make the approximation that

$$\frac{\partial g_i}{\partial \lambda_\ell}$$

is constant over the domain considered, we can set $dg_i = -g_i$, let $d\lambda_\ell = \Delta\lambda_\ell$, and write

$$g_i = \sum_{\ell=1}^{M} r_{\ell i} \Delta\lambda_1 \,. \qquad i=1,2,\ldots,M \qquad (5.5)$$

Equation (5.5) consists of M equations in the M unknowns $\Delta\lambda_1, \Delta\lambda_2, \ldots, \Delta\lambda_M$. We may thus solve Eq. (5.5) for $\Delta\lambda_1, \Delta\lambda_2, \ldots, \Delta\lambda_M$ and compute $\pi_1, \pi_2, \ldots, \pi_M$ from (5.1). If the assumption about

$$\frac{\partial g_i}{\partial \lambda_\ell}$$

being constant over the domain considered was correct, then

the $x_j$ computed from (3.5) with these values for $\pi_i$ will satisfy (3.2). However, in general, they will not satisfy (3.2), but, if we were close enough to the solution so that the

$$\frac{\partial g_i}{\partial \lambda_\ell}$$

did not vary greatly in the domain considered, then the new values for $x_j$ should come closer to satisfying (3.2) than did the first set of $x_j$.

With this assumption, we may now state the iterative process:

a) Using the values at hand for $\pi_1, \pi_2, \ldots, \pi_M$, evaluate (3.5).

b) Using the values for $x_j$ obtained in step a, evaluate (5.2). If the $|g_i|$ are sufficiently small, we are done.

c) Compute $r_{i\ell}$ using (3.8) and solve (5.5) for $\Delta\lambda_i$.

d) Denoting the $\pi_i$ in step a by $\lambda_i$, we get new $\pi_i$ by (5.1).

Steps a-d are repeated until the $|g_i|$, computed in step b, are sufficiently small, or until they show no more improvement.

There is no proof of convergence for this method. In fact, the method presented here is unlikely to converge unless the starting values of $\pi_1, \pi_2, \ldots, \pi_M$ are very good, and even then there may be no convergence. This method may be used on the chemical equilibrium problem after the first-order method has resulted in a reasonably good solution. If the $\pi_i$ obtained from (3.10) in the final iteration of the first-order method are used to initiate the second-order method, the accuracy produced by the second-order method will generally be better than that which could be achieved by use of the first-order method only.

## 6. THE SECOND-ORDER CHEMICAL EQUILIBRIUM ALGORITHM

In order that the second-order linear-logarithmic method be set in the form of a chemical equilibrium problem, the same definitions as given in Sec. 4--i.e., Eqs. (4.1) through (4.5)--are used here. Since the second-order method is best used after the first-order method has been applied, the initial values of $\pi_i$ for the second-order method must be specified. The first-order method gives a set of $\pi_i'$ which are related to $\pi_i$ by Eq. (4.6). The $\pi_i$ computed by means of (4.6) are appropriate initial values for the second-order method. Using these initial values for $\pi_i$, the second-order chemical equilibrium algorithm is an iterative process for which each iteration consists of the following steps:

1) Using the current values for $(\pi_1,\pi_2,\ldots,\pi_M)$, evaluate $x_1,x_2,\ldots,x_n$ by means of (3.5).
2) Calculate $g_1,g_2,\ldots,g_m$ by means of (5.2) and set $g_{m+1},g_{m+2},\ldots,g_M$ equal to zero.
3) Compute $r_{ij}$ from (4.8) and solve (5.5) for $\Delta\lambda_1,\Delta\lambda_2,\ldots,\Delta\lambda_M$.
4) Let

$$P = \max_{i=1}^{M} |\Delta\lambda_i| .$$

If $P < \delta$, where $\delta$ is a small positive number such as $10^{-5}$, we are done; otherwise, let $Q = \min\left(\frac{1}{P}, 1\right)$.

5) Replace $\pi_i$ by $\pi_i + Q\ \Delta\lambda_i$ for $i = 1,2,\ldots,M$.

Steps 1-5 are repeated until the test at 4) is satisfied. P should decrease at every iteration; however, when the values for $\pi_i$ get close to their optimal values, P may not become zero due to round-off error. In order to prevent an endless repetition of steps 1-5 due to the selection of too small a $\delta$, we can test P against the value of P at the previous iteration. If this value has increased over the previous iteration, it can be assumed that this method has obtained as accurate a solution as possible, and we can terminate the iteration process. The reason for inserting the factor Q above is to prevent the $\pi_i$ from varying too much on one iteration.

## 7. SUMMARY OF THE COMPUTATION PROCEDURE

The best method for starting the solution of the chemical equilibrium problem depends on whether an estimate for the solution vector is available. The projection method should be used when the problem being solved is a slight variation from a problem previously solved, and in this case, the values used for $y_j$ in (2.9 - 2.12) should be the solution vector to the previous problem. Even when the estimate is no better than an intuitive guess, the projection method may still be used. The linear programming method, then, may be used as a back-up if the projection method produces a non-positive component. Of course, if no estimate is available, the linear programming method would be used immediately to provide an estimate.

The recommended procedure is, then, to use the first-order method until either no further progress can be made with this method or until the amount of change becomes small from iteration to iteration, and then to use the second-order method. It has been found that, for reasonably large problems (say m = 30, n = 100), the point at which progress ceases in the first-order method usually occurs when the indicated corrections to the components

of the solution vector average about one per cent of the components; that is, when (3.5) is accurate to about two significant digits. A switch to the second-order method at this point usually yields quite accurate results in two iterations of the second-order method. The second-order method usually satisfies (1.1) to an accuracy of about five significant digits on a machine that carries eight significant digits. This accuracy is typically about three orders of magnitude above what is usually obtained in experimental data.

To summarize, the typical procedure for solving a chemical equilibrium problem is the following:

1) If an estimate is available, use the projection method to obtain a feasible estimate.

2) If step 1 yields a strictly positive estimate, go to step 3, but if the projection method yields non-positive components, or if there was no initial estimate, then use the linear programming method to get an estimate.

3) Use the first-order method until one of the tests described in Section 4 is satisfied.

4) Use the second-order method as described in Section 6.

# Appendix A

## A FORTRAN-IV PROGRAM FOR SOLVING THE CHEMICAL EQUILIBRIUM PROBLEM

### GENERAL DESCRIPTION

The program described here is a set of FORTRAN-IV subroutines for solving chemical equilibrium problems. The calling sequence used is merely the statement:

```
CALL SOLVE
```

Communication of data into and out of the subroutines is accomplished by a block common statement:

```
 COMMON/SLVE/IV(30),TOL(20),NR(55,2),B(55),KN(120),X(121),C(121),
1  KL(26),NAM(25,2),A(55,121),PIE(65),V1(65),V2(65),V3(65),
2  V4(65),XMF(120),X1(121),X2(121),X3(121),XBAR(25),R(65,65)
```

The data that must be input before CALL SOLVE is executed consist of the following:

| COMMON Location | Quantity |
|---|---|
| IV(1) | m |
| IV(2) | M ( = m+p) |
| IV(3) | p |
| IV(4) | n |
| IV(6) | Number of the output unit. |

| COMMON Location | Quantity |
|---|---|
| IV(7) | Print flag: -1 = minimal amount of messages; 0 = one message per iteration step; +1 = all messages. |
| IV(9) | Maximum number of iterations to be allowed. |
| B(i) | $b_i$, i = 1,2,...,m. |
| X(j) | $y_j$, j=1,2,...,m, where $y_j$ is the initial estimate of the solution. If no estimate is available, set X(J) = 0. |
| C(j) | $c_j$, j=1,2,...,n. |
| A(i,j) | $a_{ij}$, i=1,2,...,m; j=1,2,...,n. |

In addition, all components in one compartment must have consecutive subscripts. That is, components $1,2,3,\ldots,k_1$ must be in compartment 1; components $k_1+1$, $k_1+2$, ..., $k_2$ must be in compartment 2; ...; and components $k_{p-1}+1$, $k_{p-1}+2$, ..., $k_p$ must be in compartment p. These k's are communicated to the subroutines by setting

$$KL(1) = 1$$

$$KL(2) = k_1+1$$

$$KL(3) = k_2+1$$

$$\vdots$$

$$KL(p) = k_{p-1}+1$$

$$KL(p+1) = k_p+1 \ .$$

In other words, KL(k) is the number of the first component in compartment k, and KL(p+1) is equal to n+1.

The above are the only numbers that need be set in order that CALL SOLVE will solve the chemical equilibrium problem. However, in order that the program can write messages, in cases of infeasibility, etc., names for the rows, components, and compartments may be input:

| COMMON Location | Quantity |
|---|---|
| NR(I,1), NR(I,2) | Two-word row name for row I. |
| KN(J) | One-word component name for component J. |
| NAM (K,1), NAM(K,2) | Two-word compartment name for compartment K. |

In addition, TOL(1) through TOL(5) are tolerances used by the program. If they are zero when the program is entered, they are set by the subroutines to nominal values. These values may also be set by the user of the subroutines, in which case the nominal values will not be set in the subroutines. These tolerances are the following:

| Tolerance | Nominal Value | Meaning |
|---|---|---|
| TOL(1) | 0.01 | $\epsilon$ in step 3 of the first-order method (see Sec. 4). |

| Tolerance | Nominal Value | Meaning |
|---|---|---|
| TOL(2) | $10^{-5}$ | $\delta$ in step 4 of the second-order method (see Sec. 6). |
| TOL(3) | $10^{-12}$ | Minimum value any $x_j$ is allowed to have. |
| TOL(4) | $10^{-6}$ | Minimum starting value that any component will have is the lesser of TOL(4) and $\frac{1}{2}y_{n+1}$ (see Sec. 2). |
| TOL(5) | $10^{-8}$ | Problem is assumed to be degenerate if any $S_k$ becomes less than TOL(5). |

With the above as input, the statement CALL SOLVE will cause an attempt to solve the chemical equilibrium problem. If, upon completion of this attempt, a solution is obtained, the cell

IV(10)

will contain a 1 and the following data will be in storage:

| COMMON Location | Data |
|---|---|
| X(i) | $x_i$, i=1,2,...,n (the solution). |
| XBAR(k) | $S_k$, k=1,2,...,p. |
| PIE(i) | $\pi_i$, i=1,2,...,m. |
| XMF(i) | $\hat{x}_i$, i=1,2,...,n. |

If IV(10) is not 1, the subroutines have failed to solve the chemical equilibrium problem. The reason for this failure is written on output unit IV(6). In such a case, X(i) will contain the latest value of these quantities.

## SUBROUTINES

There are nine subroutines in the set used for the solution of the chemical equilibrium problem. A brief description of these subroutines follows.

### 1. Subroutine SOLVE

SOLVE is the master subroutine, and is divided into four functional segments. Each segment calls other subroutines which do specific tasks. The four segments are:

a) The projection and linear programming routines for obtaining the initial solution (lines 18-42).

b) The first-order method (lines 43-122).

c) The second-order method (lines 123-163).

d) Output messages (lines 164-203).

### 2. Subroutine BAR

BAR calculates the $S_k$.

3. Subroutine BERROR

BERROR calculates

$$g_i = b_i - \sum_{j=1}^{N} a_{ij}x_j . \qquad i=1,2,\ldots,M$$

4. Subroutine DEL

DEL sets

$$w_j = \sum_{i=1}^{m} a_{ij}q_i . \qquad j=1,2,\ldots,n$$

5. Subroutine RCALC

RCALC calculates the $r_{ij}$ array (4.8).

6. Subroutine CLOG

CLOG computes

$$\alpha_j = c_j + \log \hat{x}_j . \qquad j=1,2,\ldots,n$$

7. Subroutine LP

LP sets up the linear programming problems.

8. Subroutine SIMPLE

SIMPLE solves the linear programming problems. Information is communicated to this routine via a

calling sequence rather than by COMMON as in subroutines 1-7. The dimension of A in SIMPLE should agree with the dimension of A in the first seven subroutines, but all other dimensions are dummy statements.

9. Subroutine MATINV

MATINV solves simultaneous equations. As in SIMPLE, no COMMON is used. The dimension of A in MATINV should agree with that of R (not A) in SOLVE. All other dimensions are singly subscripted and are irrelevant as to magnitude.

* * *

Each of the first seven subroutines has a COMMON statement which should be the same in all seven. The dimensions of the variables in this COMMON statement may be set to the values for the largest problem to be solved. With m, M, p, and n as previously defined, these dimensions must be at least:

| Symbol | Minimum Dimension |
|---|---|
| IV | 30 |
| TOL | 20 |
| NR | (m,2) |
| B | m |
| KN | n |
| X | n+1 |
| C | n+1 |
| KL | p+1 |
| NAM | (p,2) |
| A | (m,n+1) |
| PIE | M |
| V1,V2,V3,V4 | M |
| XMF | n |
| X1,X2,X3 | n+1 |
| XBAR | p |
| R | (M,M) . |

A listing of these subroutines follows. This listing does not necessarily represent an actual program. The language used was that version of FORTRAN described in [6]. The machine used for the solution of chemical equilibrium problems was the IBM-7044, which uses a floating-point number with eight bits for the exponent and 28 bits for the sign and mantissa.

## LISTING

```
      SUBROUTINE SOLVE                                                  S0001
      COMMON/SLVE/IV(30),TOL(20),NR(55,2),B(55),KN(120),X(121),C(121),  S0002
     1  KL(26),NAM(25,2),A(55,121),PIE(65),V1(65),V2(65),V3(65),        S0003
     2  V4(65),XMF(120),X1(121),X2(121),X3(121),XBAR(25),R(65,65)       S0004
      INTEGER PF                                                        S0005
      EQUIVALENCE  (TOL(3),XMIN),(TOL(4),XSTART),(TOL(5),BARMIN)        S0006
      EQUIVALENCE (IV(1),M),(IV(2),MEND),(IV(3),NCOMP),(IV(4),N,NTOT),  S0007
     1 (IV(5),NIT),(IV(6),NOT),(IV(7),PF),(IV(8),ITER),(IV(9),ITMAX),   S0008
     2  (IV(10),IERROR),(IV(11),LASTCP),(IV(12),KE)                     S0009
      DIMENSION DX(1),ALPHA(1),TH(1),G(1)                               S0010
      EQUIVALENCE (G,V1),(DX,X1),(ALPHA,X2),(TH,X3)                     S0011
      IF (TOL(1).LE.0.0)   TOL(1) = 0.01                                S0012
      IF (TOL(2).LE.0.0)   TOL(2) = 1.E-5                               S0013
      IF (XMIN.LE.0.0) XMIN = 1.E-12                                    S0014
      IF (BARMIN.LE.0.0)  BARMIN = 1.E-8                                S0015
      IF (ITMAX.LE.0)  ITMAX = 40                                       S0016
      DO 152  J = 1, NTOT                                               S0017
        IF (X(J).LE.0.) GO TO 5                                         S0018
  152 CONTINUE                                                          S0019
C   IF X IS STRICTLY POSITIVE,  BEGIN PROJECTION                        S0020
      CALL BAR( X,XBAR )                                                S0021
    2 CALL BERROR(ERR)                                                  S0022
      CALL RCALC                                                        S0023
      CALL MATINV(R,MEND,G,-1,V2,V3,V4,KE)                              S0024
      IF (KE.NE.0)  GO TO 5                                             S0025
      CALL DEL (DX,G)                                                   S0026
      DO 3 K = 1,NCOMP                                                  S0027
        KTA = KL(K)                                                     S0028
        KTB = KL(K+1)-1                                                 S0029
        MK = M + K                                                      S0030
        DO 4  J = KTA,KTB                                               S0031
          X(J) = X(J) *  ( 1. + DX(J) + G(MK) )                         S0032
          IF  (X(J).LE.0.)  GO TO 5                                     S0033
    4   CONTINUE                                                        S0034
    3 CONTINUE                                                          S0035
      GO TO 7                                                           S0036
C   LINEAR PROGRAMMING ROUTINE                                          S0037
    5 CALL LP(KF)                                                       S0038
      IF (KF.NE.0) GO TO 10006                                          S0039
    7 CALL BAR(X,XBAR)                                                  S0040
      CALL CLOG(X,XBAR)                                                 S0041
      FE2 = 1.E+20                                                      S0042
C   FIRST ORDER METHOD LOOP                                             S0043
      DO 899 ITER=1,ITMAX                                               S0044
        CALL BERROR(ERR)                                                S0045
        DO 7110 I=1,MEND                                                S0046
          PIE(I) = 0.                                                   S0047
 7110   CONTINUE                                                        S0048
        DO 7111  K = 1, NCOMP                                           S0049
          KTA = KL(K)                                                   S0050
          KTB = KL(K+1) - 1                                             S0051
          MK = M + K                                                    S0052
          DO 7112 J = KTA, KTB                                          S0053
            AX = ALPHA(J) * X(J)                                        S0054
            PIE(MK) = PIE(MK)  + AX                                     S0055
            DO 7113 I = 1,M                                             S0056
              PIE(I) = PIE(I) + AX * A(I,J)                             S0057
 7113         CONTINUE                                                  S0058
 7112       CONTINUE                                                    S0059
 7111     CONTINUE                                                      S0060
```

```
      DO 7114 I = 1,M                                                    S0061
        PIE(I) = G(I) + PIE(I)                                           S0062
7114  CONTINUE                                                           S0063
      CALL RCALC                                                         S0064
      CALL MATINV(R,MEND,PIE,-1,V2,V3,V4,KE)                             S0065
      IF(KE.NE.0) GO TO 10003                                            S0066
      DMAX = 1.E+20                                                      S0067
      CALL DEL(TH,PIE)                                                   S0068
7105  GNORM=0.                                                           S0069
      TDA = 0.                                                           S0070
      FE = 0.                                                            S0071
      DO 7104  K=1,NCOMP                                                 S0072
        MK = M + K                                                       S0073
        KTA = KL(K)                                                      S0074
        KTB = KL(K+1)  -1                                                S0075
        DO 7103 J = KTA, KTB                                             S0076
          TH(J) =     TH(J) +PIE(MK)- ALPHA(J)                           S0077
          GNORM = GNORM + TH(J) **2                                      S0078
          TH(J) = TH(J) * X(J)                                           S0079
          TDA = TDA + TH(J) * ALPHA(J)                                   S0080
          IF (X(J).LT.-DMAX*TH(J)) DMAX = -X(J)/TH(J)                    S0081
          FE = FE + X(J) * ALPHA(J)                                      S0082
7103      CONTINUE                                                       S0083
7104  CONTINUE                                                           S0084
      EPS=   SQRT ( GNORM/FLOAT (NTOT) )                                 S0085
      DFE = FE - FE2                                                     S0086
      FE2 = FE                                                           S0087
      IF (ITER.EQ.1) GO TO 7120                                          S0088
      ITR = ITER - 1                                                     S0089
      IF(PF.GE.0) WRITE(NOT,799) ITR, DFE,OPTL,EPS                       S0090
7120  OPTL =AMIN1  ( 1., .99*DMAX )                                      S0091
      IF(PF.GT.0)WRITE(NOT,8241)          DMAX,OPTL,TDA,ERR              S0092
      IF (EPS.LE.TOL(1))  GO TO 8269                                     S0093
 826  IF  (TDA.GE.0.) GO TO 8267                                         S0094
8260  DO 8265  II =1,54                                                  S0095
        DO 8301 J = 1,N                                                  S0096
          DX(J) = AMAX1(X(J) + OPTL*TH(J)      ,XMIN)                    S0097
8301      CONTINUE                                                       S0098
          CALL BAR(DX,XBAR)                                              S0099
          CALL CLOG(DX,XBAR)                                             S0100
          TDA = 0.                                                       S0101
          DO 8266 J = 1,NTOT                                             S0102
            TDA = TDA + TH(J)*ALPHA(J)                                   S0103
8266      CONTINUE                                                       S0104
          IF(PF.GT.0)WRITE(NOT, 8262)II,OPTL,     TDA                    S0105
          IF ( TDA.LT.0.)  GO TO 828                                     S0106
8264      OPTL = OPTL /1.4142                                            S0107
8265  CONTINUE                                                           S0108
      CALL BAR(X,XBAR)                                                   S0109
      GO TO 8271                                                         S0110
 828  DO 8281 J =1,NTOT                                                 S0111
        X(J) = DX(J)                                                     S0112
8281  CONTINUE                                                           S0113
      FE = 0.                                                            S0114
      DO 8231 J=1,N                                                      S0115
        FE = FE + ALPHA(J)*X(J)                                          S0116
8231  CONTINUE                                                           S0117
8288  CALL SSWTCH(5,LABEL)                                               S0118
      IF (LABEL.NE.2) GO TO 10004                                        S0119
 899 CONTINUE                                                            S0120
```

```
C  END OF FIRST ORDER METHOD LOOP                                          S0121
      GO TO 10002                                                         S0122
 6000 ITER1 = ITER + 1                                                    S0123
      PMAX  = 1.E+20                                                      S0124
      PMAX1 = 1.E+21                                                      S0125
C  SECOND ORDER METHOD LOOP                                               S0126
      DO 6002  ITER = ITER1,ITMAX                                         S0127
        CALL DEL(DX,PIE)                                                  S0128
        DO 6003  K =1,NCOMP                                               S0129
          MTA = KL(K)                                                     S0130
          MTB = KL(K+1) - 1                                               S0131
          DO 6010 J = MTA,MTB                                             S0132
            XMF(J)  = EXP (  DX(J) -C(J) )                                S0133
            X(J) = XMF(J)*XBAR(K)                                         S0134
 6010     CONTINUE                                                        S0135
        IF  (XBAR(K).LE.BARMIN)  GO TO 10005                              S0136
 6003   CONTINUE                                                          S0137
        IF (PMAX.LE.TOL(2).OR.(PMAX.GE.PMAX1.AND.PMAX.GE.PMAX2) )         S0138
     1    GO TO 10001                                                     S0139
        CALL BERROR(ERR)                                                  S0140
 6006   CALL RCALC                                                        S0141
        CALL MATINV(R,MEND,G,  -1,V2,V3,V4,KE)                            S0142
        IF(KE.NE.0) GO TO 10003                                           S0143
        PMAX2 = PMAX1                                                     S0144
        PMAX1 = PMAX                                                      S0145
        PMAX = 0.                                                         S0146
        DO 6004 I = 1,MEND                                                S0147
          PMAX =AMAX1 ( PMAX, ABS (G(I)) )                                S0148
 6004   CONTINUE                                                          S0149
        IF (PMAX.EQ.0.0) GO TO 10001                                      S0150
        ZM =AMIN1 ( 1./PMAX,1.)                                           S0151
        DO 6005 I =1,M                                                    S0152
          PIE(I) = PIE(I) + ZM* G(I)                                      S0153
 6005   CONTINUE                                                          S0154
        DO 6011 K = 1,NCOMP                                               S0155
          MK = M+K                                                        S0156
          XBAR(K) = XBAR(K)* EXP ( ZM * G(MK) )                           S0157
 6011   CONTINUE                                                          S0158
        IF (PF.GE.0) WRITE(NOT,6099) ITER,PMAX,ERR                        S0159
        CALL SSWTCH(5,LABEL)                                              S0160
        IF (LABEL.NE.2) GO TO 10004                                       S0161
 6002 CONTINUE                                                            S0162
C  END OF SECOND ORDER METHOD LOOP                                        S0163
10002 IERROR = 2                                                          S0164
      WRITE(NOT,20002)                                                    S0165
20002 FORMAT(27H ITERAION LIMIT EXCEEDED    )                             S0166
      ITER = ITMAX                                                        S0167
      GO TO 10000                                                         S0168
10003 IERROR = 3                                                          S0169
      WRITE(NOT,20003) KE                                                 S0170
20003 FORMAT(21H R MATRIX HAS NULLITY,I3)                                 S0171
      GO TO 10000                                                         S0172
10004 IERROR = 4                                                          S0173
      WRITE(NOT,20004)                                                    S0174
20004 FORMAT(56H SOLVE ROUTINE TERMINATED BECAUSE SENSE SWITCH 5 IS DOWN  S0175
     1)                                                                   S0176
      GO TO 10000                                                         S0177
10005 IERROR = 5                                                          S0178
      WRITE(NOT,20005)    NAM(K,1),NAM(K,2)                               S0179
20005 FORMAT(13H COMPARTMENT ,2A6,10H TOO SMALL )                         S0180
```

```
      LASTCP = K                                                        S0181
      GO TO 10000                                                       S0182
10006 IERROR = 6                                                        S0183
      GO TO 10000                                                       S0184
10001 IERROR = 1                                                        S0185
10000 RETURN                                                            S0186
 8241 FORMAT(15H     LAMBDA MAX=1PE12.4,13H, OPT LAMBDA=E10.3,6H, TDA=E12  S0187
     1.5,16H, MAX ROW ERROR=E12.5)                                      S0188
 8267 IF (PF.GE.0) WRITE (NOT,8268) ITER                                S0189
 8268    FORMAT(10H ITERATION,I4,30H POSITIVE TDA, GO TO METHOD 2 )     S0190
        GO TO 6000                                                      S0191
 8269 IF (PF.GE.0) WRITE (NOT,8270) ITER                                S0192
 8270 FORMAT(10H ITERATION,I4,42H AV THETA LESS THAN TOL(1), GO TO METHO  S0193
     1D 2)                                                              S0194
      GO TO 6000                                                        S0195
 8271 IF (PF.GE.0) WRITE (NOT,8272) ITER                                S0196
 8272 FORMAT(10H ITERATION,I4,36H STEP SIZE TOO SMALL, GO TO METHOD 2)   S0197
      GO TO 6000                                                        S0198
 8262     FORMAT(10X, 4HSTEP,I2, 9H  LAMBDA=1PE10.3,6H, TDA=E15.6)      S0199
  799   FORMAT(10H ITERATION,I4,24H  CHANGE IN FREE ENERGY=1PE15.8,12H    S0200
     1STEP SIZE=E15.8,10H AV THETA=E12.5)                               S0201
 6099 FORMAT(10H ITERATION,I4,19H MAX CHANGE IN PIE=1PE15.8,15H MAX ROW  S0202
     1ERROR=E15.8                                                       S0203
      END                                                               S0204
```

```
      SUBROUTINE BAR(W,WBAR)                                              W0001
      COMMON/SLVE/IV(30),TOL(20),NR(55,2),B(55),KN(120),X(121),C(121),    W0002
     1  KL(26),NAM(25,2),A(55,121),PIE(65),V1(65),V2(65),V3(65),           W0003
     2  V4(65),XMF(120),X1(121),X2(121),X3(121),XBAR(25),R(65,65)          W0004
      EQUIVALENCE (IV(1),M),(IV(2),MEND),(IV(3),NCOMP),(IV(4),N,NIO1),    W0005
     1 (IV(5),NIT),(IV(6),NOT),(IV(7),PF),(IV(8),IIER),(IV(9),ITMAX),     W0006
     2  (IV(10),IERROR),(IV(11),LASTCP),(IV(12),KE)                       W0007
      DIMENSION W(1),WBAR(1)                                              W0008
    7 DO 701 K = 1,NCOMP                                                  W0009
        KTA = KL(K)                                                       W0010
        KTB = KL(K+1) - 1                                                 W0011
        WBAR(K) = 0.                                                      W0012
        DO 702 J = KTA,KTB                                                W0013
          WBAR(K) = WBAR(K) + W(J)                                        W0014
  702   CONTINUE                                                          W0015
  701 CONTINUE                                                            W0016
      END                                                                 W0017
```

```
      SUBROUTINE BERROR(BMAX)                                           B0001
      COMMON/SLVE/IV(30),TOL(20),NR(55,2),P(55),KN(120),X(121),C(121),  B0002
     1  KL(26),NAM(25,2),A(55,121),PIE(65),V1(65),V2(65),V3(65),         B0003
     2  V4(65),XMF(120),X1(121),X2(121),X3(121),XBAR(25),R(65,65)        B0004
      EQUIVALENCE (IV(1),M),(IV(2),MEND),(IV(3),NCOMP),(IV(4),N,NTOT),  B0005
     1 (IV(5),NIT),(IV(6),NCT),(IV(7),PF),(IV(8),IIER),(IV(9),IIMAX),   B0006
     2  (IV(10),IERROR),(IV(11),LASTCP),(IV(12),KE)                     B0007
      DIMENSION G(1)                                                    B0008
      EQUIVALENCE (G,V1)                                                B0009
      DO 101 I = 1,M                                                    B0010
        ZT = 0.                                                         B0011
        DO 102 J = 1,N                                                  B0012
          IF(A(I,J).NE.0.) ZT = ZT - X(J) * A(I,J)                      B0013
  102   CONTINUE                                                        B0014
        G(I) = ZT + B(I)                                                B0015
  101 CONTINUE                                                          B0016
      DO 110 K = 1,NCOMP                                                B0017
        ZT = 0.                                                         B0018
        MTA = KL(K)                                                     B0019
        MTB = KL(K+1) - 1                                               B0020
        DO 111 J = MTA,MTB                                              B0021
          ZT = ZT + X(J)                                                B0022
  111   CONTINUE                                                        B0023
        MK = M + K                                                      B0024
        G(MK) = XBAR(K) - ZT                                            B0025
  110 CONTINUE                                                          B0026
      BMAX = 0.                                                         B0027
      DO 120 I = 1,MEND                                                 B0028
        IF (ABS(G(I)).GT. ABS(BMAX) ) BMAX = G(I)                       B0029
  120 CONTINUE                                                          B0030
      RETURN                                                            B0031
      END                                                               B0032
```

```
      SUBROUTINE DEL(W,G)                                                    D0C01
      COMMON/SLVE/IV(30),TOL(25),NR(55,2),B(55),KN(120),X(121),C(121),       D0C02
     1  KL(26),NAM(25,2),A(55,121),PIL(65),V1(65),V2(65),V3(65),             D0C03
     2  V4(65),XMF(125),X1(121),X2(121),X3(121),XBAR(25),R(65,65)            D0C04
      EQUIVALENCE (IV(1),M),(IV(2),MEND),(IV(3),NCOMP),(IV(4),N,NIOI),       D0C05
     1 (IV(5),NIT),(IV(6),NOT),(IV(7),PF),(IV(8),ITER),(IV(9),ITMAX),        D0C06
     2  (IV(10),IERROR),(IV(11),LASTCP),(IV(12),KE)                          D0C07
      DIMENSION W(1),G(1)                                                    D0C08
      DO 20 J = 1,N                                                          D0C09
        WW=0.                                                                D0C10
        DO 10 I = 1,M                                                        D0C11
          IF (A(I,J).NE.0.)  WW = WW + A(I,J) * G(I)                         D0C12
10      CONTINUE                                                             D0C13
        W(J) = WW                                                            D0C14
20    CONTINUE                                                               D0C15
      RETURN                                                                 D0C16
      END                                                                    D0C17
```

```
      SUBROUTINE RCALC                                                    R0001
      COMMON/SLVE/IV(30),TOL(20),NR(55,2),3(55),KN(120),X(121),C(121),    R0002
     1  KL(26),NAM(25,2),A(55,121),PIE(65),V1(65),V2(65),V3(65),           R0003
     2  V4(65),XMF(120),X1(121),X2(121),X3(121),XBAR(25),R(65,65)          R0004
      EQUIVALENCE (IV(1),M),(IV(2),MEND),(IV(3),NCOMP),(IV(4),N,NTOT),    R0005
     1 (IV(5),NIT),(IV(6),NOT),(IV(7),PF),(IV(8),ITER),(IV(9),ITMAX),     R0006
     2  (IV(10),IERROR),(IV(11),LASTCP),(IV(12),KE)                       R0007
COMPUTE R                                                                 R0008
      DO 1 I = 1,MEND                                                     R0009
        DO 2 J =1,I                                                       R0010
          R(I,J) = 0.0                                                    R0011
    2   CONTINUE                                                          R0012
    1 CONTINUE                                                            R0013
      DO 10  K = ',NTOT                                                   R0014
        DO 11  I=1,M                                                      R0015
          IF (A(I,K).EQ.0.) GO TO 11                                      R0016
          AIKX = A(I,K) * X(K)                                            R0017
          DO 12  J =1,I                                                   R0018
            IF (A(J,K).NE.0.) R(I,J) = A(J,K) * AIKX + R(I,J)             R0019
   12     CONTINUE                                                        R0020
   11   CONTINUE                                                         R0021
   10 CONTINUE                                                          R0022
      DO 20 K = 1,NCOMP                                                   R0023
        IH = K + M                                                        R0024
        MTA =KL(K)                                                        R0025
        MTB =KL(K+1) - 1                                                  R0026
        DO 21 L =MTA,MTB                                                  R0027
          DO 22 J =1,M                                                    R0028
            IF (A(J,L).NE.0.)  R(IH,J) = R(IH,J) + A(J,L) * X(L)          R0029
   22     CONTINUE                                                        R0030
   21   CONTINUE                                                         R0031
   20 CONTINUE                                                          R0032
      DO 30 J = 2,MEND                                                    R0033
        JL = J-1                                                          R0034
        DO 31 I = 1,JL                                                    R0035
          R(I,J) = R(J,I)                                                 R0036
   31   CONTINUE                                                         R0037
   30 CONTINUE                                                          R0038
   50 RETURN                                                            R0039
      END                                                                 R0040
```

```
      SUBROUTINE CLOG(W,WBAR)                                          C0001
      COMMON/SLVE/IV(30),TCL(20),NR(55,2),B(55),KN(120),X(121),C(121),  C0002
     1  KL(26),NAM(25,2),A(55,121),PIE(65),V1(65),V2(65),V3(65),        C0003
     2  V4(65),XMF(120),X1(121),X2(121),X3(121),XBAR(25),R(65,65)       C0004
      EQUIVALENCE (IV(1),M),(IV(2),MEND),(IV(3),NCOMP),(IV(4),N,NTOT),  C0005
     1 (IV(5),NIT),(IV(6),NOT),(IV(7),PF),(IV(8),ITER),(IV(9),ITMAX),   C0006
     2  (IV(10),IERROR),(IV(11),LASTCP),(IV(12),KE)                     C0007
      DIMENSION  W(1),WBAR(1),ALPHA(1)                                 C0008
      EQUIVALENCE (X2,ALPHA)                                           C0009
      DO 1 K = 1, NCOMP                                                C0010
        KLA = KL(K)                                                    C0011
        KLB = KL(K+1)-1                                                C0012
        DO 2 J = KLA,KLB                                               C0013
          ALPHA(J) = C(J)                                              C0014
          XXX = W(J)/WBAR(K)                                           C0015
          IF(XXX.GT.0.0) ALPHA(J) = C(J)+ALOG(XXX)                     C0016
2     CONTINUE                                                         C0017
1     CONTINUE                                                         C0018
      RETURN                                                           C0019
      END                                                              C0020
```

```
      SUBROUTINE LP  (MON)                                                    L0001
      COMMON/SLVE/IV(30),TOL(20),NR(55,2),B(55),KN(120),X(121),C(121),        L0002
     1  KL(26),NAM(25,2),A(55,121),PIE(65),V1(65),V2(65),V3(65),              L0003
     2  V4(65),XMF(120),X1(121),X2(121),X3(121),XBAR(25),R(65,65)             L0004
      INTEGER PF                                                              L0005
      EQUIVALENCE  (TOL(3),XMIN),(TOL(4),XSTART),(TOL(5),BARMIN)              L0006
      EQUIVALENCE (IV(1),M),(IV(2),MEND),(IV(3),NCOMP),(IV(4),N,NTOT),        L0007
     1 (IV(5),NIT),(IV(6),NOT),(IV(7),PF),(IV(8),ITER),(IV(9),ITMAX),         L0008
     2  (IV(10),IERROR),(IV(11),LASTCP),(IV(12),KE)                          L0009
      DIMENSION    XX(1),KOUT(7),CC(1),P(1)                                   L0010
      EQUIVALENCE(CC,XMF),(XX,X2),(P,V1)                                      L0011
      MON= 0                                                                  L0012
      IF (XSTART.LE.0.0) XSTART = 1.E-6                                       L0013
      DO 10 I = 1,M                                                           L0014
        P(I) = B(I)                                                           L0015
      A(I,NTOT+1) = 0.0                                                       L0016
        DO 15 J = 1,NTOT                                                      L0017
          A(I,NTOT+1) = A(I,NTOT+1) + A(I,J)                                  L0018
   15   CONTINUE                                                             L0019
   10 CONTINUE                                                             L0020
      DO 1  J = 1,NTOT                                                        L0021
        CC(J) = 0.0                                                           L0022
    1 CONTINUE                                                                L0023
      CC(N+1) = -1.0                                                          L0024
C  ZERO-TH SIMPLEX IS TO DETERMINE FEASIBILITY                                L0025
      CALL SIMPLE(0,M,N+1,A,P,CC,KOUT,XX,PIE,V2,V3,V4,X3,R)                   L0026
      ZT = XX(N+1)                                                            L0027
      IF(PF.GE.0)WRITE (NOT,106)KOUT(2),ZT,KOUT(1)                            L0028
  106 FORMAT(12HOSIMPLEX  0,I4,29H ITERATIONS, MAX MIN ELEMENT=1PE15.8,       L0029
     1 12H, CONDITION ,I3)                                                    L0030
      ZZT   =AMIN1(ZT/2.0, XSTART)                                            L0031
      DO 104 I = 1,M                                                          L0032
        P(I) = P(I) - ZZT*A(I,N+1)                                            L0033
  104 CONTINUE                                                                L0034
  200 DO 201  J = 1,NTOT                                                      L0035
        X(J) = XX(J)                                                          L0036
        XMF(J) = 1.0                                                          L0037
  201 CONTINUE                                                                L0038
      IF (ZT.LE.0..OR.KOUT(1).NE.0)  GO TO 40                                 L0039
C   SIMPLEX LOOP                                                              L0040
      FR2=1.E+20                                                              L0041
      DO 301 NN = 1, NCOMP                                                    L0042
        DO 302 J = 1, NTOT                                                    L0043
          CC(J)    = C(J) + XMF(J) - 1.0                                      L0044
  302   CONTINUE                                                            L0045
      FN = FLOAT(NN) - 1.0                                                    L0046
      CALL SIMPLE(1,M,N  ,A,P,CC,KOUT,XX,PIE,V2,V3,V4,X3,R)                   L0047
        IF (KOUT(1).NE.0) GO TO 50                                            L0048
  300   DO 303 J = 1,NTOT                                                     L0049
        X(J) = XX(J)                                                          L0050
      X(J) = ( FN*X1(J) + X(J) ) / (FN + 1.0)                                 L0051
          X1(J) = X(J)                                                        L0052
  303   CONTINUE                                                            L0053
        CALL BAR(X,XBAR)                                                      L0054
        K = 1                                                                 L0055
      FR = 0.0                                                                L0056
        DO 310  J = 1,N                                                       L0057
          IF (J.GE.KL(K+1))  K = K + 1                                        L0058
          IF (J.EQ.KL(K).AND.XBAR(K).GT.0.0)FR=FR-XBAR(K)*ALOG(XBAR(K))       L0059
            IF (X(J).GT.0.0)  FR = FR + X(J)*(ALOG(X(J)) + C(J) )             L0060
```

```
        XMF(J) = 0.                                                          L0061
        IF ( XBAR(K).NE.0.) XMF(J) = X(J) / XBAR(K)                          L0062
310     CONTINUE                                                             L0063
        IF (PF.GE.0) WRITE(NOT,305) NN,KOUT(2),FR                            L0064
305     FORMAT(8H SIMPLEX,I3,1H,,I4,12H ITERATIONS ,8H FR ENG=1PE15.6)       L0065
        IF (FR.GE.FR2) GO TO 399                                             L0066
        FR2=FR                                                               L0067
301   CONTINUE                                                               L0068
399   DO 400 J = 1,N                                                         L0069
        X(J) = X(J) + ZZT                                                    L0070
400   CONTINUE                                                               L0071
      RETURN                                                                 L0072
 40   IF (KOUT(1).GT.1)  GO TO 50                                            L0073
      WRITE (NOT,41)                                                         L0074
 41   FORMAT(72HOTHIS PROBLEM IS INFEASIBLE.  THE FOLLOWING LINEAR COMBI     L0075
     1NATION OF ROWS, /1X)                                                   L0076
      DO 140 I =1,M                                                          L0077
        IF (PIE(I).NE.0.)  WRITE(NOT,141) PIE(I),NR(I,1),NR(I,2)             L0078
141     FORMAT(10X,3H+ (,F15.8,5H ) * ,2A6)                                  L0079
140   CONTINUE                                                               L0080
      WRITE (NOT,142)                                                        L0081
142   FORMAT(48HO  LEADS TO THE FOLLOWING INFEASIBLE EQUATION,  /1X)         L0082
      DO 150 K =1,NCOMP                                                      L0083
        MTA = KL(K)                                                          L0084
        MTB = KL(K+1) - 1                                                    L0085
        DO 151 J = MTA, MTB                                                  L0086
          D = 0.                                                             L0087
          DO 152 I =1,M                                                      L0088
            D = PIE(I)* A(I,J) + D                                           L0089
152       CONTINUE                                                           L0090
          IF (D.NE.0.) WRITE (NOT,143) D,KN(J),NAM(K,1),NAM(K,2)             L0091
143   FORMAT(10X,3H+ (,F15.8,5H ) * ,A6,4H IN ,2A6)                          L0092
151     CONTINUE                                                             L0093
150   CONTINUE                                                               L0094
      D = 0.                                                                 L0095
      DO 160 I =1,M                                                          L0096
        D = PIE(I)*B(I) + D                                                  L0097
160   CONTINUE                                                               L0098
      WRITE (NOT,144) D                                                      L0099
144   FORMAT(1H0,15X, 7H+ 0.0 =,F15.8)                                       L0100
 70   MON = 1                                                                L0101
      RETURN                                                                 L0102
 50   IF (KOUT(1).NE.2)  GO TO 60                                            L0103
      JT = KOUT(7)                                                           L0104
      DO 51  K = 1,NCOMP                                                     L0105
        IF ( JT.GE.KL(K))  GO TO 52                                          L0106
 51   CONTINUE                                                               L0107
 52   WRITE (NOT,952)  KN(JT),NAM(K,1),NAM(K,2)                              L0108
952   FORMAT(14H THE VARIABLE ,A6,4H IN ,2A6,33H IS UNBOUNDED AND MUST B     L0109
     1E REMOVED)                                                             L0110
      GO TO 70                                                               L0111
 60   WRITE (NOT,960)                                                        L0112
960   FORMAT(60H SIMPLEX ROUTINE HAS FAILED DUE TO EXCESSIVE ROUND-OFF E     L0113
     1RROR)                                                                  L0114
      GO TO 70                                                               L0115
      END                                                                    L0116
```

Calling Sequence for Simplex Subroutine

The simplex subroutine, SIMPLE, may be used to solve a general linear programming problem of the form: Minimize

$$\sum_{j=1}^{n} C_j x_j \qquad (1)$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \ . \qquad i=1,2,3,\ldots,m \qquad (2)$$

The $a_{ij}$ is stored in a two-dimensional array, A, with $a_{ij}$ in cell A(i,j); $C_j$ is stored in a one-dimensional array, C, with $C_j$ in cell C(j); and $b_i$ is stored in a one-dimensional array, B, with $b_i$ in cell B(i).

The calling sequence is

```
CALL SIMPLE(II,M,N,A,B,C,KO,X,P,JH,XX,Y,PE,E)
```

where

II = 0;

M = No. of rows, m;

N = No. of variables, n;

A, B, C Are as above;

KO = A subscripted variable of dimension 7;

X = A subscripted variable of dimension n or more;

P, JH, XX, Y, and PE = Subscripted variables of dimension m or more; and

E = A subscripted variable of dimension $m^2$ or more.

Upon exiting from the subroutine,

X(1),X(2),...,X(n) Contains $x_1, x_2, \ldots, x_n$ (the solution);

P(1),P(2),...,P(m) Contains the shadow prices;

KO(1) Contains an 0 if the problem was feasible, 1 if the problem was infeasible, 2 if the problem had an infinite solution, and 3, 4, or 5 if the algorithm did not terminate;

KO(2) The number of iterations taken;

KO(3) The number of pivots performed since the last inversion;

KO(4) The number of inversions performed;

KO(5) The number of pivot steps performed;

KO(6) A logical variable that is "true" if and only if the problem was feasible; and

KO(7) Contains, if the problem had an infinite solution, the number of the variable that was infinite.

The dimension of A (line X0009) must agree (at least in the first subscript) with the dimension of A in the calling program. The other dimensions need not agree with those of the calling program.

If an initial basis is available, this basis may be communicated to the subroutine by letting

$$II = 1 ,$$

$$X(i) = \begin{cases} 0.0 \text{ if variable i is not in basis,} \\ \text{(non-zero) if variable i is in basis,} \end{cases}$$

and the other quantities remain as above.

This subroutine differs from other linear programming routines in several respects. If the restraints (2) are linearly dependent, the problem is considered to be infeasible. This is the case because the chemical equilibrium problem cannot be solved if the restraints are dependent. In addition, this subroutine was written to be as scale-free

as possible; this was accomplished by computing tolerances internally in the subroutine.

```
C  AUTOMATIC SIMPLEX       REDUNDANT EQUATIONS CAUSE INFEASIBILITY          X0001
      SUBROUTINE SIMPLE(INFLAG,MX,NN,A,B,C,KOUT,KB,P,JH,X,Y,PE,E)           X0002
      DIMENSION           B(1),C(1),KOUT(7),JH(1),X(1),P(1),Y(1),           X0003
     1  KB(1),E(1),PE(1),KO(7)                                             X0004
      EQUIVALENCE    (K,KO(1)),(ITER,KO(2)),(INVC,KO(3)),                  X0005
     2  (NUMVR,KO(4)),(NUMPV,KO(5)),(FEAS,KO(6)),(JT,KO(7))                X0006
      EQUIVALENCE (XX,LL)                                                  X0007
C THE FOLLOWING DIMENSION SHOULD BE THE SAME HERE AS IT IS IN CALLER.       X0008
      DIMENSION A(55,121)                                                  X0009
      LOGICAL FEAS,VER,NEG,TRIG,KG,ABSC                                    X0010
C                                                                          X0011
C                          MOVE INPUTS  ...  ZERO OUTPUTS                  X0012
      DO 1341  I = 1,7                                                     X0013
        KO(I) = 0                                                          X0014
 1341 CONTINUE                                                             X0015
      M = MX                                                               X0016
      N = NN                                                               X0017
      TEXP  =  .5**16                                                      X0018
      NCUT  =  4*M + 10                                                    X0019
      NVER  =  M/2  +  5                                                   X0020
      M2 = M**2                                                            X0021
      IF  (INFLAG.NE.0)  GO TO 1400                                        X0022
C* 'NEW'    START  PHASE ONE WITH SINGLETON BASIS                          X0023
      DO 1402  J = 1,N                                                     X0024
        KB(J) = 0                                                          X0025
        KG = .FALSE.                                                       X0026
        DO 1403  I = 1,M                                                   X0027
          IF (A(I,J).EQ.0.0)  GO TO 1403                                   X0028
          IF (KG.OR.A(I,J).LT.0.0) GO TO 1402                              X0029
          KG = .TRUE.                                                      X0030
 1403   CONTINUE                                                           X0031
        KB(J) = 1                                                          X0032
 1402 CONTINUE                                                             X0033
 1400 IF (INFLAG.GT.1 )  GO TO 1320                                        X0034
      DO 1401  I =1,M                                                      X0035
        JH (I) = -1                                                        X0036
 1401 CONTINUE                                                             X0037
C* 'VER'    CREATE INVERSE FROM 'KB' AND 'JH'                              X0038
 1320 VER = .TRUE.                                                         X0039
 1121 INVC = 0                                                             X0040
 1122 NUMVR   =  NUMVR  +1                                                 X0041
      DO 1101  I = 1,M2                                                    X0042
        E(I) = 0.0                                                         X0043
 1101 CONTINUE                                                             X0044
      MM=1                                                                 X0045
      DO 1113  I = 1,M                                                     X0046
        E(MM) = 1.0                                                        X0047
        PE(I)  =  0.0                                                      X0048
        X(I) =  B(I)                                                       X0049
        IF (JH(I) .NE.0) JH(I) = -1                                        X0050
        MM = MM + M + 1                                                    X0051
 1113 CONTINUE                                                             X0052
C                  FORM INVERSE                                            X0053
      DO 1102  JT = 1,N                                                    X0054
        IF (KB(JT).EQ.0)  GO TO 1102                                       X0055
        GO TO 600                                                          X0056
C 600   CALL JAY                                                          X0057
C                         CHOOSE PIVOT                                     X0058
 1114   TY = 0.0                                                          X0059
        DO 1104 I = 1,M                                                    X0060
```

```
          IF (JH(I).NE.-1)  GO TO 1104                                  X0061
          IF ( ABS(Y(I)).LE.TY)  GO TO 1104                             X0062
          IR = I                                                        X0063
          TY = ABS(Y(I))                                                X0064
 1104   CONTINUE                                                        X0065
        KB(JT) = 0                                                      X0066
C                          TEST PIVOT                                   X0067
        IF (TY.LE.TPIV)  GO TO 1102                                     X0068
C                          PIVOT                                        X0069
        JH(IR) = JT                                                     X0070
        KB(JT) = IR                                                     X0071
        GO TO 900                                                       X0072
C 900   CALL PIV                                                        X0073
 1102 CONTINUE                                                          X0074
C                    RESET ARTIFICIALS                                  X0075
      DO 1109  I = 1,M                                                  X0076
        IF (JH(I).EQ.-1)  JH(I) = 0                                     X0077
 1109 CONTINUE                                                          X0078
 1200 VER = .FALSE.                                                     X0079
C                                  PERFORM ONE ITERATION                X0080
C* 'XCK'    DETERMINE FEASIBILITY                                       X0081
      FEAS= .TRUE.                                                      X0082
      NEG = .FALSE.                                                     X0083
      DO 1201  I = 1,M                                                  X0084
        IF (X(I).LT.0.0)  GO TO 1250                                    X0085
        IF (JH(I).EQ.0)  FEAS = .FALSE.                                 X0086
 1201 CONTINUE                                                          X0087
C* 'GET'    GET APPLICABLE PRICES                                       X0088
      IF (.NOT.FEAS)  GO TO 501                                         X0089
C                          PRIMAL PRICES                                X0090
      DO 503 I = 1,M                                                    X0091
        P(I) = PE(I)                                                    X0092
  503 CONTINUE                                                          X0093
      ABSC = .FALSE.                                                    X0094
      GO TO 599                                                         X0095
C                          COMPOSITE PRICES                             X0096
 1250 FEAS = .FALSE.                                                    X0097
      NEG  = .TRUE.                                                     X0098
  501 DO 504  J = 1, M                                                  X0099
        P(J) = 0.                                                       X0100
  504 CONTINUE                                                          X0101
      ABSC = .TRUE.                                                     X0102
      DO 505  I = 1,M                                                   X0103
        MM = I                                                          X0104
        IF (X(I).GE.0.0)  GO TO 507                                     X0105
        ABSC = .FALSE.                                                  X0106
        DO 508 J = 1,M                                                  X0107
          P(J) = P(J) + E(MM)                                           X0108
          MM = MM + M                                                   X0109
  508   CONTINUE                                                        X0110
        GO TO 505                                                       X0111
  507   IF (JH(I).NE.0) GO TO 505                                       X0112
        IF (X(I).NE.0.) ABSC = .FALSE.                                  X0113
        DO 510  J = 1,M                                                 X0114
          P(J) = P(J) - E(MM)                                           X0115
          MM = MM + M                                                   X0116
  510   CONTINUE                                                        X0117
  505 CONTINUE                                                          X0118
C* 'MIN'    FIND MINIMUM REDUCED COST                                   X0119
  599 JT = 0                                                            X0120
```

```
      BB  =  0.0                                                        X0121
      DO 701  J =1,N                                                    X0122
C                                SKIP COLUMNS IN BASIS                  X0123
        IF (KB(J).NE.0)   GO TO 701                                     X0124
        DT = 0.0                                                        X0125
        DO 303 I = 1,M                                                  X0126
          IF (A(I,J).NE.0.0)  DT = DT + P(I) * A(I,J)                   X0127
  303   CONTINUE                                                        X0128
        IF (FEAS)  DT = DT + C(J)                                       X0129
        IF  (ABSC)  DT = - ABS(DT)                                      X0130
        IF (DT.GE.BB)  GO TO 701                                        X0131
        BB = DT                                                         X0132
        JT = J                                                          X0133
  701 CONTINUE                                                          X0134
C  TEST FOR NO PIVOT COLUMN                                             X0135
      IF (JT.LE.0)  GO TO 203                                           X0136
C  TEST FOR ITERATION LIMIT EXCEEDED                                    X0137
      IF (ITER.GE.NCUT)  GO TO 160                                      X0138
      ITER = ITER +1                                                    X0139
C* 'JMY'    MULTIPLY INVERSE TIMES A(.,JT)                             X0140
  600 DO 610  I= 1,M                                                    X0141
        Y(I) = 0.0                                                      X0142
  610 CONTINUE                                                          X0143
      LL = 0                                                            X0144
      COST = C(JT)                                                      X0145
      DO 605   I= 1,M                                                   X0146
        AIJT = A(I,JT)                                                  X0147
        IF (AIJT.EQ.0.) GO TO 602                                       X0148
        COST = COST + AIJT * PE(I)                                      X0149
        DO  606  J = 1,M                                                X0150
          LL = LL + 1                                                   X0151
          Y(J) = Y(J) + AIJT * E(LL)                                    X0152
  606   CONTINUE                                                        X0153
        GO TO 605                                                       X0154
  602   LL = LL + M                                                     X0155
  605 CONTINUE                                                          X0156
C         COMPUTE PIVOT TOLERANCE                                       X0157
      YMAX = 0.0                                                        X0158
      DO 620  I  =  1,M                                                 X0159
        YMAX = AMAX1( ABS(Y(I)),YMAX )                                  X0160
  620 CONTINUE                                                          X0161
      TPIV  =  YMAX * TEXP                                              X0162
C            RETURN TO INVERSION ROUTINE, IF INVERTING                  X0163
      IF (VLR)  GO TO 1114                                              X0164
C       COST TOLERANCE CONTROL                                          X0165
      IF (TRIG.AND.BB.GE.-TPIV)  GO TO 203                              X0166
      TRIG = .FALSE.                                                    X0167
      IF (BB.GE.-TPIV)  TRIG = .TRUE.                                   X0168
C* 'ROW'    SELECT PIVOT ROW                                           X0169
C AMONG EQS. WITH X=0, FIND MAXIMUM Y  AMONG ARTIFICIALS, OR, IF NONE,  X0170
C  GET MAX POSITIVE Y(I) AMONG REALS.                                   X0171
 1000 IR = 0                                                            X0172
      AA = 0.0                                                          X0173
      KQ = .FALSE.                                                      X0174
      DO  1050  I =1,M                                                  X0175
        IF (X(I).NE.0.0.OR.Y(I).LE.TPIV)  GO TO 1050                    X0176
        IF (JH(I).EQ.0)  GO TO 1044                                     X0177
        IF (KQ) GO TO 1050                                              X0178
 1045   IF (Y(I).LE.AA)  GO TO 1050                                     X0179
        GO TO 1047                                                      X0180
```

```
 1044    IF (KQ) GO TO 1045                                                    X0181
         KQ = .TRUE.                                                           X0182
 1047    AA = Y(I)                                                             X0183
         IR = I                                                                X0184
 1050 CONTINUE                                                                 X0185
      IF (IR.NE.0)  GO TO 1099                                                 X0186
 1001 AA = 1.0E+20                                                             X0187
C                FIND MIN. PIVOT AMONG POSITIVE EQUATIONS                     X0188
      DO 1010  I  = 1,M                                                        X0189
        IF (Y(I).LE.TPIV.OR.X(I).LE.0.0.OR.Y(I)*AA.LE.X(I) ) GO TO 1010        X0190
        AA = X(I)/Y(I)                                                         X0191
        IR = I                                                                 X0192
 1010 CONTINUE                                                                 X0193
      IF (.NOT.NEG)  GO TO 1099                                                X0194
C  FIND PIVOT AMONG NEGATIVE EQUATIONS, IN WHICH X/Y IS LESS THAN THE          X0195
C MINIMUM X/Y IN THE POSITIVE EQUATIONS, THAT HAS THE LARGEST ABSF(Y)          X0196
 1016 BB = - TPIV                                                              X0197
      DO 1030  I = 1,M                                                         X0198
        IF (X(I).GE.0..OR.Y(I).GE.BB.OR.Y(I)*AA.GT.X(I) )  GO TO 1030          X0199
        BB = Y(I)                                                              X0200
        IR = I                                                                 X0201
 1030 CONTINUE                                                                 X0202
C  TEST FOR NO PIVOT ROW                                                       X0203
 1099 IF  (IR.LE.0)  GO TO 207                                                 X0204
C* 'PIV'    PIVOT ON (IR,JT)                                                   X0205
C                          LEAVE TRANSFORMED COLUMN IN Y(I)                   X0206
  900 NUMPV  =  NUMPV  +  1                                                    X0207
      YI = -Y(IR)                                                              X0208
      Y(IR) = -1.0                                                             X0209
      LL = 0                                                                   X0210
C                                  TRANSFORM INVERSE                          X0211
      DO  904  J = 1,M                                                         X0212
        L = LL + IR                                                            X0213
        IF (E(L).NE.0.0)  GO TO 905                                            X0214
        LL = LL + M                                                            X0215
        GO TO 904                                                              X0216
  905   XY = E(L) / YI                                                         X0217
        PE(J) = PE(J) + COST * XY                                              X0218
        E(L) = 0.0                                                             X0219
        DO 906  I = 1,M                                                        X0220
          LL  =  LL + 1                                                        X0221
          E(LL) = E(LL)  + XY * Y(I)                                           X0222
  906   CONTINUE                                                               X0223
  904 CONTINUE                                                                 X0224
C                                  TRANSFORM X                                X0225
      XY  =  X(IR) / YI                                                        X0226
      DO 908  I  = 1, M                                                        X0227
        XNEW = X(I) + XY * Y(I)                                                X0228
        IF (VER.OR.XNEW.GE.0..OR.Y(I).GT.TPIV.OR.X(I).LT.0.) GO TO 907         X0229
        X(I) = 0.0                                                             X0230
        GO TO 908                                                              X0231
  907   X(I) = XNEW                                                            X0232
  908 CONTINUE                                                                 X0233
C                          RESTORE Y(IR)                                      X0234
      Y(IR) = -YI                                                              X0235
      X(IR) = -XY                                                              X0236
      IF (VER) GO TO 1102                                                      X0237
  221 IA = JH(IR)                                                              X0238
      IF (IA.GT.0)  KB(IA) = 0                                                 X0239
  213 KB(JT)   = IR                                                            X0240
```

```
      JH(IR)    = JT                                                    X0241
      IF (NUMPV.LE.M) GO TO 1200                                        X0242
C  TEST FOR INVERSION ON THIS ITERATION                                  X0243
      INVC    = INVC    +1                                            X0244
      IF  (INVC.EQ.NVER)    GO TO 1320                                  X0245
      GO  TO  1200                                                      X0246
C*  END OF ALGORITHM, SET EXIT VALUES                                   X0247
C             INFINITE SOLUTION                                         X0248
  207 K   =  2                                                          X0249
      GO TO 250                                                         X0250
C          PROBLEM IS CYCLING                                           X0251
  160 K   =  4                                                          X0252
      GO TO 250                                                         X0253
C          FEASIBLE OR INFEASIBLE SOLUTION                              X0254
  203 K   =  0                                                          X0255
  250 IF (.NOT.FEAS)  K = K + 1                                         X0256
      DO 1399  J = 1,N                                                  X0257
        XX   =  0.0                                                     X0258
        KBJ   =  KB(J)                                                  X0259
        IF (KBJ.NE.0)  XX = X(KBJ)                                      X0260
        KB(J) = LL                                                      X0261
 1399 CONTINUE                                                          X0262
C       SET 'KOUT'                                                      X0263
 1392 DO 1393 I = 1,7                                                   X0264
        KOUT(I) = KB(I)                                                 X0265
 1393 CONTINUE                                                          X0266
      RETURN                                                            X0267
       END                                                              X0268
```

```
C       MATRIX INVERSION WITH ACCOMPANYING SOLUTION OF LINEAR EQUATIONS      M0001
        SUBROUTINE MATINV(A,N,B,M,INA,INB,IP,ISING)                          M0002
C                                                                            M0003
        DIMENSION B(1),INA(1),INB(1),IP(1)                                   M0004
        LOGICAL IP                                                           M0005
        DIMENSION A(65,65)                                                   M0006
C                                                                            M0007
C       INITIALIZATION                                                       M0008
        DO 20 J = 1,N                                                        M0009
          IP(J) = .FALSE.                                                    M0010
   20   CONTINUE                                                             M0011
C BIG LOOP ON I                                                              M0012
        DO 575  I = 1,N                                                      M0013
          AMAX  = 0.0                                                        M0014
C       SEARCH FOR PIVOT ELEMENT                                             M0015
          DO 105 J = 1,N                                                     M0016
            IF  (IP(J))  GO TO 105                                           M0017
            DO 100 K = 1,N                                                   M0018
              IF  (IP(K) .OR. ABS(AMAX).GE.ABS(A(J,K)) )  GO TO 100          M0019
              IROW  = J                                                      M0020
              ICOL = K                                                       M0021
              AMAX = A(J,K)                                                  M0022
  100       CONTINUE                                                         M0023
  105     CONTINUE                                                           M0024
          IF (AMAX.EQ.0.0)  GO TO 750                                        M0025
          IP(ICOL) = .TRUE.                                                  M0026
C       INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL                    M0027
          IF (IROW.EQ.ICOL) GO TO 260                                        M0028
          DO 200 L = 1,N                                                     M0029
            SWAP = A(IROW,L)                                                 M0030
            A(IROW,L) = A(ICOL,L)                                            M0031
            A(ICOL,L) = SWAP                                                 M0032
  200     CONTINUE                                                           M0033
          IF (M.EQ.0) GO TO 260                                              M0034
          SWAP = B(IROW)                                                     M0035
          B(IROW)   = B(ICOL)                                                M0036
          B(ICOL)   = SWAP                                                   M0037
  260     INA(I) = IROW                                                      M0038
          INB(I) = ICOL                                                      M0039
C       DIVIDE PIVOT ROW BY PIVOT ELEMENT                                    M0040
          A(ICOL,ICOL) = 1.0                                                 M0041
          DO 350  L = 1,N                                                    M0042
            A(ICOL,L) = A(ICOL,L) / AMAX                                     M0043
  350     CONTINUE                                                           M0044
          IF (M.NE.0)  B(ICOL) = B(ICOL) / AMAX                              M0045
C       COMPLETE THE PIVOT                                                   M0046
  380     DO 550 LL = 1,N                                                    M0047
            IF (LL.EQ.ICOL) GO TO 550                                        M0048
            SWAP = A(LL,ICOL)                                                M0049
            A(LL,ICOL) = 0.0                                                 M0050
            DO 450  L = 1,N                                                  M0051
              A(LL,L) = A(LL,L) - A(ICOL,L) * SWAP                           M0052
  450       CONTINUE                                                         M0053
            IF (M.NE.0)  B(LL)   = B(LL)   - B(ICOL) * SWAP                  M0054
  550     CONTINUE                                                           M0055
  575   CONTINUE                                                             M0056
  600   IF (M.LT.0) RETURN                                                   M0057
C       INTERCHANGE COLUMNS                                                  M0058
        DO 710  I = 1,N                                                      M0059
```

```
      L = N + 1 - I                                                    M0060
      IF (INA(L).EQ.INB(L))  GO TO 710                                 M0061
      IROW = INA(L)                                                    M0062
      ICOL = INB(L)                                                    M0063
      DO 705 K = 1,N                                                   M0064
        SWAP = A(K,IROW)                                               M0065
        A(K,IROW) = A(K,ICOL)                                          M0066
        A(K,ICOL) = SWAP                                               M0067
  705 CONTINUE                                                         M0068
  710 CONTINUE                                                         M0069
  740 RETURN                                                           M0070
C SINGULARITY FLAG                                                     M0071
  750 ISING = 1 + N - I                                                M0072
      GO TO 600                                                        M0073
      END                                                              M0074
```

## Appendix B

## MATRIX NOTATION AND FURTHER PROOFS

The derivations in the preceding sections would be facilitated by the use of matrix notation rather than subscripted variables. We introduce the following symbols to correspond to the subscripted variables used in Sec. 3.

| Subscripted Variable | Matrix | Size of Matrix |
|---|---|---|
| $a_{ij}$ | A | MxN |
| $b_i$ | B | Mx1 |
| $y_j$ | Y | Nx1 |
| $d_j$ | D | Nx1 |
| $c_j$ | C | Nx1 |
| $\pi_i$ | $\pi$ | Mx1 |
| $r_{i\ell}$ | R | MxM |
| $x_j$ | X | Nx1 |

The single-column matrices may also be thought of as vectors. We use here the convention that an operator applied to a matrix means that the operator operates on each element of the matrix. For example, log Y is the Nx1 matrix consisting of

$$\begin{pmatrix} \log y_1 \\ \log y_2 \\ \vdots \\ \log y_N \end{pmatrix}$$

The superscript $^{\tau}$ indicates the transposition of a matrix. We assume that the elementary results of matrix theory are known. For example, it is known that the inverse of an invertable symmetric matrix is symmetric. The square diagonal matrix whose diagonal is one of the vectors previously defined will be denoted by the previously defined vector in elongated type; that is,

$$\mathbb{D} = \text{diag } (D)$$

and

$$\mathbb{Y} = \text{diag } (Y)$$

Equations (3.2) and (3.7) in matrix notation are

$$AX = B \tag{B.1}$$

$$X = \mathbb{Y}(\mathbb{D}^{-1}A^{\tau}\pi - \mathbb{D}^{-1}C - \log Y) . \tag{B.2}$$

To see the ease of matrix notation, we may substitute (B.2) into (B.1) to get

$$A\Upsilon D^{-1}A^{\tau}\pi = B + A\Upsilon(D^{-1}C + \log Y) \ . \tag{B.3}$$

By letting

$$R = A\Upsilon D^{-1}A^{\tau} \tag{B.4}$$

and

$$S = B + A\Upsilon(D^{-1}C + \log Y) \ , \tag{B.5}$$

we see that

$$R\pi = S \tag{B.6}$$

corresponds to (3.10).

In Sec. 4, we evaluated

$$\sum_{j=1}^{N} \frac{\theta^2 d_{jj}}{y_j} \tag{B.7}$$

but we did not give the details of the computation. The algebra of this evaluation is very difficult unless matrix algebra is used. In matrix notation, (B.7) is $\theta^T D Y^{-1} \theta$, where $\theta = X - Y$. From (B.2) we have

$$\theta = Y(D^{-1}A^T\pi - D^{-1}C - \log Y) - Y \ . \qquad (B.8)$$

Hence,

$$\theta^T D Y^{-1} \theta = (\pi^T A D^{-1} - C^T D^{-1} - \log Y^T) Y D Y^{-1} \theta - Y^T D Y^{-1} \theta$$

$$= \pi^T A (D^{-1} Y D Y^{-1}) \theta - (C^T D^{-1} + \log Y^T) D Y Y^{-1} \theta - Y^T Y^{-1} D \theta$$

$$= \pi^T A \theta - (C^T D^{-1} + \log Y^T) D \theta - D^T \theta \ . \qquad (B.9)$$

Since $AX = B$, $A\theta = AX - AY = B - AY$. Also, in the chemical equilibrium formulation,

$$D^T \theta = \sum_{j=1}^{n} \theta_j - \sum_{j=n+1}^{N} \theta_j = \sum_{k=1}^{p} \left( \sum_{j \in \langle k \rangle} \theta_j - \theta_{k+m} \right) = 0$$

and

$$(C^T\mathbb{D})^{-1} + \log Y^T)\mathbb{D}\theta$$

$$= \sum_{j=1}^{n} (c_j + \log y_j)\theta_j + \sum_{j=n+1}^{N} \log y_j(-\theta_j)$$

$$= \sum_{k=1}^{p} \left( \sum_{j \in \langle k \rangle} \theta_j(c_j + \log y_j) - \theta_k \log S_k \right)$$

$$= \sum_{k=1}^{p} \left( \sum_{j \in \langle k \rangle} \theta_j(c_j + \log y_j - \log S_k) \right)$$

$$= \sum_{j=1}^{n} \theta_j(c_j + \log \hat{y}_j) \ .$$

Hence,

$$\sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j} = \sum_{i=1}^{m} \pi_i \left( b_i - \sum_{j=1}^{n} a_{ij} y_j \right) - \sum_{j=1}^{n} \theta_j(c_j + \log \hat{y}_j) \quad \text{(B.10)}$$

in the context of the chemical equilibrium problem used in Sec. 4.

Next we wish to show that

$$\sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j} \geq 0$$

as stated in (4.14). First, we prove

Lemma 1: Let $y_1, y_2, \ldots, y_r$ be positive numbers and let $\theta_1, \theta_2, \ldots, \theta_r$ be any real numbers. Let

$$G = \sum_{j=1}^{r} \frac{\theta_j^2}{y_j} - \frac{\left(\sum_{j=1}^{r} \theta_j\right)^2}{\sum_{j=1}^{r} y_j} .$$

Then,

i) $G \geq 0$

ii) $G = 0$ if and only if

$$\frac{\theta_1}{y_1} = \frac{\theta_2}{y_2} = \ldots = \frac{\theta_r}{y_r} .$$

Proof: Let $\alpha_j = \theta_j / y_j$, $j=1,2,\ldots,r$. Then,

$$G = \sum_{j=1}^{r} \alpha_j^2 y_j - \frac{\left(\sum_{j=1}^{r} \alpha_j y_j\right)^2}{\sum_{j=1}^{r} y_j}$$

$$= \left( \sum_{j=1}^{r} y_j \right)^{-1} \left[ \left( \sum_{j=1}^{r} y_j \right) \left( \sum_{j=1}^{r} \alpha_j^2 y_j \right) - \left( \sum_{j=1}^{r} \alpha_j y_j \right)^2 \right]$$

$$= \left( \sum_{j=1}^{r} y_j \right)^{-1} \left[ \sum_{i=1}^{r} \left( \sum_{j=1}^{r} \left( \alpha_j^2 y_i y_j - \alpha_i \alpha_j y_i y_j \right) \right) \right]$$

$$= \left( \sum_{j=1}^{r} y_j \right)^{-1} \left[ \sum_{i=1}^{r} \left( \sum_{j=1}^{i} \left( \alpha_j^2 y_i y_j - 2\alpha_i \alpha_j y_i y_j + \alpha_i^2 y_i y_j \right) \right) \right]$$

$$= \left( \sum_{j=1}^{r} y_j \right)^{-1} \left( \sum_{j<i} y_i y_j (\alpha_j - \alpha_i)^2 \right) \geq 0 \ ,$$

which is result i). The proof is completed by noting that $G = 0$ if and only if $\alpha_i = \alpha_j$ for all i and j; this proves ii).

Now we can prove

Theorem 1: In the chemical equilibrium problem

i) $\sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j} \geq 0$

ii) $\sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j} = 0$ if and only if there exist

numbers $\alpha_1, \alpha_2, \ldots, \alpha_p$ such that

$$\text{a)} \quad \theta_j = \alpha_{[j]} y_j \qquad j \leq n$$

$$\text{b)} \quad \theta_j = \alpha_{j-n} S_{j-n} \,. \qquad j > n$$

Proof: The proof follows by noting that for $i > n$

$$\theta_i = \sum_{j \in \langle i-n \rangle} \theta_j \,.$$

Then,

$$\sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j} = \sum_{j=1}^{n} \frac{\theta_j^2}{y_j} - \sum_{k=1}^{p} \frac{\theta_{k+n}^2}{S_k}$$

$$= \sum_{k=1}^{p} \left( \sum_{j \in \langle k \rangle} \frac{\theta_j^2}{y_j} - \frac{\left( \sum_{j \in \langle k \rangle} \theta_j \right)^2}{\sum_{j \in \langle k \rangle} y_j} \right) \geq 0$$

by lemma 1. Furthermore, by lemma 1, if the equality holds, then for each k there is a number $\alpha_k$ such that $\theta_j = \alpha_k y_j$ if $j \in k$. This, noting that b) follows from the fact that

$$\theta_i = \sum_{j \in \langle i-n \rangle} \theta_j \quad \text{for} \quad i>n \ ,$$

completes the proof of the theorem.

Our final result is

Theorem 2: In the chemical equilibrium problem, with $(y_1,y_2,\ldots,y_n)$ feasible and $\theta_1,\theta_2,\ldots,\theta_n$ calculated as in (4.7)

$$\text{i)} \quad \sum_{j=1}^{n} \theta_j(c_j + \log \hat{y}_j) \leq 0$$

$$\text{ii)} \quad \sum_{j=1}^{n} \theta_j(c_j + \log \hat{y}_j) = 0 \quad \text{if and only if}$$

$(y_1,y_2,\ldots,y_n)$ is optimal.

Proof: i) follows from Theorem 1, (B.10), and the fact that $(y_1,y_2,\ldots,y_n)$ is feasible.

To prove ii), we assume that

$$\sum_{j=1}^{n} \theta_j(c_j + \log \hat{y}_j) = 0 \ .$$

Then,

$$\sum_{j=1}^{N} \frac{\theta_j^2 d_j}{y_j} = 0 ,$$

and $\theta_j$ is as in ii) of Theorem 1. Combining b) of Theorem 1 and (4.12) we have

$$\theta_{k+n} = S_k \pi'_{m+k} = \alpha_k S_k$$

or

$$\alpha_k = \pi'_{m+k} .$$

Next, we combine a) of Theorem 1 with (4.7) to get

$$\theta_j = y_j \left[ \sum_{i=1}^{m} \pi'_i a_{ij} - c_j - \log \hat{y}_j + \pi'_{[j]+m} \right]$$

$$= y_j \alpha_{[j]} = y_j \pi'_{[j]+m}$$

or

$$\sum_{i=1}^{m} \pi'_i a_{ij} - c_j - \log \hat{y}_j = 0 .$$

This last result is the optimality condition for $(y_1, y_2, \ldots, y_n)$ as given by (1.4), and this demonstrates the forward implication of ii). The converse follows from the fact that optimality implies that the objective function cannot be decreased.

## REFERENCES


1. Shapiro, N. Z., and L. S. Shapley, Mass Action Laws and the Gibbs Free Energy Function, The RAND Corporation, RM-3935-1-PR, September 1964.

2. Dantzig, G. B., Linear Programming and Extensions, The RAND Corporation, R-366-PR, August 1963. Also published by Princeton University Press, Princeton, New Jersey, 1963.

3. Kaplan, Wilfred, Advanced Calculus, Addison-Wesley Press, Inc., Cambridge, Massachusetts, 1952.

4. Clasen, R. J., The Linear-Logarithmic Programming Problem, The RAND Corporation, RM-3707-PR, June 1963.

5. White, W. B., S. M. Johnson, and G. B. Dantzig, Chemical Equilibrium in Complex Mixtures, The RAND Corporation, P-1059, October 8, 1957. Also published in J. Chem. Phys., 28 (1958) 751-755.

6. International Business Machines Corporation, "IBM 7040/7044 Operating System, FORTRAN IV Language," IBM Systems Reference Library, Form C28-6329, Poughkeepsie, New York, 1963.

7. Dantzig, G. B., and J. C. DeHaven, "On the Reduction of Certain Multiplicative Chemical Equilibrium Systems to Mathematically Equivalent Additive Systems," J. Chem. Phys., 36 (1962) 2620-2627.

8. Shapiro, N. Z., A Generalized Technique for Eliminating Species in Complex Chemical Equilibrium Calculations, The RAND Corporation, RM-4205-PR, August 1964.

9. Dantzig, G. B., J. C. DeHaven, I. Cooper, S. M. Johnson, E. C. DeLand, H. E. Kanter, and C. F. Sams, "A Mathematical Model of the Human External Respiratory System," Perspectives in Biol. & Med., 4 (1961) 324-376.

10. Maloney, J. V., Jr., M.D., J. C. DeHaven, E. C. DeLand, and G. B. Bradham, M.D., Analysis of Chemical Constituents of Blood by Digital Computer, The RAND Corporation, RM-3541-PR, April 1963.

11. DeHaven, J. C., and E. C. DeLand, Reactions of Hemoglobin and Steady States in the Human Respiratory System: An Investigation Using Mathematical Models and an Electronic Computer, The RAND Corporation, RM-3212-PR, December 1962.

12. DeHaven, J. C., E. C. DeLand, N. S. Assali, and W. Manson, Physiochemical Characteristics of Placental Transfer, The RAND Corporation, P-2565, March 1962.

13. Warga, J., "A Convergent Procedure for Solving the Thermo-Chemical Equilibrium Problem," J. Soc. Indust. Appl. Math., 11 (1963) 594-606.