AN ABSTRACT OF THE THESIS OF

Scott A. Sinquefield for the degree of Master of Science in Chemical Engineering presented on May 22, 1991.

Title: A Microcomputer Software Package for Simulation of Non-Ideal Aqueous Electrolyte Systems at Equilibrium

Abstract approved: Redacted for Privacy

William J. Frederick

The non-ideal aqueous electrolyte simulator (NAELS) is composed of three major parts: a Newton-Raphson non-linear optimization program written by Weare, et al (1987); an activity coefficient subroutine for non-ideal electrolyte systems based on Pitzer's model; and an extensive, user expandable database. It is robust, stable, and requires neither thermodynamic data nor initial guesses as input. NAELS provides very good estimates of equilibrium speciation and solubility in concentrated electrolyte systems. NAELS was assembled as a technical utility package for use on IBM-compatable microcomputers.

A Microcomputer Software Package for Simulation of Non-Ideal Aqueous Electrolyte Systems at Equilibrium

by

Scott A. Sinquefield

A THESIS
submitted to
Oregon State University

in partial fulfillment of
the requirements for the
degree of
Master of Science

Completed May 22, 1991
Commencement June 1992

APPROVED:

Redacted for Privacy

Associate Professor of Chemical Engineering in charge of major

Redacted for Privacy

Head of Department of Chemical Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis presented: May 22, 1991

Presented by Scott Sinquefield

ACKNOWLEDGMENTS

A special thank you goes to my parents for their love, understanding, and support over the years. Because of them, I will never stop challenging myself. I would like to thank Dr. William J. Fredrick, my advisor, for his seemingly endless patience and encouragement over the five years that I have known him. His energy and enthusiasm have been a source of motivation for me. In addition, I would like to thank some of the many professors at Oregon State University from whom I have learned: Dr. Peter Nelson, Dr. Keith Levien, Dr. Charles Wicks, and Dr. Octave Levenspiel.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A Microcomputer Software Package for Simulation of Non-Ideal Aqueous Electrolyte Systems at Equilibrium

INTRODUCTION

If one is faced with a simple (single salt) aqueous solubility problem, one could simply look up a value in a reference book or measure the solubility experimentally. When there are a number of different salts present in the system, the problem of predicting the phase assemblage becomes disproportionatly more complex due to the interdependent nature of the chemical species involved. Each specie affects the behavior of all of the other species present resulting in a highly non-linear problem. This interdependence is weak enough to be ignored at low concentrations (i.e. when the total ionic strength is less than about 0.1 moles per kilogram). However, when concentrations are sufficiently high such that solid phases precipitate, then the interdependence of ion activities must be taken into consideration if any reasonable predictions are to be made. The problem is further compounded by the fact that the equilibrium concentrations of the various species in any system will usually vary by several orders of magnitude. This type of problem is ideally suited for a computer.

Solving highly non-ideal equilibrium problems with a computer requires three things: 1) A mathmatical formulation of the equilibrium problem and a suitable algorithm to solve it. 2) A method of estimating the activities of the solution species which takes into account their interdependent nature. 3) The necessary physical, chemical, and thermodynamic data of the species in the system as well as any empirical data required by the activity model.

The overall objective of this thesis is to provide a microcomputer software package for simulating non-ideal aqueous electrolyte systems at equilibrium. This project has been guided by the following specific objectives:

1) To assemble a reliable yet robust aqueous equlibrium simulator that can handle concentrated (highly non-ideal) multicomponent electrolyte systems.

2) It should provide good estimates of equilibrium concentrations and solid phase speciation at temperatures up to $100^{\circ}C$.

3) All necessary data should be read from a database that is easily expanded to meet the user's individual needs.
4) The program should run on a IBM or compatable microcomputer as this is what is typically readily available to the average engineer.

A number of computer programs are available for calculating chemical equilibria in electrolyte systems. These are reviewed in the following section of this thesis. All of these programs failed to meet the requirements set forth in at least one aspect.

The result of this thesis is NAELS (Nonideal Aqueous ELectrolyte Systems). NAELS is a microcomputer software package that calculates the equilibrium composition of aqueous electrolyte systems. In the most general case it can solve aqueous systems with a gas phase and several solid phases present. The soluble gaseous species, however, must be at constant partial pressure (i.e. an open system). It is currently dimensioned to handle a maximum of 18 cations, 18 anions, 8 neutral aqueous species, and 96 total species in all phases combined. It uses a modified Newton-Raphson algorithm to find the phase assemblage at which the total free energy of the system is at a minimum. This algorithm, developed by Weare, et al. (1987), is combined with a user expandable data base and various activity coefficient estimation methods to form the program NAELS. The development of NAELS and the content of its components are described in this thesis, and examples of its application to solubility problems is demonstrated.

REVIEW OF EXISTING ELECTROLYTE EQUILIBRIUM PROGRAMS

There are a number of existing computer programs for solving equilibrium problems. None could be found that meet all of the objectives of this project. Some of the more widely used ones are summarized here.

WATEQ

The WATEQ program was developed by the U.S. Geological Survey in order to predict both trace and major species concentrations and solid formation. It is limited to 25°C and low to moderate ionic strength (0-3 molal). It uses an extended Debye-Huckel activity coefficient model. The reader is refered to Zemaitis (1986) for the specifics of the Debye-Huckel model. It is limited to low ionic strengths. It is written in FORTRAN and will run on a microcomputer.

REDEQL

This program was developed by the Environmental Protection Agency. It is based on the Davies equation and is intended for water quality use. The program includes an extensive databank for metal-ligand complexes. Based upon the Newton-Raphson technique, it allows for solid formation. It is limited to moderate ionic strengths (0-1 molal) and 25°C. It is also microcomputer compatable.

ECES

This is an extensive general purpose vapor-liquid-solid prediction program for aqueous electrolyte systems developed by OLI systems Inc., 52 South Street, Morristown, NJ 07960. It uses an optional Bromley or Pitzer model to estimate activity coefficients. It can model systems from 0-200 atm., 0-250°C, and 0-30 molal ionic strength. It requires a VAX, UNIVAC, or CDC/NOS computer to run. It has an extensive private database.

MICROQL

Written in the BASIC computer language, MICROQL has no database and no model for activity coefficients. The user inputs constant activity coefficients. It will simulate systems at any temperature as the user inputs all thermodynamic data. It was written for microcomputer use.

SOLGASMIX

Created at the University of Umea, Sweden (1974), SOLGASMIX is a general gas-liquid-solid equlibrium program. There is no thermodynamic database and the user must also supply the activity coefficient model for the type of problem under consideration. The program is proprietary.

GIBBS

Created by R. Gautem for his PhD thesis at the Dept. of Chemical Engineering of the University of Pennsilvania (1979). Written in the form of a FORTRAN subroutine, GIBBS computes physical and chemical equilibrium. The user must supply the thermodynamic data as well as the activity coefficient equations. This program is public domain and can be run on a microcomputer.

ISIS

Created by B. Kelly for his M.S. Thesis at Oregon State University (1988), ISIS uses Pitzer's model for estimating activity coefficients. It also includes a database. ISIS was written for use on a CDC Cyber computer and had limited success when used on a microcomputer.

There are several considerations when evaluating an equilibrium program. Computers solve problems iteratively. The stability and robustness of the algorithm determine whether or not the program will converge to a solution and how long it will take. In all of the above programs (except possibly ECES) the activity coefficients are treated as constants and then revised at each step of the iterative process. In highly non-ideal electrolyte systems the activity coefficients can change faster than the mole numbers as the iterations proceed. Since the activity is the product of the activity coefficient and the mole number, the algorithm can cycle or diverge rather than converge.

Another consideration is the activity model itself. In dilute solutions, the simpler models (i.e. extended Debye-Huckel, Davies, etc) work fine. However, when electrolyte concentrations are high enough such that solids precipitate, then the total ionic strength will be moderate to high in most systems. This is especially true in multicomponent systems. Pitzer's model provides good estimates at high ionic strengths and at temperatures other than

25°C. and the empirical parameters for common species can be found in the literature. Weare, (1987) has developed an extended Pitzer model which takes into account the interactions between charged and neutral species as well as unlike neutrals. Unfortunately, only a very few of the empirical parameters have been measured at this time. Each unique doublet or triplet of species has associated with it an empirical parameter. Pitzer considered interactions between only charged species. Weare choose to include neutral species. The reader can estimate the number of combinations possible.

The presence of a user expandable database is of enormous convenience when using an equilibrium simulator if one is dealing with many different systems. If a database is not present, then all of the necessary thermodynamic and mass balance information, along with the empirical parameters for the activity coefficient model must be read from an input file in matrix form. If a species is added or removed, then the entire matrix must be restructured, redimensioned, and reentered.

THE SOLUBILITY PROBLEM

We know from thermodynamics that any system will seek equilibrium at its lowest energy state. In a chemical system at equilibrium, the Gibb's free energy is at a minimum. The general expression for the free energy of species i is written as:

$$G_i = \mu_i n_i$$

where μ_i is the chemical potential and is simply the Gibb's free energy for species i on a molar basis. n_i is the number of moles of species i. The total free energy of the system is just the summation of the free energies of all of the species in all of the phases. At equilibrium this sum will be at a minimum:

$$\text{Minimize } G = \sum_i \mu_i n_i \text{ for all species in all phases.}$$

The problem is subject to the constraints of bulk compostion, mass balance, electroneutrality, and non-negativity (ie $n_i \geq 0$). As an optimization problem it would be written as follows:

$$\text{minimize } G = \sum_{j=1}^{n^t} \mu_j n_j$$

$$\text{subject to } \sum_{j=1}^{n^t} A_{ji} n_j = b_i \quad i=1, m_c \quad \text{(mass balance)}$$

$$\sum_{j \text{ in } s} z_j n_j = 0 \quad s=1, e_c \quad \text{(charge balance)}$$

$$n_j \geq 0 \quad \text{for all } j \quad \text{(non-negativity)}$$

where G is the Gibbs free energy

μ_j is the chemical potential of species j

n_j is the number of moles of species j

n^t is the total number of species in the system

m_c is the number of system elements

A_{ji} is the number of moles of element i in one mole of species j

z_j is the charge of the j^{th} species in electrolyte solution phase s

e_c is the number of electrolyte solution phases

b_i is the number of moles of element i

To solve the problem the method of Lagrangian multipliers is used to incorporate the constraints into an unconstrained objective function. The reader is refered to Weare, et al., (1987) for a complete description of the algorithm used to find the solution phase assemblage. It is outlined here. The unconstrained Lagrangian function is written as

$$L(\vec{n}, \vec{t}, \vec{\kappa}, \vec{\eta}, \vec{\omega}) = \sum_{j=1}^{n^t} \mu_j n_j - \sum_{i=1}^{c} \kappa_i \sum_{j=1}^{n^t} (A_{ji} n_j - b_i) - \sum_{i=1}^{e} \eta_i \sum_{j \text{ in } i} z_j n_j - \sum_{j=1}^{n^t} \omega_j (n_j - t_j^2)$$

where κ_i is the Lagrangian multiplier for mass balance constraint i

η_i is the Lagrangian multiplier for charge balance constraint i

ω_j is the Lagrangian multiplier for the j^{th} non-negativity constraint

t_j^2 is the slack variable for the inequality constraint

The necessary conditions for a local minimum are obtained by differentiating L with respect to all variables and setting the derivatives equal to zero:

$$\frac{\partial L}{\partial n_k} = 0 = \mu_k - \sum_{i=1}^{c} A_{ki} \kappa_i - \omega_k \quad \text{for species k in non-electrolyte phase}$$

$$\frac{\partial L}{\partial n_k} = 0 = \mu_k - \sum_{i=1}^{c} A_{ki} \kappa_i - \eta_s z_k - \omega_k \quad \text{for species k in electrolyte phase s}$$

$$\frac{\partial L}{\partial \kappa_k} = 0 = \sum_{j=1}^{n^t} (A_{jk} n_j - b_k) \quad k=1, c$$

$$\frac{\partial L}{\partial \eta_k} = 0 = \sum_{j \text{ in } k} z_j n_j \quad k=1, e$$

$$\frac{\partial L}{\partial t_k} = 0 = 2\omega_k t_k \quad k=1,\ n^t$$

$$\frac{\partial L}{\partial \omega_k} = 0 = n_k - t_k^2 \quad k=1,\ n^t$$

The algorithm begins with a feasible point and tests for a local minimum. If a minimum is not present, then a Newton-Raphson step is taken and tests are performed to determine if the addition or removal of a phase will further decrease the system free energy. When a local minimum has been reached and addition or removal of a phase results in an increase in free energy, then a global minimum has been reached.

ACTIVITY COEFFICIENT ESTIMATION

In order to carry out the minimization procedure we must have an expression for the chemical potential. The expression for the chemical potential of species i is written as

$$\mu_i = \mu_i^\circ + RT\ln(a_i) \tag{1}$$

where μ_i° is the standard chemical potential for species i. R is the ideal gas constant and T is absolute temperature. a_i is the activity of species i. For solids (and gases at constant partial pressure) the activity is taken as unity. The chemical potential for solids reduces to $\mu_i = \mu_i^\circ$. For solution species, activity is expressed as the product of the activity coefficient, γ, and the molality, m:

$$a_i = \gamma_i m_i \tag{2}$$

Combining equations 1 and 2 and rearranging yields

$$\frac{\mu_i}{RT} = \frac{\mu_i^\circ}{RT} + \ln(\gamma_i) + \ln(m_i) \tag{3}$$

In an aqueous ideal solution phase the activity coefficients are taken as unity. The free energy function then becomes

$$\frac{\mu_i}{RT} = \frac{\mu_i^\circ}{RT} + \ln(m_i) \tag{4}$$

A relatively simple model for activity coefficients is the Davies (1962) equation. It is valid for $I<0.5$ mol/liter. Equation 5 is used to calculate activity coefficients for charged species only; neutral species are assigned unit activity.

$$\ln \gamma_k = -3A^\phi Z_k^2\left[\frac{\sqrt{I}}{1+\sqrt{I}} - 0.3I\right] \tag{5}$$

$$I = \frac{1}{2}\sum_j m_j z_j^2 \qquad \text{ionic strength} \tag{6}$$

$$A^\phi = \tfrac{1}{3}(2\pi N_0 d_w/1000)^{1/2}(e^2/DkT)^{3/2} \tag{7}$$

Z_i is the charge of species k, I is the ionic strength, and A^ϕ is the Debye-Huckel constant for osmotic coefficients. In spite of its appearance, A^ϕ is a function of temperature only. Beyer and Staples (1986) provide a thorough discussion of A^ϕ as well as a tabulation of A^ϕ vesus T from 0 to 350°C. This data was used to fit the polynomial $A^\phi(T)$ in the database.

Pitzer's model, though rather complex, provides the best estimates of solubility in multicomponent systems. A brief summary of the equations used is presented here for convenience. This model contains many measured values refered to as 'ion interaction parameters'. A familiarity with the equations below simplifies the task of searching the literature for ion interaction parameters not allready contained in the database. For a complete derivation of the model refer to Pitzer and coworkers (1973-1975). In the equations that follow the subscripts M, c, and c′ refer to cations while X, a, and a′ refer to anions. The interaction parameters will be labeled as such.

$$\ln \gamma_M = z_M^2 F + \sum_a m_a(2B_{Ma} + ZC_{Ma}) + \sum_c m_c\left[2\Phi_{Mc} + \sum_a m_a\psi_{Mca}\right] + \sum_{a \neq a'}\sum m_a m_{a'}\psi_{aa'M} + |z_M| \sum_c\sum_a m_c m_a C_{ca} \quad (8a)$$

$$\ln \gamma_X = z_X^2 F + \sum_c m_c(2B_{cX} + ZC_{cX}) + \sum_a m_a\left[2\Phi_{Xa} + \sum_c m_c\psi_{Xac}\right] + \sum_{c \neq c'}\sum m_c m_{c'}\psi_{cc'X} + |z_X| \sum_c\sum_a m_c m_a C_{ca} \quad (8b)$$

$$F = -A^\phi\left(\frac{\sqrt{I}}{1+b\sqrt{I}} + \frac{2}{b}\ln(1+b\sqrt{I})\right) + \sum_c\sum_a m_c m_a B'_{ca} + \sum_{c \neq c'}\sum m_c m_{c'}\Phi'_{cc'} + \sum_{a \neq a'}\sum m_a m_{a'}\Phi'_{aa'} \quad (9)$$

b is an empirical parameter equal to 1.2. $\psi_{cc'a}$ and $\psi_{caa'}$ are ternary mixing parameters and exist for unlike cation-cation-anion and cation-anion-anion triplets.

$$Z = \sum_i m_i|z_i| \quad (10)$$

$$C_{MX} = C^\phi_{MX}/2\sqrt{|z_M z_X|} \quad (11)$$

$$B_{MX} = \beta^0_{MX} + \beta^1_{MX}\, g(\alpha_1\sqrt{I}) + \left\{\beta^2_{MX}\, g(\alpha_2\sqrt{I})\right\} \quad (12a)$$

$$B'_{MX} = \beta^1_{MX}\, g'(\alpha_1\sqrt{I})/I + \left\{\beta^2_{MX} g'(\alpha_2\sqrt{I})/I\right\} \quad (12b)$$

β^0_{MX}, β^1_{MX}, β^2_{MX}, and C^{ϕ}_{MX} are single salt mixing parameters. They exist for each cation-anion combination. However β^2_{MX} is only defined for salts in which both ions are divalent (ie $CaSO_4$). $\alpha_1=1.4$ and $\alpha_2=12.0$ for 2-2 electrolytes in equations 12. For all other charge combinations, $\alpha_1=2.0$ and the bracketed terms do not exist. The functions g and g′ are defined below with $x = \alpha\sqrt{I}$.

$$g(x) = 2[1-(1+x)e^{-x}]/x^2 \quad (13a)$$

$$g'(x) = -2[1-(1+x+\tfrac{1}{2}x^2)e^{-x}]/x^2 \quad (13b)$$

$$\Phi_{ij} = \theta_{ij} + {}^{E}\theta_{ij}(I) \quad (14a)$$

$$\Phi'_{ij} = {}^{E}\theta'_{ij}(I) \quad (14b)$$

$${}^{E}\theta_{ij}(I) = \frac{z_i z_j}{4I}\left[J0(X_{ij}) - \tfrac{1}{2}J0(X_{ii}) - \tfrac{1}{2}J0(X_{jj})\right] \quad (15a)$$

$${}^{E}\theta'_{ij}(I) = \frac{z_i z_j}{8I^2}\left[J1(X_{ij}) - \tfrac{1}{2}J1(X_{ii}) - \tfrac{1}{2}J1(X_{jj})\right] - \frac{{}^{E}\theta_{ij}}{I} \quad (15b)$$

$$X_{ij} = z_i z_j A^{\phi}\sqrt{I}$$

θ_{ij} is a parameter that accounts for interactions between ions of like sign. It exists for unlike cation-cation and anion-anion pairs. ${}^{E}\theta_{ij}(I)$ and ${}^{E}\theta'_{ij}(I)$ are zero when $z_i = z_j$ and are functions of ionic strength, ion pair type, and temperature only. Equations 15 are for cations. Similar equations exist for unlike anion pairs. The expressions for J0(X) and J1(X) are given by Pitzer as

$$J0(X) = \frac{X}{4} - 1 + \frac{1}{X}\int_0^{\infty}\left[1 - e^{\left(-\frac{X}{Y}e^{-Y}\right)}\right] Y^2 dY \quad (16a)$$

$$J1(X) = \frac{X}{4} - \frac{1}{X}\int_0^{\infty}\left[1 - (1+\frac{X}{Y}e^{-Y})\; e^{\left(-\frac{X}{Y}e^{-Y}\right)}\right] Y^2 dY \quad (16b)$$

NAELS uses a Chebychev expansion to evaluate the above integrals numerically.

In most equilibrium simulators the activity coefficients are treated as constants from one step to the next and then recalculated based on the new

concentration values. This works as long as the change in activity coefficients from one step to the next is small compared to the change in concentrations. As discussed previously, in highly non-ideal systems the activity coefficients can change faster than the concentrations from step to step. It makes liitle sense, then, to go to the trouble of using sophisticated algorithms to update concentrations from step to step only to multiply them by ever changing 'constants'. NAELS treats activity coefficients as variable functions of mole numbers and therefore includes the derivatives of activity coefficients with respect to mole numbers in the algorithm.

DATABASE

The database contains all of the necessary thermodynamic data and molecular formulas to describe each chemical species. It is divided in three sections: Mass balance and chemical potential, Pitzer betas, and Pitzer thetapsis. Because of its size, the entire database is not listed in this thesis. Portions of each section are shown in Figures 1, 2, and 3. All of the data found in the database was taken from the periodic table, Wagman, et.al. (1982), Woods and Garrels, (1987), Frederick and Kim (1988), Pitzer (1979), and Weare (1987). The dimensionless free energy values for the species indicated by an asterick in Appendix B were calculated from solubility data found in Linke (1965), and Seidell (1935).

The first column in the mass balance section contains the species identification number. Water is assigned number 99. Cations, begining with H^+, are assigned numbers 100 through 199. Anions, begining with OH^-, are assigned numbers 200 through 299. Neutral solution species are numbered 300 through 399, and pure species (solids and gasses) are numbered from 400 onward. The next column is a character field 24 spaces wide containing the name or chemical formula of the species. Next is a 2 space integer field for the charge of the species. This is left blank for solid species. Following this is 4 space real number field followed by a 3 space integer field. There are 7 pairs of these fields. The integer fields contain atomic numbers of elements as found on the periodic table. The real numbers are the molar amounts found in a given species. For example, specie 215 is $H_2PO_4^-$. It carries a charge of -1. It is made up of 1 atom of element 15 (phosphorous), 4 atoms of element 8 (oxygen), and 2 atoms of element 1 (hydrogen). The second line of each species entry contains a 24 space character field followed by the coefficients of a fourth order polynomial. The character variable contains the temperature range (in °C) over which the polynomial is valid as well as a flag indicating the source of the data used to generate the polynomial. The polynomial relates the dimensionless standard chemical potential (from Equation 3) to temperature and has the following form:

$$\frac{\mu^\circ}{RT} = C_0 + C_1(T - T_{ref}) + C_2(T - T_{ref})^2 + C_3(T - T_{ref})^3 + C_4(T - T_{ref})^4 \quad (17)$$

where R is the gas constant, T is in kelvin, and T_{ref}=298.15 K. This form was chosen for convenience. If μ is known only at 25°C then C_0 will be the only non-zero coefficient. The chemical potential is defined conceptually as follows:

$$\mu = H - TS \tag{18}$$

The difference in chemical potential between two temperatures at constant pressure would then be

$$\mu_2 - \mu_1 = H_2 - H_1 - (T_2S_2 - T_1S_1) \tag{19}$$

but $$H_2 - H_1 = \int_1^2 C_p \, dT \qquad \text{and} \qquad S_2 = S_1 + \int_1^2 \frac{C_p}{T} dT \tag{20}$$

If μ_1 is arbitrarily set equal to the standard free energy of formation at 25°C, and S_1 is the standard entropy at 25°C; then equations 19 and 20 can be integrated, substituted, and rearranged to yield:

$$\mu_2^\circ = \Delta G_f^\circ + \overline{C_p}|_{t_0}^{t_2} \Delta T - t_2 \, \overline{C_p}|_{t_0}^{t_2} \ln(t_2/t_0) - S_{25}^\circ \Delta T \tag{21}$$

where t_0=25°C, $\Delta T = (t_2 - t_0)$, and $\overline{C_p}|_{t_0}^{t_2}$ is the average heat capacity between t_0 and t_2. If $C_p(T)$ data are available, then the integrals in equation 20 can be carried out numerically. Heat capacity equations are published in the literature for many compounds. If the heat capacity of a solid is known only at 25°C, then as an approximation it can be assumed constant over a small temperature range. For solution species, heat capacities change with temperature considerably but are typically available only at 25°C if at all. Appendix C outlines the method of estimating $C_p(T)$ for solution species that was used in this work. Equation 21 should be used only for solids and only in the absence of experimental solubility data. The method of using solubility data to generate chemical potentials for solid (or gaseous) species is discussed in the section on sources of error. This is the preferred method for these species.

It is important to note that μ_2° is not, in general, equivalent to the standard free energy of formation at t_2. The difference between the two is the free energy required to raise the temperature of the elements in the formation reaction from t_0 to t_2. Equilibrium calculations are based on

differences in chemical potential between reactants and products in a balanced equlibrium reaction. The aforementioned difference between μ_2° and ΔG_f at t_2 is the same on both sides of a balanced equilibrium reaction and drops out when ΔG_{rxn} is calculated. Therefore, ΔG_{rxn} is identical to $\Delta\mu_{2,rxn}^\circ$. The extra work of calculating ΔG_f as a function of temperature is unnecessary.

Figure 2 shows the second section of the database which contains the Pitzer beta values. This section is preceded by polynomial coefficients for the function $A^\phi(T)$, and values for the Pitzer alphas. Each line corresonds to a cation-anion combination referenced by species identification numbers in the left hand columns. The column for β^2 is blank for all except 2-2 electrolytes.

The last section, shown in Figure 3, is for Pitzer theta and psi values. In the first half of this section a theta will exist for each unlike cation-cation pair and a psi will exist for each unlike cation-cation-anion triple. The theta will appear on the same line as the two species I.D. numbers. Starting on the following line, the psis will be listed, one for each anion in the order they appear in the database. For example theta for H^+-Mg^{+2} is 0.0891 and psi for H^+-Mg^{+2}-HSO_4^- is -0.0178. The other half of this section is a completely analogous list of anion-anion theta's and cation-anion-anion psi's. Adding data to the Pitzer section involves more than simply adding lines to the end of a list. The arrangment should be studied carefully before any attempt is made to build onto it.

FIGURE 1
Selected portions of mass balance section of database
FORMAT(I4, A24, I2, 7(F4.3, I3) /14X, 5E13.5) aqueous
FORMAT(I4, A24, 2X,7(F4.3, I3) /14X, 5E13.5) solids

```
 99 H2O                0  2. 1  1. 8
          -9.56611E+1   2.91704E-1  -9.82177E-4   2.57158E-6  -3.53354E-9
100 H+                +1  1. 1
   25-200   0.00000E+0   4.35345E-3  -1.27810E-4   2.85164E-7  -3.67000E-10
101 Na+               +1  1. 11
   25-200  -1.05656E+2   3.22011E-1  -1.13238E-3   2.89251E-6  -3.93920E-9
102 K+                +1  1. 19
   25-200  -1.14275E+2   3.24604E-1  -1.13209E-3   2.90867E-6  -3.97770E-9
103 Mg++              +2  1. 12
   25-200  -1.83472E+2   7.05555E-1  -2.40987E-3   6.13515E-6  -8.33210E-9
104 MgOH+             +1  1. 12  1.  8  1.  1
   25      -2.52822E+2
  :
112 Mn++              +2  1. 25
   25-200  -9.20184E+1   3.60767E-1  -1.29107E-3   3.26602E-6  -4.42840E-9
113 MnOH+             +1  1. 25  1.  8  1.  1
   25-200  -1.63382E+2   5.60973E-1  -1.91448E-3   4.88671E-6  -6.63760E-9
200 OH-               -1  1.  8  1.  1
   25-200  -6.34344E+1   2.14183E-1  -5.44900E-4   1.45959E-6  -2.02110E-9
201 Cl-               -1  1. 17
   25-200  -5.29394E+1   1.38462E-1  -3.06790E-4   8.53212E-7  -1.19710E-9
202 CO3--             -2  1.  6  3.  8
   25-200  -2.12926E+2   7.38806E-1  -2.04005E-3   5.37846E-6  -7.40120E-9
  :
215 H2PO4-            -1  1. 15  4.  8  2.  1
   25-200  -4.55971E+2   1.46582E+0  -4.62432E-3   1.19862E-5  -1.64150E-8
216 VO3-              -1  1. 23  3.  8
   25-200  -3.16115E+2   1.02330E+0  -3.05737E-3   7.99006E-6  -1.09820E-8
300 H2CO3(aq)          0 3.  8  1.  6  2.  1
   25      -2.51410E+2
301 NaHCO3(aq)         0  1. 11  3.  8  1.  6  1.  1
   25      -3.42461E+2
302 MgCO3(aq)          0  1. 12  3.  8  1.  6
   25      -4.04426E+2
  :
307 H3PO4(aq)          0  1. 15  4.  8  3.  1
   25      -4.60945E+2
400 SiO2                  1. 14  2.  8
   25-200* -3.43184E+2   1.12871E+0  -3.64467E-3   9.43049E-6  -1.29007E-8
401 Al2O3                 2. 13  3.  8
   25-200* -6.38322E+2   2.11400E+0  -6.82365E-3   1.76541E-5  -2.41494E-8
402 Al(OH)3               1. 13  3.  8  3.  1
   25      -4.65659E+2
403 AlCl3                 1. 13  3. 17
   25-200* -2.53667E+2   8.03667E-1  -2.63307E-3   6.84154E-6  -9.37387E-9
404 AlCl3*6H2O            1. 13  3. 17  6.  8 12.  1
   25-200* -9.12159E+2   2.92210E+0  -9.54994E-3   2.47961E-5  -3.39650E-8
405 Al2(SO4)3             2. 13  3. 16 12.  8
   25-200* -1.25056E+3   4.08540E+0  -1.32543E-2   3.43418E-5  -4.70031E-8
```

FIGURE 2
Pitzer betas, sample section of database
FORMAT(I3, 1X, I3, 3X, 4F14.8)

```
 :
486 KAl3Si3O10(OH)2          1. 19  3. 14  3. 13 12.  8  2.  1
    25-200*    -2.26251E+3   7.44237E+0  -2.40404E-2   6.22104E-5  -2.51059E-8
487 KH2PO4                   1. 19  1. 15  4.  8  2.  1
    25-200*    -5.71174E+2   1.85567E+0  -6.01954E-3   1.55961E-5  -2.13458E-8
488 2Na2SO4*Na2CO3           2. 16  6. 11 11.  8  1.  6
    25*        -1.44996E+3
900 APHI
    3.76978E-1    4.44771E-4    4.92829E-6   -1.41090E-8    4.31240E-11
    1.2           2.0           1.4          12.0
                BETA0        BETA1        BETA2        C-PHI
100 200         .00000       .00000                    .00000
100 201         .20332      -.01668                   -.00372
100 202         .00000       .00000                    .00000
100 203         .00000       .00000                    .00000
100 204         .00000       .00000                    .00000
100 205         .02980       .00000                    .04380
100 206         .20650       .55560                    .00000
100 207         .00000       .00000                    .00000
100 208         .00000       .00000                    .00000
100 209         .00000       .00000                    .00000
100 210         .00000       .00000                    .00000
100 211         .00000       .00000                    .00000
100 212         .00000       .00000                    .00000
100 213         .00000       .00000                    .00000
100 214         .00000       .00000                    .00000
100 215         .00000       .00000                    .00000
100 216         .00000       .00000                    .00000
101 200         .08640       .25300                    .00440
101 201         .07722       .25183                    .00106
101 202         .03990      1.38900                    .00440
 :   :
```

FIGURE 3
Pitzer theta-psis, sample section of database
FORMAT(I3, 1X, I3, 8X, F9.5, 2(/ 7X, 9F8.4))

```
 :   :
113 214        .00000        .00000                    .00000
113 215        .00000        .00000                    .00000
113 216        .00000        .00000                    .00000
901
100 101            .03600
          .0000  -.0033  .0000  .0000  .0000  .0000  -.0129  .0000  .0000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000
100 102            .00670
          .0000  -.0081  .0000  .0000  .0000  .1970  -.0265  .0000  .0000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000
100 103            .08910
          .0000  -.0006  .0000  .0000  .0000  .0000  -.0178  .0000  .0000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000
100 104            .00000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000  .0000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000
100 105            .00000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000  .0000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000
100 106            .06820
          .0000   .0043  .0000  .0000  .0000  .0000   .0000  .0000  .0000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000
100 107            .00000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000  .0000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000
100 108            .00000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000  .0000
          .0000   .0000  .0000  .0000  .0000  .0000   .0000  .0000
 :   :
```

THE INPUT FILE

An example input file is shown in Figure 4. Since the input file is formatted, it is best to modify an existing input file to suit a user's individual needs rather than to type one from scratch. It must be a standard ASCII file with the left margin set to zero.

The first line is a label for the file and is ignored by the program. The next four lines are print options for the output file. They should be assigned either 0 or 1 depending on the options desired. When IPRINT=1, diagnostics of the main Newton minimization procedure are written to the output file at each Newton iteration. Diagnostics include mole numbers, free energies, step directions, and the type of step (feasibility, phase addition or deletion, refinement, or normal step). Unless a simulation is crashing for no apparent reason it is not recomended that IPRINT be set to 1 as the resulting output file will be quite large.

When IPR=1, the solution to the problem is written to the output file. When IPR2=1, the diagnostics of the subminimization procedure (called the "dual" problem) are written to the output file. When IPR3=1, the phase (Lagrangian) multipliers at the solution point will be written to the output file.

All of these input variables have integer values. The next input, the system temperature, must be in the form of a real number. Fortran distinguishes between real and integer numbers with the use of a decimal point. Input the system temperature in degrees Celcius.

At this point it is easiest to skip a few lines and return to them later. The line labeled "NAME" is for a user specified identifier that will be written to the output file. The next two lines are column headings and must be left as is. The symbols indicate available column width for convenience. In this section the species that are to be considered as possibly present in the phase are specified along with the type of phase under consideration. The aqueous phase must be entered first. The first line of any phase will consist of a integer in the first column indicating the type of phase:

1—Pure phase (i.e. solid or gas)

2—Aqueous ideal solution phase

3—Aqueous phase with Davies activity coefficient model

4—Aqueous phase with Pitzer activity coefficient model

Following the phase type indicator, all species that are to be considered possibly present in the phase should be listed by their database I.D. numbers. Executing the program "DBLIST.EXE" will produce a list of all the species, both aqueous and solid, contained in the database (appendix B). The aqueous phase must be entered first and the first specie must be water (specie #99). The remaining aqueous species must be in ascending order as shown in the example. A zero marks the end of a phase. A second zero would mark the end of the input file. A phase type identifier marks the presence of another phase (usually a solid or gas phase. Although each solid is technically a separate phase, all solids and gases can be listed together as one phase. A gas phase is distinguished from a solid phase by specifying its partial pressure. Aside from that they are treated identically.

Once the identities of the component species have been specified, the total elemental concentrations for the whole system must be specified. The variable TAU is used for this purpose. Its units are either moles or molality depending on the value of TAU chosen for specie #99. The number of moles in a kilogram of water is 55.50807. If one wishes to work in molal units, then TAU for specie #99 should be 55.550807. If there are several solutes dissolved in 10.0 moles of water then TAU would be 10.0 for specie #99. TAU can be specified for aqueous species as well as pure phases but not both within a given reaction set. For example if TAU were specified for K^+ and for Cl^-, then TAU for KCl(s) must be left blank; and vise-versa. It is usually simplest to input all salts in aqueous form in their most ionized state (i.e. aluminum in the form of Al^{3+}, phosphate in the form of PO_4^{3-}, etc.) and let NAELS deal with the speciation.

When predicting the solubility of a particular salt it is tempting to specify an absurdly large amount of its component ions on the assumption that the excess will simply precipitate. While this approach will work for a anhydrate salt, it will cause numerical difficulties with a hydrated salt. In effect, the water is used up. Since most precipitates in aqueous systems are hydrates, it is best to calculate solubilities by starting with a subsaturated system and progressively concentrate it until the precipitate appears. This

means making several runs gradually increasing the bulk concentration until a saturation is reached. If TAU is specified for any aqueous species, eletroneutrality must be observed. An alternative is to specify TAU for one or more solids and let them dissolve.

The next input to consider is ETA. The natural logarithm of ETA (if non-zero) is added to the standard chemical potential of the species. This is how the fugacity of a gas is fixed in a open system. Input ETA in units of atmospheres. Specifying a value for ETA has no effect on TAU, which must be dealt with separately. By definition, a gas in an open system at constant partial pressure (ie O_2, N_2, CO_2, etc in the atmosphere) can never be used up. However a finite bulk concentration must be specified. TAU for a gaseous species should be large enough such that not all of it is dissolved into the system.

It is possible to perform a series of calculations with NAELS, making changes in the total system concentration at each step. For example, titration, evaporation, dillution, precipitation followed by filtration, etc., all fall in this catagory. The variables NTR, FRAX, ALPHA, ALPHA2, SIG, and ZETA are the parameters used to control the multi-step calculations. NTR is the desired number of steps in the series ($NTR \geq 1$). FRAX is the fraction of solids formed that is to be removed at each step ($0 \leq FRAX \leq 1$). When FRAX=0 no solid removal takes place. When FRAX=1 all solids formed are removed at each step. This is useful to model a process involving stepwise filtration. ALPHA and SIG are used to add or remove an absolute **amount** of a specie at each step ($0 \leq ALPHA \leq 1$, $SIG < 0$ for removal, $SIG > 0$ for addition). The number of moles added or removed is the product ALPHA*SIG. The stoichiometry must be choosen to ensure that electroneutrality is preserved after the step. ALPHA and SIG are typically used for step-wise additions of a solute. ALPHA2 and ZETA are used to add or removed a specific **fraction** of a specie at each step ($0 \leq ALPHA2 \leq 1$, $ZETA < 0$ for removal, $ZETA > 0$ for addition). The number of moles added or removed is the product ALPHA2*ZETA multiplied by the equilibrium molality of the specie. The most common usage of ALPHA2 and ZETA is to add or remove water. It is important to note that a step-wise change, whether fractional or absolute, is relative to the equilibrim concentration of a given specie, which may be quite different than the value

of TAU for that specie.

Examples of input files are shown on the following pages. The corresponding output file for example 1 is shown in figure 6 in the section 'OUTPUT FILE'.

FIGURE 4

Example input file for $CaCl_2$–$MgCl_2$ system

```
INPUT FILE FOR EQUILIBRIUM SIMULATION PROGRAM
IPRINT=0
IPR=1
IPR2=0
1PR3=0
SYSTEM TEMPERATURE (C)=25.0
NTR=5
ALPHA=1.0
FRAX=0.8
ALPHA2=0.0
NAME:  CaCl2–MgCl2 system
 #        TAU                SIG                ETA                ZETA
###$$$$$$$$$$$$$$$**************@@@@@@@@@@@@@@@&&&&&&&&&&&&&&&
4
99       55.50807                  —3.0
100
103      4.500
104
106      0.5000                    0.1
107
200
201     10.00                      0.2
0
1
427
435
436
450
0
0
```

Figure 4 shows an example input file for a 5 step evaporation sequence. 4.5 moles of magnesium chloride and 0.5 moles of calcium chloride are dissolved in 1 kilogram of water. At each step, 3 moles of water are removed, 0.1 moles of calcium chloride are added, and 80% of the solids (if any) are removed. The solid species to be considered are $MgCl_2 \cdot 6H_2O$, $Ca(OH)_2$, $CaCl_2 \cdot 6H_2O$, and $CaCl_2 \cdot 2MgCl_2 \cdot 12H_2O$. Pitzer's activity model is to be used for the aqueous phase.

FIGURE 5

Example input file for $CaCO_3$–$Ca(OH)_2$ system

```
INPUT FILE FOR EQUILIBRIUM SIMULATION PROGRAM
IPRINT=0
IPR=1
IPR2=0
1PR3=0
SYSTEM TEMPERATURE (C)=25.0
NTR=10
ALPHA=0.1
FRAX=0.0
ALPHA2=0.05
NAME:  CaCO3–Ca(OH)2 system
 #        TAU                SIG                ETA                ZETA
###$$$$$$$$$$$$$$$**************@@@@@@@@@@@@@@@&&&&&&&&&&&&&&&
4
99      55.50807                                                    –1.0
100
106                                2.0
107
108
200                                4.0
202
203
204
300
301
304     5.00
0
1
435
445
489      10.00                                  0.005
0
0
```

Figure 5 shows an example input file for a 10 step sequence. A 5.0 molal calcium carbonate solution is placed in contact with a carbon dioxide atmosphere at 0.005 atm. partial pressure. At each step 0.2 moles of calcium hydroxide are added while 5% of the water is removed. The 10 moles specified for initial CO_2 is somewhat arbitrary. In the case of a gas in an open system, an amount needs to be specified such that not all of it is used up. $CaOH_2$, $CaCO_3$, and CO_2(gas) are the pure phases to be considered.

THE OUTPUT FILE

The output file for Example 1 is shown in Figure 6. It begins by printing all of the relevent data retrieved from the database: dimensionless standard chemical potential, Pitzer parameters, atomic composition, bulk composition, etc. The solution itself lists total moles, the logarithm of the activity, and the molar concentration, of each species. The molality is printed in both scientific and standard notation for convenience. The activity of water and the ionic strength are also provided. Only the solids that actually formed are listed are listed along with their respective mole numbers. A solution will be printed for each step in a multistep simulation.

Some other information of lesser utility is included in the output file. ITER is the number of **full** Newton steps taken in the convergence process. NAELS will take only a partial step if a full step crosses a constraint boundary. IFUN is the number of functional evaluations at a full Newton step. If ITER=IFUN, then all Newton steps were full steps. IREFIN is the number of refinement steps taken. When a solution species is at or below a boundary value, a small refinement step is taken. The MAX RESIDUAL is the largest absolute value of the residual vector that is formed by subtracting the dot product of the solution vector and the mass balance matrix from the vector of total mole numbers. It is written in equation form as

$$\text{MAX} \quad \left\| A_{ij} \cdot n_j - b_i \right\| \qquad j=1,\ n^t \text{ and } i=1,\ m_c$$

This number should be significantly smaller than the smallest value in the MOLES column of the solution. NAELS calculates a SCALE FACTOR that is unique to each minimization based on the characteristics of the uncontrained Lagrangian objective function. The entire problem is scaled up to improve the precision of the solution. For detailed information on the algorithm itself the reader is refered to Weare, et al. (1987).

FIGURE 6

Example output file for $CaCl_2$–$MgCl_2$ system

SOLUTION PHASE 1 PHASE TYPE 4

	SPECIES	CHARGE	U0/RT
1	H2O	0	-95.66110000
2	H+	1	.00000000
3	Mg++	2	-183.47200000
4	MgOH+	1	-252.82200000
5	Ca++	2	-223.32200000
6	CaOH+	1	-289.81500000
7	OH-	-1	-63.43440000
8	Cl-	-1	-52.93940000

NON-IDEAL AQUEOUS ELECTROLYTE SOLUTION DATA

APHI= .39097 BETA= 1.200 ALPHA= 2.000 1.400 12.000

		B0	B1	B2	CPHI
H+	OH-	.00000	.00000	.00000	.00000
H+	Cl-	.20332	-.01668	.00000	-.00372
Mg++	OH-	.00000	.00000	.00000	.00000
Mg++	Cl-	.35573	1.61738	.00000	.00474
MgOH+	OH-	.00000	.00000	.00000	.00000
MgOH+	Cl-	-.10000	1.65800	.00000	.00000
Ca++	OH-	-.17470	-.23030	.00000	.00000
Ca++	Cl-	.32579	1.38412	.00000	-.00174
CaOH+	OH-	.00000	.00000	.00000	.00000
CaOH+	Cl-	.00000	.00000	.00000	.00000

THETA-PSI		THETA	OH-	Cl-
H+	Mg++	.0891	.0000	-.0006
H+	MgOH+	.0000	.0000	.0000
H+	Ca++	.0682	.0000	.0043
H+	CaOH+	.0000	.0000	.0000
Mg++	MgOH+	.0000	.0000	.0280
Mg++	Ca++	.0070	.0000	-.0120
Mg++	CaOH+	.0000	.0000	.0000
MgOH+	Ca++	.0000	.0000	.0000
MgOH+	CaOH+	.0000	.0000	.0000
Ca++	CaOH+	.0000	.0000	.0000

THETA-PSI		THETA	H+	Mg++	MgOH+	Ca++	CaOH+
OH-	Cl-	-.0500	.0000	.0000	.0000	-.0250	.0000

PURE PHASES

	SPECIES	U0/RT
9	MgCl2*6H2O	-852.97000000
10	Ca(OH)2	-362.46300000
11	CaCl2*6H2O	-893.80500000
12	CaCl2*2MgCl2*12H2O	-2015.79000000

FIGURE 6 (continued)

```
 CONSTRAINT EQUATIONS
    SPECIES                0      1      8      12     17     20
   1 H2O                  .0    2.0    1.0     .0     .0     .0
   2 H+                  1.0     .0     .0     .0     .0     .0
   3 Mg++                2.0     .0     .0    1.0     .0     .0
   4 MgOH+               1.0    1.0    1.0    1.0     .0     .0
   5 Ca++                2.0     .0     .0     .0     .0    1.0
   6 CaOH+               1.0    1.0    1.0     .0     .0    1.0
   7 OH-                -1.0    1.0    1.0     .0     .0     .0
   8 Cl-                -1.0     .0     .0     .0    1.0     .0
   9 MgCl2*6H2O          .0   12.0    6.0    1.0    2.0     .0
  10 Ca(OH)2             .0    2.0    2.0     .0     .0    1.0
  11 CaCl2*6H2O          .0   12.0    6.0     .0    2.0    1.0
  12 CaCl2*2MgCl2*12H    .0   24.0   12.0    2.0    6.0    1.0
     RANK=  5     MAXIMUM PHASES=  4
INPUT DATA      CaCl2-MgCl2 solubility
NTR,ALPHA,FRAX,ALPHA2=     5  1.00000    .80000    .00000
      SPECIES      TAU          SIG          ETA          ZETA
1     1 H2O    5.55081D+01 - 3.00000D+00  0.00000D+00  0.00000D+00
3     1 Mg++   4.50000D+00   0.00000D+00  0.00000D+00  0.00000D+00
5     1 Ca++   5.00000D-01   1.00000D-01  0.00000D+00  0.00000D+00
8     1 Cl-    1.00000D+01   2.00000D-01  0.00000D+00  0.00000D+00

   STEP    1  SCALE FACTOR=  9.0077D-03
SPECIES                  MOLES          LN(A)          CONC          CONC
 1 H2O                5.55081D+01   -8.05002D-01   5.55084D+01   55.50837
 2 H+                 4.95595D-06    1.22278D+01   4.95598D-06     .00000
 3 Mg++               4.50000D+00    1.52827D+00   4.50002D+00    4.50002
 4 MgOH+              4.94609D-06   -1.64166D+01   4.94612D-06     .00000
 5 Ca++               5.00000D-01   -5.36143D+00   5.00003D-01     .50000
 6 CaOH+              8.82285D-09   -2.31067D+01   8.82290D-09     .00000
 7 OH-                1.04237D-09   -2.08039D+01   1.04238D-09     .00000
 8 Cl-                1.00000D+01    7.66219D+00   1.00001D+01   10.00006
  AH2O=       .44708708  IONIC STRENGTH=      15.00007838 mol/kg
 CONVERGENCE CRITERIA    1.077D-17   1.556D-08   0.000D+00
ITER=    10  IFUN=    18  IREFIN=     1

   STEP    2  SCALE FACTOR=  9.5223D-03
SPECIES                  MOLES          LN(A)          CONC          CONC
 1 H2O                5.25081D+01   -9.23134D-01   5.55084D+01   55.50837
 2 H+                 5.09816D-06   -1.20758D+01   5.38947D-06     .00001
 3 Mg++               4.49999D+00   -1.41153D+00   4.75712D+00    4.75712
 4 MgOH+              5.08809D-06   -1.65700D+01   5.37883D-06     .00001
 5 Ca++               6.00000D-01   -5.23566D+00   6.34284D-01     .63428
 6 CaOH+              9.34971D-09   -2.32511D+01   9.88395D-09     .00000
 7 OH-                7.19375D-10   -2.10740D+01   7.60480D-10     .00000
 8 Cl-                1.02000D+01    8.23443D+00   1.07828D+01   10.78283
  AH2O=       .39727187  IONIC STRENGTH=      16.17423594 mol/kg
 CONVERGENCE CRITERIA    1.210D-17   3.351D-09   0.000D+00
ITER=     4  IFUN=     4  IREFIN=     1
```

FIGURE 6 (continued)

```
   STEP    3  SCALE FACTOR=  1.0099D-02
SPECIES                  MOLES        LN(A)          CONC          CONC
 1 H2O                4.95081D+01  -1.06542D+00   5.55084D+01   55.50837
 2 H+                 5.23386D-06  -1.19171D+01   5.86820D-06     .00001
 3 Mg++               4.49999D+00  -1.27630D+00   5.04539D+00    5.04539
 4 MgOH+              5.22391D-06  -1.67358D+01   5.85704D-06     .00001
 5 Ca++               7.00000D-01  -5.14997D+00   7.84839D-01     .78484
 6 CaOH+              9.47302D-09  -2.34664D+01   1.06211D-08     .00000
 7 OH-                4.78277D-10  -2.13751D+01   5.36244D-10     .00000
 8 Cl-                1.04000D+01   8.87384D+00   1.16605D+01   11.66047
  AH2O=       .34458235  IONIC STRENGTH=      17.49069246 mol/kg
 CONVERGENCE CRITERIA    1.104D-17   7.136D-09   0.000D+00
ITER=     4  IFUN=     4  IREFIN=     1

   STEP    4  SCALE FACTOR=  1.0751D-02
SPECIES                  MOLES        LN(A)          CONC          CONC
 1 H2O                4.33541D+01  -1.15415D+00   5.55084D+01   55.50837
 2 H+                 4.84053D-06  -1.19161D+01   6.19758D-06     .00001
 3 Mg++               3.97433D+00  -1.43630D+00   5.08853D+00    5.08853
 4 MgOH+              4.83007D-06  -1.69855D+01   6.18419D-06     .00001
 5 Ca++               8.00000D-01  -5.15081D+00   1.02428D+00    1.02428
 6 CaOH+              1.00864D-08  -2.35570D+01   1.29141D-08     .00000
 7 OH-                3.71690D-10  -2.14648D+01   4.75893D-10     .00000
 8 Cl-                9.54866D+00   9.35431D+00   1.22256D+01   12.22563
  AH2O=       .31532387  IONIC STRENGTH=      18.33844247 mol/kg

 9 MgCl2*6H2O         5.25669D-01   0.00000D+00   1.00000D+00    1.00000

 CONVERGENCE CRITERIA    2.810D-17   4.443D-10   0.000D+00
ITER=     6  IFUN=    12  IREFIN=     1


   STEP    5  SCALE FACTOR=  1.2200D-02
SPECIES                  MOLES        LN(A)          CONC          CONC
 1 H2O                3.43453D+01  -1.18360D+00   5.55084D+01   55.50837
 2 H+                 3.97916D-06  -1.21246D+01   6.43106D-06     .00001
 3 Mg++               2.97287D+00  -2.06687D+00   4.80471D+00    4.80471
 4 MgOH+              3.96736D-06  -1.74369D+01   6.41198D-06     .00001
 5 Ca++               9.00000D-01  -5.34106D+00   1.45457D+00    1.45457
 6 CaOH+              1.14569D-08  -2.35681D+01   1.85165D-08     .00000
 7 OH-                3.46996D-10  -2.12857D+01   5.60809D-10     .00000
 8 Cl-                7.74575D+00   9.75794D+00   1.25186D+01   12.51856
  AH2O=       .30617424  IONIC STRENGTH=      18.77783878 mol/kg

 9 MgCl2*6H2O         1.10659D+00   0.00000D+00   1.00000D+00    1.00000

 CONVERGENCE CRITERIA    4.291D-17   7.398D-08   0.000D+00
ITER=     3  IFUN=     3  IREFIN=     1
NORMAL TERMINATION OF EQUIL
```

SOME SOLUBILITY PREDICTIONS

Figures 7 through 11 compare various solubility curves as predicted by NAELS with experimental data from the literature. Qualitatively, NAELS has predicted the shape of the curves and the approximate position of the invariant points. Invariant points (labeled 'ip' on the graphs) are points where all of the phases that can coexist do coexist at equilibrium (i.e. the system has zero degrees of freedom). Quantitatively, only the invariant points will be compared. If the solubility estimate is perfect, then the experimental invariant points will be identical to those predicted by NAELS. The noninvariant points generated by NAELS were not intended to duplicate the experimental points; but rather, that the resulting curves should coincide. For the invariant points the error will be defined for each solid as follows:

$$\% \text{ error} = \frac{\text{NAELS solubility} - \text{experimental solubility}}{\text{experimental solubility}} \times 100\%$$

In Figure 7 points 2 and 3 are the invariant points as measured experimentally and as predicted by NAELS, respectively. The error is 47% with respect to KCl and 3% with respect to $CaCl_2 \cdot 6H_2O$. In the region between point 1 and the invariant points, only KCl forms as a precipitate. In the region between point 4 and the invariant points, only $CaCl_2 \cdot 6H_2O$ forms. Qualitatively, the agreement is good away from the invariant point. In this type of situation the predicted invariant point could be moved closer to the experimental point by adjusting the value of the Pitzer parameter $\theta_{K^+-Ca^+}$ until the most satasfactory fit is achieved. This approach is commonly practiced when attempting to model a particular electrolyte system. It is not done here because the intent is for NAELS to be a generally applicable simulator.

In the region between points 5 and 6 in Figure 8, the solid formed is $MgCl_2 \cdot 6H_2O$. Point 6 represents an invariant point both experimentally and as predicted by NAELS. The error is 0.7% on the Mg axis and 2% on the Ca axis. In the region between points 6 and 7 (or 8), only the compound salt $CaCl_2 \cdot 2MgCl_2 \cdot 12H_2O$ forms. Point 7 is the invariant point for the formation of the complex salt together with $CaCl_2 \cdot 6H_2O$ as predicted by NAELS. Point 8 is the corresponding experimental point. The error is 12% on the Mg axis and 7% on the Ca axis. Below point 8, only $CaCl_2 \cdot 6H_2O$ forms.

In Figure 9 the regions to the left of the invariant points are where the hydrated salt $Na_2SO_4 \cdot 10H_2O$ forms. The invariant points are where both the anhydrate and the hydrated salt coexist. To the right of the invariant points the anhydrate salt exists alone until point 9 is reached, where the solid $NaOH \cdot H_2O$ precipitates. The invariant points are off by 30 % for Na_2SO_4 and 18% for NaOH. The interesting point to notice about this graph is the similarity in the shapes of the curves. The curious vertical deflection of the curves appears to be due, at least in part to a discrepancy in the pure salt solubility of $Na_2SO_4 \cdot 10H_2O$. The curves should start from the same point on the Na_2SO_4 axis. The fact that they do not indicates that the experimental data would lead to a different value for the standard free energy of formation for $Na_2SO_4 \cdot 10H_2O$ than the one used to generate the NAELS curve in Figure 9. This discrepancy could have been corrected to yield a better fit (in practice the single salt solubilities should be consistent with the experimental data). However, the procedure used in this work was to use a single salt solubility that was measured independently of the system of comparison. Adjusting the standard free energy for $Na_2SO_4 \cdot 10H_2O$ would yield a much better fit in this case.

In Figure 10 the Na_2CO_3-Na_2SO_4 system predictions are shown. The region to the left of the invariant points is the solubility curve for $Na_2SO_4 \cdot 10H_2O$ while the region below is for $Na_2CO_3 \cdot 10H_2O$. The error in the invariant points is 7% on the Na_2CO_3 axis and 8% on the Na_2SO_4 axis.

In Figure 11 NaCl is the solid formed above the invariant points while $Na_2CO_3 \cdot H_2O$ is formed below the invariant points. The invariant point error is 0.2% on the Na_2CO_3 axis and 14% on the NaCl axis.

Qualitatively, these are good estimates of solubility. They are representative of the type of predictions that can be made with NAELS. They would doubtless be better if more of the Pitzer ion interaction parameters were known for the many possible combinations of ions found in these systems. Whether or not the predictions are sufficiently accurate is a question that must be answered by the individual user.

Figure 7

$CaCl_2$—KCl solubility at 25°C

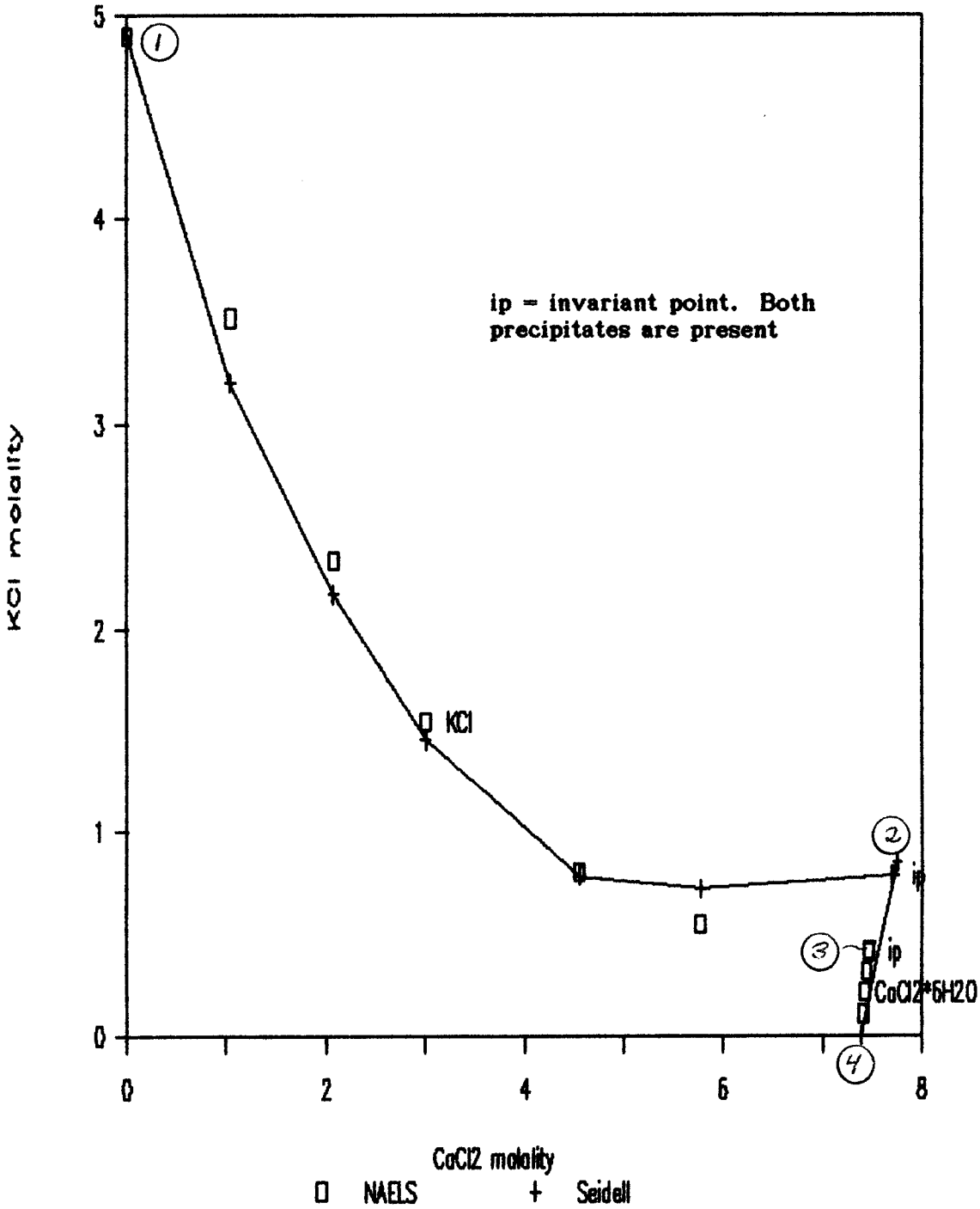


Figure 8

$CaCl_2$—$MgCl_2$ solubility at 25°C

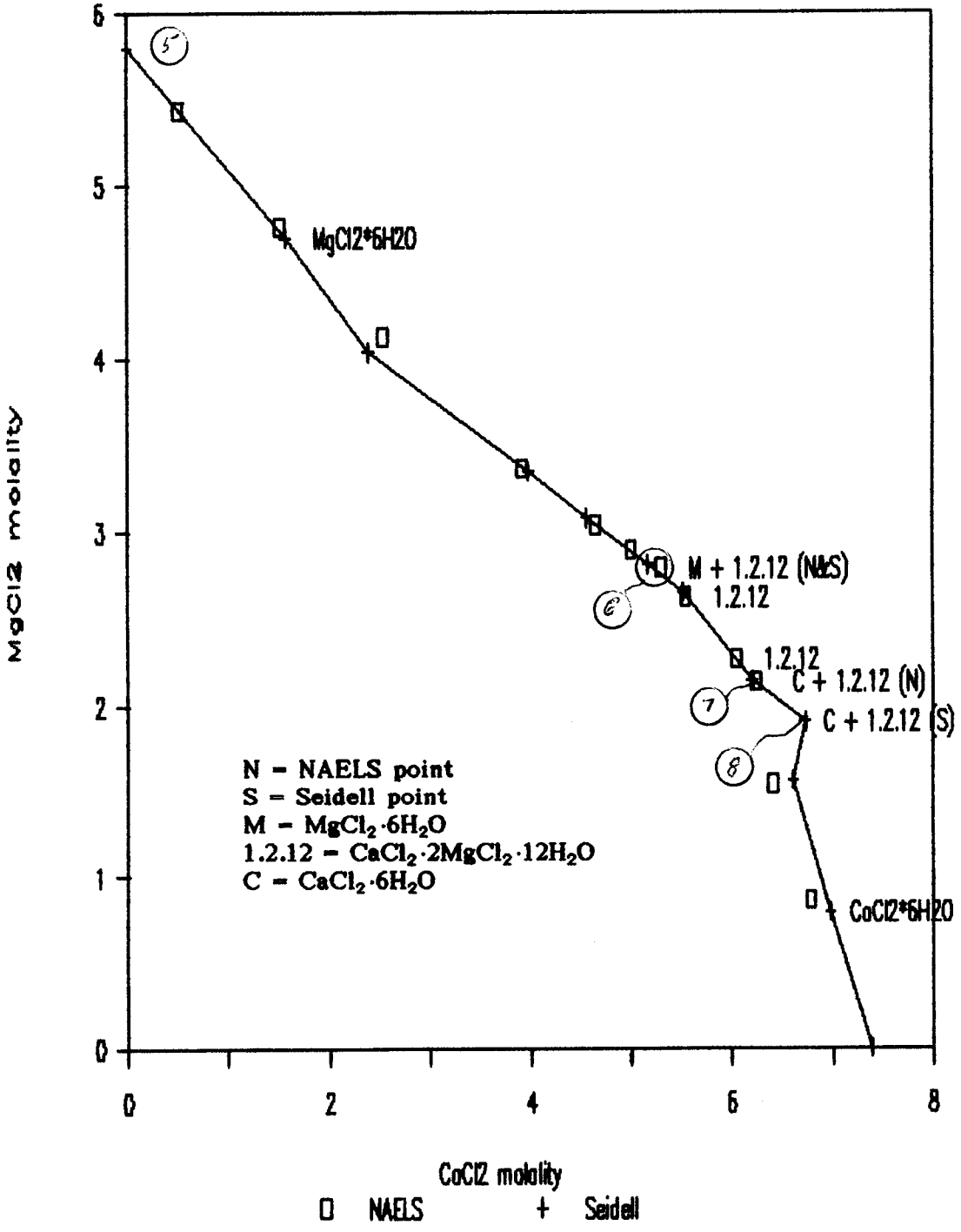


Figure 9

NaOH—Na_2SO_4 solubility at 25°C

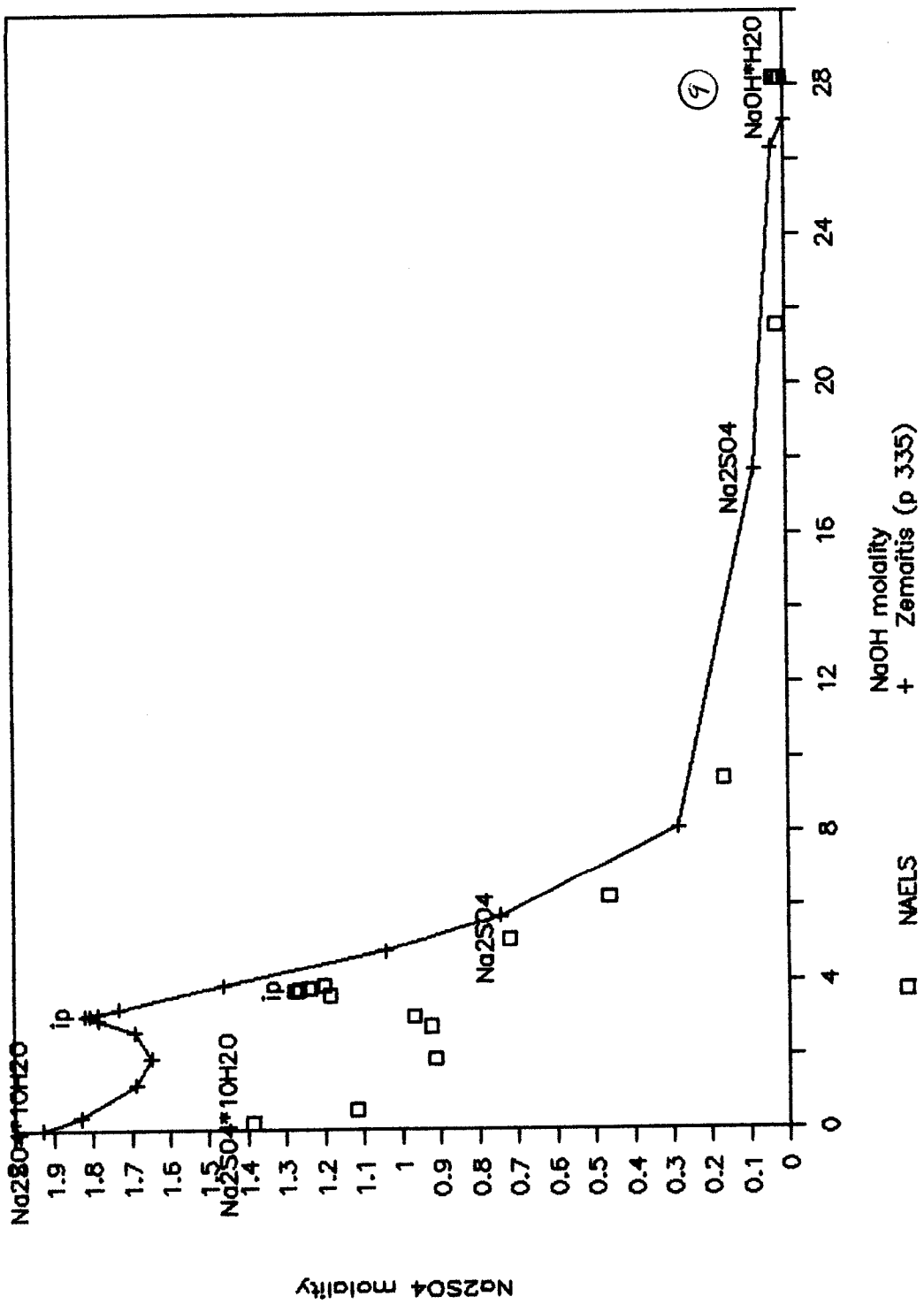


Figure 10

Na_2CO_3—Na_2SO_4 solubility at 25°C

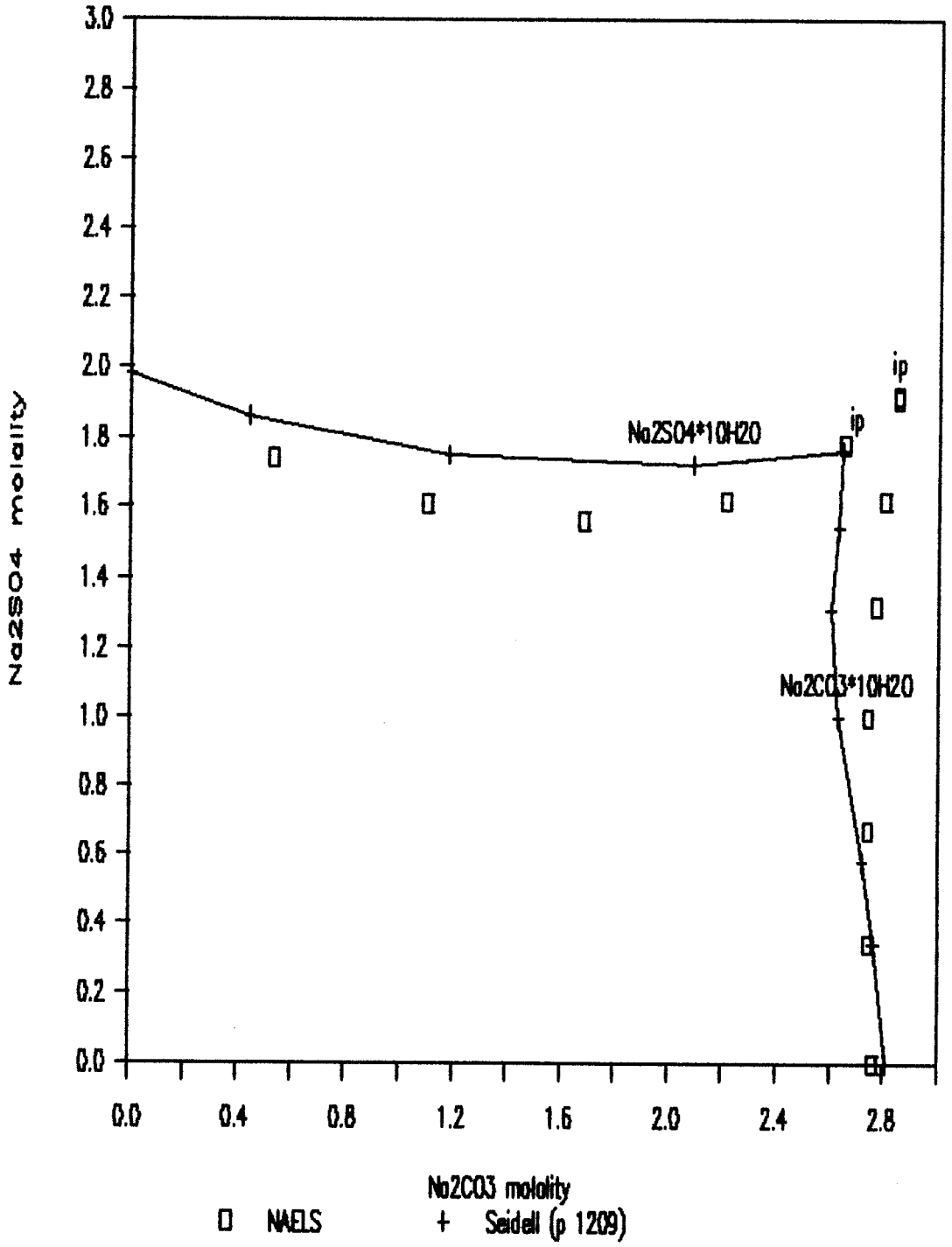
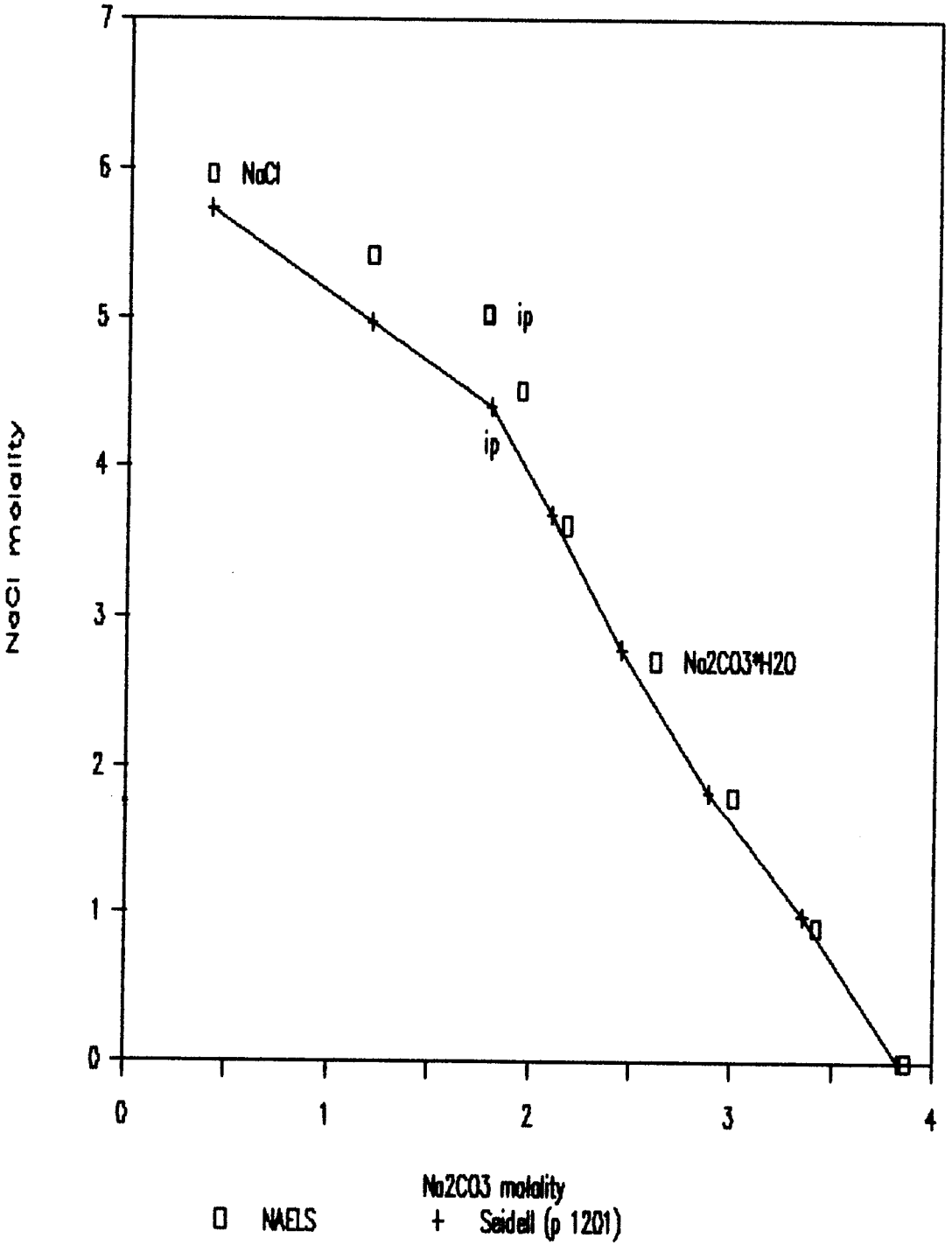


Figure 11

NaCl—Na_2CO_3 solubility at 40°C



LIMITATIONS AND SOURCES OF ERROR

The major drawback associated with using a free energy minimization approach to solving equilibrium problems is the need for very accurate data. Wagman, et al., lists only one value of ΔG°_{f} for each species (i.e. the value that the authors feel is the from the most reliable source). Woods and Garrels make no such judgements and present thermodynamic data from as few as one to as many as a dozen or more sources for the more common species. It has been said that a man with one watch always knows what time it is while a man with two is never sure. Such is the case in solubility predictions. Discrepancies of a few percent in standard free energy values can translate into much larger discrepancies in solubility predictions. The three solibility curves in Figure 12 were generated using three of the standard free energy values for KCl at 25°C found in Woods and Garrels. The same standard free energy for NaCl was used throughout. Note how the small variations in ΔG°_{f} lead to relatively large variations in predicted solubility. The predicted locations of the invariant points (circled) are also affected.

Figure 12 illustrates why it is better to start with experimental solubility data and back-calculate what the chemical potential of the precipitate should be to produce a good fit. NAELS can easily do this. Linke (1965) is an excellent source of solubility data. Whatever the souce, solubility data is usually in the form of mass of dissolved solute per mass of solvent (or per mass of solution). This must be converted to units of molality. The procedure is best illustrated with an example. Assume that the vanadate ion was allready in the database but the solid was a new addition. Consider the formation of sodium vanadate:

$$Na^{+}(aq) + VO_3^{-}(aq) \Leftrightarrow NaVO_3(s) \qquad (22)$$

At equilibrium the sum of the chemical potentials on each side of Equation 22 must be equal. If Equation 1 is rearranged and written for each specie, the result will be

$$\frac{\mu^{\circ}_{Na^+}}{RT} + \ln(a_{Na^+}) + \frac{\mu^{\circ}_{VO_3^-}}{RT} + \ln(a_{VO_3^-}) = \frac{\mu^{\circ}_{NaVO_3}}{RT} + \ln(a_{NaVO_3}) \qquad (23)$$

The activity of the solid is taken as unity and all four terms on the left hand side of Equation 23 can be found in the output file of NAELS. The dimensionless standard chemical potential of the solid is the only unknown. Seidell reports the solubility of sodium vanadate at 25°C to be 21.10 grams per 100 grams H_2O. Its molecular weight is 121.93. Therefore the saturation concentration of both Na^+ and VO_3^- is 1.7305 mol/kg. This concentration is then used in the input file for both Na^+ and VO_3^-. H^+, OH^- and H_2O are included in the input list and no solids are listed. Equation 23 must hold at saturation equilibrium. NAELS will calculate the equilibrium activities on the left hand side of Equation 23 (the standard chemical potentials are already known). Executing NAELS produces the following values, respectively, for the terms in Equation 23:

$$(-105.656)+(-0.0375964)+(-316.115)+(-0.498978)=\frac{\mu^{\circ}_{NaVO_3}}{RT}+0.0$$

$$\frac{\mu^{\circ}_{NaVO_3}}{RT}=-422.308 \quad \text{dimensionless}$$

This number can now be added to the database as the dimensionless standard chemical potential for sodium vanadate (C_0 in Equation 17). If data is available at several temperatures, then the procedure can be repeated at each temperature to generate values which can then be fit to a polynomial of the form of Equation 17.

A word of caution is in order with respect to fitting chemical potential data to a polynomial. Whether the chemical potential data to be fitted comes from Equation 21 or from solubility data, some data sets turn out smoother than others. A utility called 'TK Solver' was used for all polynomial fitting in this project. The regression coefficient, R^2, exceeded 0.9998 for all of the polynomials in the database (0.99999 was typical). Once the polynomial has been fit is is wise to check the discrepancy between the data points and the corresponding functional values. If the fit is unacceptable, then a new polynomial should be fit over a smaller 'temperature range of interest'.

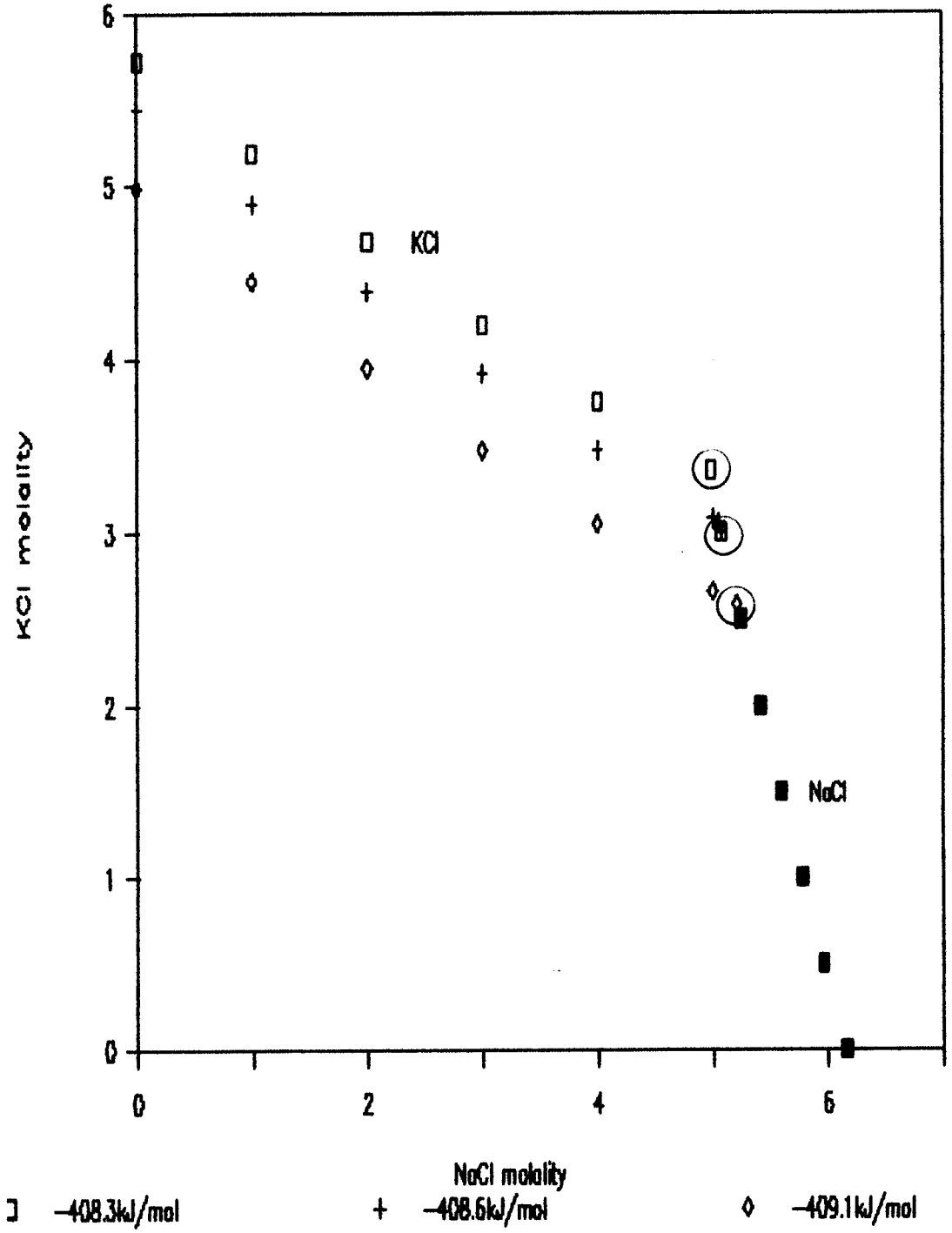
The lack of Pitzer parameters is another potential source of error in solubility predictions. It is obvious from a quick look at the Pitzer section of the database that the majority of the parameters are zero. The importance of Pitzer parameters varies with the concentration of the species in question.

For a dilute species the Pitzer parameters will never be missed if they are not available. The resulting product of the Pitzer parameters and the low molality is negligable compared to the corresponding terms for the major species in Equations 8. For the major species in a solution it is good to have them if they have been measured. The work of Pitzer and coworkers has been going on for decades but when one looks at the number of ions that exist, and the number of pairs and triplets that result, then the lack of data is not surprising. The amount of Pitzer data in the database does not represent an exhaustive search of the literature. Frederick and Kim (1988), Zemaitis, et al. (1986), and Weare, et al. (1987) were sources used in this work. A thorough search of the literature with emphasis on Pitzer would doubtless provide a good deal of parameters to add to the database.

With regard to systems at temperatures other than 25°C, the ion interaction parameters are assumed constant while A^{ϕ} is a function of temperature. In reality Pitzer ion interaction parameters do vary with temperature. Pitzer (1979) gives derivatives with respect to temperature at 25°C for a few species. The slopes are very small at 25°C. While it would be safe to use these slopes to extrapolate parameters near 25°C, it is not sound to use a derivative at 25°C to extrapolate to a value at 100°C. Due to the small number of species for which temperature derivatives are available, and the fact even those are only known at 25°C, the Pitzer parameters are assumed constant over the range 25-100°C.

Figure 12

KCl—NaCl solubility
dependence on KCl standard free energy



CONCLUSIONS

In the course of this thesis an aqueous electrolyte simulator was assembled that meets all of the criteria set forth in the introduction.

It has never failed to converge to a solution in the course of this work even though the ionic strength frequently exceeded 30 molal. The time required to converge varied from under one minute to four minutes on an IBM compatable AT using an INTEL 80286 processer operating at 12 MHz. At the time of this writing this is not considered to be a very powerful computer.

Non-ideality, in any aqueous system, is greatest at saturation. At saturation the ionic strength is high and precipitates are in equilibrium with the aqueous phase. Therefore, solubility predictions provide the most challenging test for a program such as NAELS. Qualitatively, the solubility diagrams predicted by NAELS agreed well with the experimental data from the literature. NAELS predicted the overall shapes of the experimental solubility diagrams and, in most cases, closely predicted the locations of the invariant points.

A large, user expandable database was assembled so as to minimize the amount of input required on the part of the user. Unfortunately, at the time of this writing, Pitzer ion interaction parameters have only been measured for a few dozen of the more common ionic species. Even fewer of the ternary (ion triplets) mixing parameters have been measured. As time goes on, hopefully the volume of Pitzer data will increase.

FUTURE WORK

NAELS could be improved by expanding the database. Solubility data, preferably over a range of temperatures, could be used to add solid species to the database (the procedure is outlined in this work). Solution species can be added to the database with a trial and error procedure using solubility data. Solution species can be added directly if the standard free energy of formation is known.

A literature search of Pitzer ion interaction parameters would doubtless yield a number of them not found during the course of this work. Following a literature search, the experimental measurement of Pitzer ion interaction parameters for the user's system(s) of interest would be fruitful.

The capability of modeling a closed gas-liquid-solid system would make NAELS a more powerful utility. A closed system would require a model for predicting the activity coefficients of the gaseous species. As gaseous species enter and leave the solution phase, their partial pressures change. The total pressure is the sum of the partial pressures. If the total system pressure is to remain constant, then the system volume must be allowed to vary. If the volume is fixed, then the total pressure will vary. Since solubility is largely pressure dependent, a closed system is more complex to model. Constant pressure and constant volume closed systems could be two additional modeling capabilities for NAELS.

REFERENCES

Beyer, R.P. and Staples, B.R., Journal of Solution Chemistry, 9(15) (1986)

Connick, R.E. and Powell, R.E., Journal of Chemical Physics, 21(12) (1953)

Criss, C.M. and Cobble, J.W., Journal of the American Chemical Society, 86, 5385 and 5390 (1964)

Davies, C.W., Ion Association, Butterworths Scientific Publications, London (1962)

Frederick W.J., and Kim, H.T., Journal of Chemical Enginering Data, 33(2),(1988)

Frederick W.J., and Kim, H.T., Journal of Chemical Enginering Data, 33(3),(1988)

Linke, W.F., Solubilities: Inorganic and Metal-Organic Compounds—A Compilation of Solubility Data from the Periodical Literature. Vol. I: A-I, D. Van Nostrand Co. Prinston N.J., 1958, Vol. II: K-Z, American Chemical Society, Washington, D.C., 1965

Pitzer, K.S. and Mayorga, G., Journal of Physical Chemistry, 77(19), 2300 (1973)

Pitzer, K.S. and Mayorga, G., Journal of Solution Chemistry, 3, 539 (1974)

Pitzer, K.S., Journal of Physical Chemistry, 77(2), 268 (1973)

Pitzer, K.S. and Kim, J.J., Journal of the American Chemical Society, 96(18), 5701 (1974)

Pitzer, K.S., Theory: Ion interaction Approach. Activity Coefficients in Electrolyte Solutions, vol. 1 R.M. Pytkowicz, ed., CRC Press (1979)

Seidell, A., Solubilities of Inorganic and Metal Organic Compounds 3rd ed. v. II D. Van Nostrand Co. New York (1940)

Wagman, D.D., et al., Journal of Physical and Chemical Reference Data, 11 sup 2 (1982)

Weare, J.H., Greenberg, J.P., and Harvie, C.E. Geochimica et Cosmochimica Acta, 51, 1045 (1987)

Weare, J.H., Moller, N., and Harvie, C.E. Geochimica et Cosmochimica Acta, 48, 723 (1984)

Weare, J.H., Reviews in Minerology, vol. 17, Mineralogical Society of America (1987)

Woods, T.L. and Garrels, R.M., Thermodynamic Values At Low Temperature for Natural Inorganic Materials, Oxford University Press (1987)

Zemaitis, J.F., et al., Handbook of Aqueous Electrolyte Thermodynamics American Institute of Chemical Engineers (1986)

APPENDICES

APPENDIX A

Fortran language and hardware considerations

NAELS is written in the FORTRAN 77 computer language and compiled with the MICROSOFT FORTRAN compiler (version 4.1). There are 5 files associated with NAELS: The executable file NAELS.EXE, the database THERMO.DAT, the input file SALT, the output file SOLUTION, and the database listing program DBLIST.EXE. NAELS requires approximately 580 K of free RAM to run.

The user should consult the Microsoft manuals for technical support with regard to the compiler or linker. The options chosen for compiling this work are as follows:

/G2–selects the 80286 processor instruction set.

/Os–optimizing procedures favor smallest code size.

/Ge–enables stack probes to check avaiable stack space.

/FPi–generates in-line instructions and selects the emulator math package

Once the individual subroutines were compiled they were linked using the MS overlay linker. When a program is overlaid, specified parts of the program share the same space in RAM memory and are only loaded if and when they are needed. This reduces the required amount of free RAM necessary to run the program. NAELS was overlaid in the fashion shown below. The subroutines in parentheses are overlays while the remaining subroutines constitute the resident part of the program.

PRIMAL+DUAL+(MAIN+DATA+DATAP)+(CONADD+CONDEL)
+(REFINE)+(GRAD+HESS+ELECT)+(DADD+DDEL)+(PHSPRT)

APPENDIX B

List of species in database

THE FOLLOWING IS A LIST OF THE CHEMICAL SPECIES FOUND IN THE DATABASE "THERMO.DAT", ALONG WITH THEIR RESPECTIVE I.D. NUMBERS, AND THE TEMPERATURE RANGES OVER WHICH THE FREE ENERGY POLYNOMIAL IS VALID. AN ASTERISK INDICATES THAT THE FREE ENERGY DATA WERE CALCULATED USING STANDARD STATE DATA FOUND IN THE LITERATURE. THE ABSENSE OF AN ASTERISK INDICATES THAT THE FREE ENERGY DATA WERE CALCULATED FROM PUBLISHED SOLUBILITY DATA. CONSULT THE USERS MANUAL FOR MORE INFORMATION.

99 H2O		100 H+	25-200
101 Na+	25-200	102 K+	25-200
103 Mg++	25-200	104 MgOH+	25
105 MgHCO3+	25	106 Ca++	25-200
107 CaOH+	25	108 CaHCO3+	25
109 Al+++	25-200	110 AlOH++	25
111 Al(OH)2+	25	112 Mn++	25-200
113 MnOH+	25-200	200 OH-	25-200
201 Cl-	25-200	202 CO3--	25-200
203 HCO3-	25-200	204 NaCO3-	25
205 SO4--	25-200	206 HSO4-	25-200
207 NaSO4-	25	208 KSO4-	25
209 Al(OH)4-	25-200	210 S--	25-200
211 HS-	25-200	212 H3SiO4-	25
213 PO4---	25-200	214 HPO4--	25-200
215 H2PO4-	25-200	216 VO3-	25-200
300 H2CO3(aq)	25	301 NaHCO3(aq)	25
302 MgCO3(aq)	25	303 MgSO4(aq)	25
304 CaCO3(aq)	25	305 CaSO4(aq)	25
306 H4SiO4(aq)	25	307 H3PO4(aq)	25
400 SiO2	25-200*	401 Al2O3	25-200*
402 Al(OH)3	N/A	403 AlCl3	25-200*
404 AlCl3*6H2O	25-200*	405 Al2(SO4)3	25-200*
406 Al2(SO4)3*6H2O	25-200*	407 Al2SiO5	25-200*
408 Al2Si2O7*2H2O	25-200*	409 Al6Si2O13	25-200*
410 Al2Si4O10(OH)2	25-200*	411 Mn(OH)2	25-200*
412 MnCl2	25-200*	413 MnCl2*H2O	25-200*
414 MnCl2*2H2O	25-200*	415 MnCl2*4H2O	25-200*
416 MnS	25-200*	417 MnSO4	25-200*
418 MnHPO4	25*	419 MnCO3	25-200*
420 MnSiO3	25-200*	421 Mn2SiO4	25-200*
422 Mg(OH)2	25-200*	423 MgCl2	25-200*
424 MgCl2*H2O	25-200*	425 MgCl2*2H2O	25-200*
426 MgCl2*4H2O	25-200*	427 MgCl2*6H2O	25-200*
428 MgS	25-200*	429 MgSO4	25-200*
430 MgSO4*H2O	25-200*	431 MgSO4*6H2O	25-200*
432 Mg3(PO4)2	25-200*	433 MgCO3	25-200*
434 (MgCO3)3*Mg(OH)2*3H2O	25*	435 Ca(OH)2	25-200*
436 CaCl2	25-200*	437 CaS	25-200*
438 CaSO4	25-200*	439 CaSO4*2H2O	25-200*

440	Ca3(PO4)2	25-200*	441	CaHPO4	25-200*
442	CaHPO4*2H2O	25-200*	443	Ca(H2PO4)2*H2O	25-200*
444	Ca10(PO4)6(OH)2	25-200*	445	CaCO3	25-200*
446	CaO*SiO2	25-200*	447	CaO*Al2O3	25-200*
448	CaO*Al2O3*2SiO2*H2O	25-200*	449	CaCO3*MgCO3	25-200*
450	(CaO)2*MgO*2SiO2	25-200*	451	NaOH	25-200*
452	NaOH*H2O	25-64	453	NaCl	25-100
454	Na2S	25-200*	455	Na2SO4	25-200*
456	NaHSO4	25-200*	457	NaHSO4*H2O	25-200*
458	Na2SO4*10H2O	25	459	Na3PO4	25-200*
460	NaH2PO4	25-200*	461	Na2HPO4	25-200*
462	Na2HPO4*2H2O	25-200*	463	Na2HPO4*7H2O	25-200*
464	Na2HPO4*12H2O	25-200*	465	Na2CO3	25-200*
466	Na2CO3*H2O	25-200*	467	Na2CO3*7H2O	25-200*
468	Na2CO3*10H2O	20-32	469	NaHCO3	25
470	NaHCO3*Na2CO3*2H2O	25	471	Na2SiO3	25-200*
472	NaAlSiO4	25-200*	473	NaVO3	25-75
474	KOH	25-200*	475	KOH*2H2O	25-200*
476	KCl	25-200*	477	K2S	25-200*
478	K2SO4	25-200*	479	KHSO4	25-200*
480	K2CO3	25-200*	481	KHCO3	25-200*
482	KAl(SO4)2	25-200*	483	KAl(SO4)2*3H2O	25-200*
484	KAl(SO4)2*12H2O	25-200*	485	KAlSiO4	25-200*
486	KAl3Si3O10(OH)2	25-200*	487	KH2PO4	25-200*
488	2Na2SO4*Na2CO3	25*			

APPENDIX C

Ionic heat capacity estimation

Partial molar heat capacity data for ions in solution is usually available only at 25°C, if at all. In general, heat capacities for solution species vary considerably with temperature. It is therefore expedient to make use of a model for the estimation of ionic heat capacities as a function of temperature. The model begins with the relationship between entropy and heat capacity:

$$S^\circ_t = S^\circ_{t_0} + \int_{t_0}^{t} C_p \, d(\ln T) \tag{C-1}$$

C_p can be replaced with an average value over the range t_0 to t:

$$S^\circ_t = S^\circ_{t_0} + \overline{C_p}|^t_{t_0} \int_{t_0}^{t} d(\ln T) \tag{C-2}$$

Integrating and rearranging yields,

$$\overline{C_p}|^t_{t_0} = \frac{S^\circ_t - S^\circ_{t_0}}{\ln(t/t_0)} \tag{C-3}$$

t_0 and t can be any two temperatures. For convenience t_0 will be the standard reference temperature 298.15 K. Standard state entropies of aqueous species are usually available in the literature. Wagman (1982), and Woods and Garrels (1987) are excellent references. The only remaining unknown in Equation C-3 is S°_t. The correspondence principal developed by Criss and Cobble (1964) states, "...If the standard state is chosen properly by fixing the entropy of H^+(aq) at each temperature, then the ionic entropies at one temperature are linearly related to their corresponding entropies at 25°C". In equation form the model looks like this:

$$S^\circ_t = a(t) + b(t) S^\circ_{25abs} \tag{C-4}$$

where

$$S^\circ_{25abs} = S^\circ_{25} - 5.0Z \tag{C-5}$$

S°_{25} in Equation C-5 is the conventional standard state entropy and Z is the charge on the ion. a(t) and b(t) in Equation C-4 are functions of temperature and ion type. Criss and Cobble's values for a(t) and b(t) are presented in Table 1. S°_t is then used in Equation C-3 to obtain a partial molar heat

capacity over a particular temperature range.

TABLE 1
Criss and Cobble's entropy parameters
(in CAL/MOLE*°C)

	Simple cations		Simple anions and OH^-		Oxy anions XO_n^{-m}		Acid oxy anions $XO_n(OH)^{-m}$		
t,°C	a(t)	b(t)	a(t)	b(t)	a(t)	b(t)	a(t)	b(t)	$S^\circ_{abs}(H^+)$
25	0	1.000	0	1.000	0	1.000	0	1.000	-5.0
60	3.9	0.955	-5.1	0.969	-14.0	1.217	-13.5	1.38	-2.5
100	10.3	0.876	-13.0	1.000	-31.4	1.476	-30.3	1.894	2.0
150	16.2	0.792	-21.3	0.989	-46.4	1.687	-50.	2.381	6.5
200	23.3	0.711	-30.2	0.981	-67.	2.02	-70.	2.960	11.1

Criss and Cobble's model, in its original form, is of limited utility. In order to calculate the chemical potential as a continuous function of temperature, one must be able to calculate the heat capacity in Equation C-3 as a continuous function of temperature. It turns out that the entropy parameters in Table 1 are themselves linear functions of temperature. Performing linear regression on the parameters in Table 1 produces the equations in Table 2, where T is in degrees celcius.

TABLE 2
Entropy parameters
as a function of temperature

	Regression coeff. R^2
Simple cations	
$a = 0.133886\ T - 3.58581$	0.991
$b = -0.00168537\ T + 1.04717$	0.997
Simple anions and OH^-	
$a = -1.174220\ T + 4.72151$	0.998
$b = -0.00018599\ T + 1.0077$	0.952
Oxy anions	
$a = -0.377372\ T + 8.69877$	0.997
$b = 0.00569148\ T + 0.871012$	0.994
Acid oxy anions	
$a = -0.401098\ T + 10.1575$	0.999
$b = 0.0111676\ T + 0.728066$	0.998
S°_{25} for H^+	
$a = 0.09416006\ T - 7.635905$	0.995

Combining the equations in Table 2 with Equations C-4 and C-5 produces a partial molar heat capacity function which is used to calculate the chemical potential as a function of temperature for charged aqueous species. Since the reference state for $H^+(aq)$ in this model is a function of temperature, the chemical potential function for any charged species added to the database must be generated using the procedure outlined above. This is necessary to maintain interal consistency in the database. However, if an aqueous species can be formed by combining species allready found in the database (ie. neutral salt species), then it is best to use equilibrium data (if available) to generate the chemical potential function for the new species.

Criss and Cobbles method requires that the standard state entropy, S°_{25}, be known. For many ions this quantity is not available in the literature. One method of estimating S°_{25} is that of Connick and Powell (1953). The entropy of an oxy-anion of the form XO_n^{-z} can be estimated fairly well from the empirical relation

$$S^\circ_{25} = 43.5 - 46.5(Z - 0.28n) \qquad \frac{\text{cal}}{\text{mol } ^\circ\text{C}}$$

Once again, a literature search would be in order as there are other methods for estimating entropies.

APPENDIX D

Nomenclature and glossary

α_1 Pitzer parameter equal to 1.4 for 2-2 electrolytes and 2.0 for all others.

α_2 Pitzer parameter equal to 12.0 for 2-2 electrolytes and 0.0 for all others.

β^0_{ca} Pitzer empirical parameter for the salt c-a.

β^1_{ca} Pitzer empirical parameter for the salt c-a.

β^2_{ca} Pitzer empirical parameter for the 2-2 salt c-a.

γ_i Activity coefficient for ion i (dimensionless).

η_i Lagrangian multiplier for charge balance constraint i.

$\theta_{cc'}$ Pitzer ion interaction parameter for unlike cation pairs (empirical).

$\theta_{aa'}$ Pitzer ion interaction parameter for unlike anion pairs (empirical).

κ_i Lagrangian multiplier for mass balance constaint i.

μ_i Chemical potential of (or Gibb's energy per mole of) species i.

μ°_i Standard state chemical potential of species i.

π Pi

$\psi_{caa'}$ Pitzer ternary interaction parameter for unlike ion triplets (empirical).

$\psi_{acc'}$ Pitzer ternary interaction parameter for unlike ion triplets (empirical).

ω_i Lagrangian multiplier for species constraint i.

A_{ij} Number of moles of element i in species j.

a_i Activity of species i.

A^ϕ Debye-Hückel constant for osmotic coefficients on a natural log basis.

b_i Number of moles of element i.

b Pitzer parameter equal to 1.2

C_{ca} Pitzer empirical parameter for the salt c-a.

C_p Heat capacity at constant pressure.

D Dielectric constant of water (temperature dependent).

e_c Number of electrolyte phases.

e Electronic charge. 1.60×10^{-19} coulombs

G Total system (Gibb's) free energy.

ΔG°_f Standard state free energy of formation

ΔG_{rxn} Free energy change in a chemical reaction.

H Enthalpy

I Ionic strength (mole/kg).

k Boltzman's constant (1.381×10^{-23} joule/kelvin)

L Lagrangian objective function to be minimized

m_i molality of species i (moles/kg).

m_c number of mass balance constraints

n_i mole number of species i (moles).

n^t total number of chemical species in the system.

N_o Avagadro's number.

R Gas constant (8.314 joules/mole*kelvin).

S Entropy

S°_{25} Standard state entropy (joules/mole*kelvin).

T Absolute temperature (kelvin).

t Temperature (celcius).

T_{ref} Reference temperature (298.15 kelvin).

t_i^2 Slack variable for the non-zero mole number constraint for species i.

z_i Charge of species i.

Species: A compound made up of elements and formed by one or more equilibrium reactions. It may be charged or neutral and it may exist in any phase. For example, CO_2(gas) and CO_2(aq) are two different species.

Phase assemblage The number of phases present in a system at equilibrium as well as the composition of each phase. The solution to an equilibrium problem.

Element: A chemical element as found in the periodic table.

Empirical: Measured experimentally

APPENDIX F

NAELS source code

```
C-----------------------------------------------------------------
C
C       THE MAIN PROGRAM. THIS PROGRAM CALCULATES CHEMICAL
C       EQUILIBRIUM BY MINIMIZING THE GIBBS FREE ENERGY.
C       INDEPENDENT REACTIONS DIRECTIONS ARE CALCULATED
C       THROUGH A QR FACTORIZATION OF THE CONSTRAINT MATRIX. A
C       SET OF VARIABLES ARE THUS GENERATED THAT CORRESPOND
C       TO THESE REACTION DIRECTIONS. SINCE THE PROGRESS TO
C       EQUILIBRIUM THAT IS DEFINED BY THESE VARIABLES DO NOT
C       VIOLATE ANY CONSTRAINTS, THE PROBLEM HAS BEEN
C       TRANSFORMED INTO AN UNCONSTRAINED MINIZATION PROBLEM.
C       NEWTONS METHOD IS USED TO CONVERGE TO A SOLUTION.  THE
C       LAGRANGIAN MULTIPLIERS WHICH CORRESPOND TO THE
C       NEGATIVE NATURAL LOGARITHM OF THE SATURATION RATIO
C       ARE THEN EVALUATED IN ORDER TO DETERMINE WHETHER OR
C       NOT OTHER PHASES SHOULD COME INTO THE SYSTEM. THE
C       SEQUENCE OF NEWTON ITERATIONS AN MULTIPLIER
C       EVALUATIONS IS CONTINUED UNTIL GLOBAL CONVERGENCE IS
C       ACHIEVED.
C
C   CURRENT DIMENSIONS:
C          A MAXIMUM TOTAL OF 96 SPECIES IN ALL INPUT PHASES,
C          A MAXIMUM OF 60 SOLUTION SPECIES IN ALL SOLUTION
C          PHASES, A MAXIMUM TOTAL OF 46 AQUEOUS SPECIES: 18
C          MAXIMUM CATIONS, 18 MAXIMUM ANIONS, AND 8 MAXIMUM
C          NEUTRALS. (NOTE THAT ONE SPECIES MUST BE H2O).
C
C
C-----------------------------------------------------------------
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 Z
      LOGICAL ERROR,CHARGE,BLANK,OUTFIL
      DIMENSION DEL(96),BB(96),U0(96),JH(96),PF(96),JFRAX(96)
      CHARACTER*24 A1
      CHARACTER*50 WATLAB
C-----------------------------------------------------------------
C
C       ERROR: IF PRIMAL RETURNS ERROR = .TRUE., THE PROBLEM HAS
C            NOT CONVERGED OR HAS CONVERGED TO A NON-CONVEX
C            SOLUTION.
C
C       CHARGE: = .TRUE. IF THE PROGRAM HAS TO MAKE A CHARGE
C                BALANCE CORRECTION
C
C-----------------------------------------------------------------
      COMMON/CONST/Q(96,96),MACT(96),JA(96),JI(96),NSC,NCC,NDC,NC0C
     1, N0EC
C-----------------------------------------------------------------
C
```

```
C       Q(I,I):  NON-DEGENERATE MASS, CHARGE AND SPECIES
C                CONSTRAINT MATRIX
C
C       MACT(J)=I : VECTOR THAT MAPS THE SPECIES NUMBER J  TO
C            EITHER CONSTRAINT NUMBER I IF THAT SPECIES IS
C            CONSTRAINED OR DEGENERATE, OR TO A NEGATIVE NUMBER
C            IF THE SPECIES IS UNCONSTRAINED (I).  THE VALUE OF A
C            POSITIVE I ALSO CORRESPONDS TO AN ELEMENT IN THE JA
C            ARRAY WHEN THE CONSTRAINT CORRESPONDS TO A SPECIES
C            CONSTRAINT.  THE ABSOLUTE VALUE OF THE NEGATIVE
C            NUMBER CORRESPONDS TO THE ELEMENT OF THE JI ARRAY
C            THAT IS EQUAL TO THAT SPECIES NUMBER.
C
C       JA(J)=I: VECTOR THAT MAPS CONSTRAINT NUMBER J  TO
C            SPECIES NUMBER I.  IF J IS LESS THEN OR EQUAL TO NC0C,
C            THE NUMBER OF NON-DEGENERATE MASS AND CHARGE
C            BALANCE CONSTRAINTS, THE SPECIES I IS DEGENERATE. THE
C            NUMBER OF DEGENERATE CONSTRAINTS CANNOT BE MORE
C            THEN NC0C. REASON: THE TOTAL NUMBER OF SPECIES IS
C            NSC. THEREFORE THE MAXIMUM NUMBER OF (DEGENERATE
C            AND NON-DEGENERATE) SPECIES CONSTRAINTS IS NSC.
C            THE MAXIMUM NUMBER OF NON-DEGENERATE SPECIES
C            CONSTRAINTS IS NCC-NC0C (TOTAL LINEARLY INDEPENDENT
C            CONSTRAINTS MINUS THE CHARGE AND MASS BALANCE
C            CONSTRAINTS). THEREFORE THE MAXIMUM NUMBER OF
C            DEGNERATE CONSTRAINTS
C
C       JI(J)=I: VECTOR THAT MAPS THE JTH INACTIVE SPECIES TO
C            UNCONSTRAINED, NON-DEGENERATE SPECIES I
C
C-----------------------------------------------------------------------
C
C        EXAMPLES:
C
C           AN AQUEOUS SYSTEM WITH 20 AQUEOUS SPECIES AND 20
C           PURE EVAPORITES.  SPECIES 24 HAS PRECIPITATED AND
C           ONLY IT AND THE SOLUTION PHASE ARE PRESENT:
C
C           JI(21) = 24
C           MACT(24) = -21
C
C           ASSUME NOW THAT SPECIES 24 COMPLETELY DISSOLVES AND
C           THAT THE SYSTEM HAS 9 MASS AND CHARGE BALANCE
C           CONSTRAINTS. WE THEN HAVE 9 + 20 (SOLID SPECIES HELD
C           OUT) = 29 TOTAL CONSTRAINTS:
C
C           JA(29) = 24
C           MACT(24) = 29
C
C-----------------------------------------------------------------------
C
C        NSC: TOTAL NUMBER OF SPECIES
C
C        NCC: TOTAL NUMBER OF NON-DEGENERATE CONSTRAINTS
```

```
C       NDC: TOTAL NUMBER OF DEGENERATE SPECIES
C
C       NC0C: TOTAL NUMBER OF NON-DEGENERATE MASS AND CHARGE
C             BALANCE CONSTRAINTS
C
C       N0EC: TOTAL NUMBER OF CHARGE BALANCE CONSTRAINTS
C
C-----------------------------------------------------------------
C
C    DIMENSIONS  AND STRUCTURE OF THE QT AND R MATRICES.
C    (T=TRANSPOSE) (A IS THE MATRIX OF LINEARLY INDEPENDENT
C     CONSTRAINTS).
C
C   \Q1T\     \R11\R12\
C   \----\     ----\----
C   \Q2T\ A = \ 0 \R22\
C   \----\     ----\----
C   \Q3T\     \ 0 \ 0 \
C
C   R11 AND R22 ARE UPPER TRIANGULAR.
C
C   DIMENSIONS: Q1T: NC0C X NSC
C               Q2T: (NCC-NC0C) X NSC
C               Q3T: (NSC-NCC) X NSC
C
C               A: NSC X (NCC-NC0C)
C
C               R11: NC0C X NC0C
C               R12: NC0C X (NCC-NC0C)
C               R22: (NCC-NC0C) X (NCC-NC0C)
C
C-----------------------------------------------------------------
      COMMON/LABEL/Z(96),U0RT(96),A1(96)
C-----------------------------------------------------------------
C
C       Z(J):    CHARGE ON EACH SPECIES J
C
C       U0RT(J): STANDARD CHEMICAL POTENTIAL OF SPECIES J
C
C       A1(J): CHARACTER ARRAY FOR THE NAME OF SPECIES J
C
C-----------------------------------------------------------------
      COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1  ,NP0C,N00C
C-----------------------------------------------------------------
C
C       R11(J,J):  STORE "R" ARRAY FROM "QR" FACTORIZATION
C
C       JMIN(J)=I : MAP PHASE NUMBER J TO MINIMUM SPECIES NUMBER
C                   I IN PHASE J
C
C       JMAX(J)=I : MAP PHASE NUMBER J TO MAXIMUM SPECIES
C                   NUMBER I IN PHASE J
C
```

```
C-----------------------------------------------------------------
C
C       EXAMPLES:
C
C           SAME PHASES AS IN ABOVE EXAMPLE
C
C           JMIN(1) =  1       JMAX(1) = 20
C           JMIN(2) = 21       JMAX(2) = 21
C           JMIN(3) = 22       JMAX(3) = 22
C
C-----------------------------------------------------------------
C
C       KSPEC(J)=I: MAPS SPECIES NUMBER J  TO PHASE NUMBER I.
C                IN SAME EXAMPLE,
C             KSPEC(1) = 1
C             KSPEC(2) = 1
C             KPSEC(21) = 2
C
C       KTYPE(J)=I: MAPS PHASE NUMBER J TO PHASE TYPE I
C             I = 1   PURE PHASE
C             I = 2   IDEAL SOLUTION PHASE
C             I = 3   NON-IDEAL DAVIES AQUEOUS PHASE
C             I = 4   NON-IDEAL PITZER AQUEOUS PHASE
C             IF I IS NEGATIVE, THAT PHASE
C             IS NOT PRESENT
C
C       NP0C: THE TOTAL NUMBER OF PHASES (INCLUDING THOSE THAT
C            ARE NOT PRESENT)
C
C       N00C: THE TOTAL (INCLUDING DEGENERATE) CHARGE AND MASS
C            BALANCE CONSTRAINTS
C
C-----------------------------------------------------------------
      COMMON/SEARCH/X(96),P(96),G(96),W0(96),H(64,64),B(96),S(32)
C-----------------------------------------------------------------
C
C       X(J): MOLE NUMBER OF EACH SPECIES J
C
C       P(J): STEP DIRECTION OF SPECIES J AT EACH ITERATION
C
C       G(J): CHEMICAL POTENTIAL OF SPECIES J
C
C       W0(J): STORED COMPOSITIONS OF SPECIES J IF J IS IN
C             A SOLUTION PHASE THAT IS NOT PRESENT.
C            (FOR USE IN SUB-MINIMIZATION PROBLEM)
C
C       H(I,J): HESSIAN MATRIX
C
C       B(J): PROJECTED NON-DEGENERATE MASS AND CHARGE
C            BALANCE VECTOR (Q1 (TRANSPOSE) X)
C
C       S(J): IMPOSED CONSTRAINT BOUNDARIES ON NON-CONVEX
C            SOLUTION PHASES (USED IN THE SUB-MINIMIZATION
C            PROBLEM)
```

```
C-----------------------------------------------------------------
      COMMON/PRINT/PP,PP2,PD
C-----------------------------------------------------------------
C
C      PP: PRINT (MOSTLY) DIAGNOSTICS IN PRIMAL INCLUDING THE
C         MOLE  NUMBERS, FREE ENERGY, STEP DIRECTION, CHEMICAL
C         POTENTIALS AT EACH NEWTON ITERATION AS WELL AS THE
C         TYPE OF STEP (FEASIBILITY, PHASE ADDITION, PHASE
C         DELETION, REFINEMENT, OR NORMAL STEP).
C
C      PP2: PRINT SOLUTION TO THE PROBLEM
C
C      PD: PRINT (MOSTLY) DIAGNOSTICS IN DUAL. THE DIGANOSTICS
C         THAT ARE PRINTED ARE THE SAME VARAIBLES AS PRINTED
C         BY PP EXCEPT HERE THEY APPLY TO THE DUAL SUBROUTINE
C         (THE SUB-MINIZATION PROCEDURE)
C
C-----------------------------------------------------------------
      COMMON/CONST2/RR(96,16)
C-----------------------------------------------------------------
C
C     RR(J,J): UNFACTORED CONSTRAINT MATRIX (INCLUDING
C            DEGENERATE  CONSTRAINTS)
C
C-----------------------------------------------------------------
      COMMON/PH/LACT,NFILE,P0
C-----------------------------------------------------------------
      COMMON/MMM/MUL,WWW(96)
C-----------------------------------------------------------------
C
C      MUL: PRINT MULTIPLIERS (SEE THE FIRST STATEMENT)
C
C      WWW(J): MULTIPLIER OF SPECIES J
C
C-----------------------------------------------------------------
      PARAMETER(BMIN=1.0D-07,DEL9=1.0D-03)
      PARAMETER(DELMP=0.5D0**49)
      PARAMETER(DEL2=1.0D-12)
C-----------------------------------------------------------------
C
C      BMIN: THE MINIMUM AMOUNT (NON-SCALED) OF A PARTICULAR
C           COMPONENT
C
C      DEL9: MAXIMUM CHARGE IMBALANCE (IF GREATER THEN THIS,
C           EXIT, IF SMALLER, CORRECT IT.
C
C      DELMP: APPROXIMATE MACHINE PRECISION SCALED TO ONE
C          (I.E. .0000000000000001 MEANS 16 DIGITS OF PRECISION)
C
C      DEL2: CONSTRAINT BOUNDARY VALUE
C
C      DEL(J):  PROJECTED MASS BALANCE STEP DIRECTION
C              (Q1 (TRANSPOSE) (DELTA)X) THE PROJECTED CHANGE IN
C              THE ABSOLUTE NUMBER OF MOLES READ FROM THE
```

```
C             INPUT FOR EACH PROBLEM. (E.G. IF "SIG" FOR SPECIES "1"
C             (SEE THE EXPLANATION OF LINES 14-16 IN THE INPUT
C             EXAMPLE) IS EQUAL TO ".1", AND ALPHA IS EQUAL TO
C             ONE, ".1" MOLES OF SPECIES "1" WILL BE ADDED AFTER
C             EACH ITERATION. DEL(J) STORES THE Q1 PROJECTED
C             VALUES OF THESE MASS BALANCE CHANGES. SEE
C             ALSO "PF".
C
C       BB(J):  UNPROJECTED, MASS AND CHARGE BALANCE ARRAY
C               (INCLUDING  DEGENERATE CONSTRAINTS)
C
C       U0(J): STANDARD CHEMICAL POTENTIAL FOR P = 1. I.E. IF P=2
C          U0RT(J)=U0(J)+LN(2.)
C
C       JH(J)=I : VECTOR THAT MAPS SPECIES NUMBER J THAT INCLUDE
C             SPECIES THAT ARE WITHHELD TO THE ACTUAL SPECIES C
C             NUMBER I.  IF THE SPECIES IS WITHHELD, I IS EQUAL TO -1
C
C          EXAMPLE:
C                SPECIES 5 IS WITHHELD
C
C         INPUT SPECIES NUMBER      ACTUAL SPECIES NUMBER (JH)
C
C                        1                    1
C                        2                    2
C                        3                    3
C                        4                    4
C                        5                   -1
C                        6                    5
C                        7                    6
C                        8                    7
C
C
C          PF(J): MASS BALANCE STEP DIRECTION FOR PERCENT CHANGE
C                IN SPECIES J (I.E. AN EVAPORATION)
C
C          I=JFRAX(J): IF I IS EQUAL TO ONE, SPECIES J ( A MINERAL)
C                    WILL NOT BE FRACTIONATED. IF I IS EQUAL TO
C                    ZERO, SPECIES J CAN BE FRACTIONATED.
C
C          WATLAB: CHARACTER ARRAY THAT STORES A LABEL THAT
C                    IDENTIFIES A PARTICULAR COMPOSITION
C
C-----------------------------------------------------------------------
C
      LOGICAL PP,PP2,PD,PD2,MUL
C
C SAS
C
      OPEN(UNIT=4, FILE='SALT', STATUS='OLD')
      OPEN(UNIT=5, FILE='THERMO.DAT', STATUS='OLD')
      REWIND(4)
      REWIND(5)
      INQUIRE(FILE='SOLUTION', EXIST=OUTFIL)
```

```
      IF(OUTFIL) THEN
      WRITE(*,5058)
 5058      FORMAT( 1X,'THE OUTPUT FILE "SOLUTION" ALLREADY
EXISTS'/
     1 1X,'FROM A PREVIOUS SIMULATION.  YOU CAN USE THE'/
     2 1X,'DOS RENAME COMMAND NOW OR JUST HIT THE RETURN'/
     3 1X,'KEY TO OVERWRITE THE OLD FILE.  PROGRAM EXECUTION'/
     4 1X,'WILL RUSUME AUTOMATICALLY EITHER WAY.'/)
      PAUSE
      OPEN(UNIT=6, FILE='SOLUTION', STATUS='UNKNOWN')
      CLOSE(UNIT=6, STATUS='DELETE')
      END IF
      OPEN(UNIT=6,FILE='SOLUTION',STATUS='NEW')
      REWIND(6)
C SAS
C
      READ(4,5002) IPRINT,IPR,IPR2,IPR3
C-----------------------------------------------------------------------
C
C      FORMAT(6I5)
C      ALL PRINT OPTION LOGICAL VARIABLES ARE INITIALIZED TO
C        .FALSE.
C      IPRINT = 1:   PP SET EQUAL TO .TRUE.
C      IPR = 1:      PP2 SET EQUAL TO TRUE.
C      IPR2 = 1:     PD  SET EQUAL TO .TRUE.
C      IPR3 = 1:     MUL SET EQUAL TO .TRUE.
C
C      THE LOGICAL VARIABLES ARE EXPLAINED ABOVE
C
C-----------------------------------------------------------------------
      WRITE(*,*) 'CALLING DATA'
      CALL DATA(JFRAX,M00,JH)


      IF(M00.LT.NC0C) THEN
C
C      THIS "IF" STATEMENT SHOULD NEVER EXECUTE
C
           WRITE(6,9988)
 9988       FORMAT('ERROR IN MATRIX FACTORIZATION ROUTINE')
           STOP
      END IF
      PP=.FALSE.
      PP2=.FALSE.
      PD=.FALSE.
      PD2=.FALSE.
      MUL=.FALSE.
      IF(IPRINT.EQ.1) PP=.TRUE.
      IF(IPR.EQ.1) PP2=.TRUE.
      IF(IPR2.EQ.1) PD=.TRUE.
      IF(IPR3.EQ.1) MUL=.TRUE.
      NS=NSC
      DO 30 J=1,NS
      U0(J)=U0RT(J)
```

```
   30   CONTINUE
        IWAT=0
         REWIND(4)
  500   READ(4,5007) NTR,ALPHA,FRAX,ALPHA2
C-----------------------------------------------------------------------
C
C       IWAT : THE NUMBER OF THE CURRENT BRINE SYSTEM BEING
C       CONSIDERED (E.G. IF THE EVAPORATION SEQUENCE OF BRINE NO.
C       1 IS BEING CALCULATED IWAT =1.  IF BRINE NO. 2 IS THEN READ
C       IN AND EVAPORATED, IWAT=2). THIS VARIABLE IS USED ONLY FOR
C       THE PURPOSE OF OUTPUT.
C
C       NTR: THE NUMBER OF ITERATIONS (I.E. THE NUMBER OF
C       EVAPORATION STEPS)
C
C       ALPHA: THE STEP LENGTH FOR THE ABSOLUTE NUMBER OF
C       MOLES THAT WILL CHANGE AT EACH STEP
C
C       FRAX: THE FRACTIONATION FACTOR (BETWEEN 0. AND  1.)
C
C       ALPHA2: THE STEP LENGTH FOR THE PERCENT NUMBER OF
C       MOLES THAT WILL CHANGE AT EACH STEP
C
C       FORMAT(I5,3F10)
C
C       SEE THE EXPLANATION OF THE READ AFTER THE NEXT ONE
C       FOR AN EXAMPLE.
C
C-----------------------------------------------------------------------
        READ(4,5001) WATLAB
C-----------------------------------------------------------------------
C
C        A LABEL THAT IS WRITTEN TO THE OUTPUT FILE TO IDENTIFY
C        THE SOLUTION FORMAT(A30)
C
C-----------------------------------------------------------------------
        IWAT=IWAT+1
        WRITE(6,5000)  IWAT,IWAT,IWAT
        WRITE(6,5001) WATLAB
        N00=N00C
        N0E=N0EC
        N0EP1=N0E+1
        NC0=NC0C
        DO 10  I=1,M00
        B(I)=0.0D0
        BB(I)=0.0D0
        DEL(I)=0.0D0
   10   CONTINUE
        DO 13 K=1,NP0C
        KTYP(K)=-IABS(KTYP(K))
   13   CONTINUE
        DO 11 J=1,NS
        X(J)=0.0D0
        P(J)=0.0D0
```

```
      PF(J)=0.0D0
      IF(MACT(J).GT.NC0) THEN
         WRITE(*,*) 'CALLING CONDEL FROM MAIN'
         CALL CONDEL(J)
      END IF
C
C      WHEN A NEW COMPOSTION IS READ IN, RELAX ALL SPECIES
C      CONSTRAINTS
C
C
   11 CONTINUE
      BBMAX=0.0D0
      WRITE(6,5004) NTR,ALPHA,FRAX,ALPHA2
      IF(FRAX.LT.0.0D0.OR.FRAX.GT.1.0D0)  GO TO 9995
      WRITE(6,5010)
      PHI=0.0D0
      PHI2=0.0D0

C SAS
C   REWIND INPUT FILE AND READ STARTING CONCENTRATIONS, ETC.
      J=1
      BLANK=.FALSE.
      REWIND(4)
      READ(4,5056)
 5056 FORMAT(/////////////)
  100 READ(4,5016) NSPEC,TAU,SIG,ETA,ZETA

C-----------------------------------------------------------------------
C
C       FORMAT(I5,4F15)
C
C       J: SPECIES NUMBER. SPECIES NUMBERS ARE ASSIGNED TO
C         SPECIES  IN ORDER AS THEY ARE READ IN WITHIN DATA.F
C
C       TAU: INITIAL MOLE NUMBER OF SPECIES J
C
C       SIG: ABSOLUTE CHANGE AT EACH STEP IN THE MASS
C            REPRESENTED BY SPECIES J (MULTIPLITED BY ALPHA)
C
C       ETA: THE NATURAL LOG OF THIS VARIABLE (IF NON-ZERO)
C            IS ADDED TO THE CHEMICAL POTENTIAL OF SPECIES J.
C            THE MOST COMMON USE OF THIS VARIABLE IS TO
C            FIX A GASES FUGACITY.
C
C       ZETA: PERCENT CHANGE AT EACH STEP IN THE MASS
C             REPRESENTED BY SPECIES J (MULTIPLITED BY ALPHA2)
C
C       EXAMPLE: DO FOUR STEPS EVAPARATING A 2M. SODIUM
C       CHLORIDE 1M. SODIUM SULFATE SOLUTION IN EQUILIBRIUM WITH
C       A CARBON DIOXIDE ATMOSPHERE AT A PRESSURE OF .00034 ATM.
C       WHERE AT EACH STEP THE SOLIDS ARE FRACTIONATED 10%, 30%
C       OF THE WATER IS REMOVED AT EACH STEP, AND .1 MOLES OF
C       SODIUM SULFATE ARE ADDED AT EACH STEP
C
```

```
C         SPECIES   SPECIES NUMBER
C
C         H2O              1
C         NA+              2
C         CL-              3
C         SO4--            4
C         CO2(AQ)          5
C         HALITE           6
C         THENARDITE       7
C         MIRABALITE       8
C         CO2(GAS)         9
C
C        THEN NTR=1, ALPHA=  1., FRAX=.1, ALPHA2= .3
C
C  SPECIES      J       TAU       SIG    ETA      ZETA
C
C  H2O           1      55.508   0.0                -1.
C  NA+           2        4.        .2                 0.0
C  CL-           3        2.       0.                  0.0
C  SO4-2        4        1.        .1                 0.
C  CO2 (GAS)    9       10.       0.0    .00034    0.0
C
C  THIS COMPLETES THE INPUT: THE FOLLOWING IS A SAMPLE INPUT
C  FILE FOR A SIMPLE SODIUM CHLORIDE WATER SOLUTION AT 25
C  DEGREES. (THE NUMBERS ON THE LEFT ARE USED WOULD NOT BE IN
C  THE INPUT FILE. THEY ARE PUT HERE ONLY FOR REFERENCE IN
C  THIS EXAMPLE. THE "\" SYMBOL REPRESENTS THE LEFT BORDER OF
C  THE FILE AND SHOULD NOT BE IN THE INPUT FILE EITHER.
C
C-----------------------------------------------------------------------
      IF(NSPEC.EQ.0) THEN
         IF(BLANK) GO TO 101
         BLANK=.TRUE.
      ELSE IF(NSPEC.LT.10) THEN
         BLANK=.FALSE.
      ELSE IF(TAU.EQ.0.0D0) THEN
         BLANK=.FALSE.
         J=J+1
      ELSE
         JJ=JH(J)
         PF(JJ)=ZETA
         IF(JJ.EQ.-1) THEN
            WRITE(6,5555) J
            STOP
         END IF
         IF(ETA.NE.0.0D0) THEN
            IF(IABS(KTYP(KSPEC(JJ))).NE.1)  THEN
               WRITE(6,9989)
9989            FORMAT('CHANGES IN THE STANDARD CHEMICAL',
   1            'POTENTIALS OF SOLUTION SPECIES ARE NOT
   2             ALLOWED')
               STOP
            END IF
            P0=ETA
```

```
            U0RT(JJ)=U0(JJ)+DLOG(ETA)
            G(JJ)=U0RT(JJ)
          END IF
          K0=KSPEC(JJ)
          KTYP(K0)=IABS(KTYP(K0))
          WRITE(6,5012) JJ,K0,A1(JJ),TAU,SIG,ETA,ZETA
          X(JJ)=X(JJ)+TAU
          PHI=PHI+SIG*Z(JJ)
          PHI2=PHI2+ZETA*Z(JJ)
          DO 12 I=N0EP1,NC0
          DEL(I)=DEL(I)+SIG*Q(JJ,I)
 12       CONTINUE
          J=J+1
          BLANK=.FALSE.
          END IF
          GO TO 100
101     CONTINUE
          IF(DABS(PHI).GT.DELMP.OR.DABS(PHI).GT.DELMP) THEN
          WRITE(6,5050) PHI,PHI2
5050       FORMAT('STEP DIRECTION NOT CHARGE BALANCED ',1P2D12.3)
          STOP
        END IF
        DO 104 K=1,NP0C
        IF(KTYP(K).GT.0) THEN
          CHARGE=.FALSE.
          JM=JMIN(K)
          JX=JMAX(K)
          SIG=0.0D0
          PHI=0.0D0
          ICMAX=0
          IAMAX=0
          ZCMAX=0.0D0
          ZAMAX=0.0D0
          DO 106 J=JM,JX
          DO 116 I=1,M00
          BB(I)=BB(I)+RR(J,I)*X(J)
 116      CONTINUE
          IF(Z(J).NE.0) THEN
            PHI=PHI+X(J)*Z(J)
            IF(Z(J).LT.0) THEN
              IF(X(J).GT.ZAMAX) THEN
                IAMAX=J
                ZAMAX=X(J)
              END IF
            ELSE IF(Z(J).GT.0) THEN
              IF(X(J).GT.ZCMAX) THEN
                ICMAX=J
                ZCMAX=X(J)
              END IF
            END IF
          END IF
 106      CONTINUE
          IF(DABS(PHI).GT.DELMP) THEN
            IF(DABS(PHI).GT.DEL9*DMAX1(ZCMAX,ZAMAX)) THEN
```

```
                WRITE(6,5052) PHI
5052             FORMAT('SOLUTION NOT CHARGE BALANCED ',1PD12.3)
                STOP
              ELSE
                CHARGE=.TRUE.
                II=0
                IF(PHI.LT.0.0D0) THEN
                   X(ICMAX)=X(ICMAX)-PHI/Z(ICMAX)
                   II=ICMAX
                ELSE
                   X(IAMAX)=X(IAMAX)-PHI/Z(IAMAX)
                   II=IAMAX
                END IF
                IF(II.EQ.0) THEN
C
C    THIS IF STATEMENT SHOULD NEVER BE TRUE
C
                    WRITE(6,9990)
9990                 FORMAT('NO SPECIES FOUND FOR CORRECTING',
   1                 'CHARGE BALANCE')
                    STOP
                ENDIF
                DO 117 I=1,M00
                BB(I)=BB(I)-RR(II,I)*PHI/Z(II)
117              CONTINUE
              END IF
          END IF
          DO 109 I=N0EP1,NC0
          DO 108 J=JM,JX
          B(I)=B(I)+Q(J,I)*X(J)
108       CONTINUE
109       CONTINUE
          IF(CHARGE) THEN
             WRITE(6,5015)
5015          FORMAT(/,'SOLUTION PHASE CHARGE BALANCE CORRECTION')
             WRITE(6,5017)
             DO 118 J=JM,JX
             IF(X(J).GT.0.0D0) WRITE(6,5012) J,K,A1(J),X(J)
118          CONTINUE
          END IF
        END IF
104     CONTINUE
        DO 107 I=N0EP1,M00
        IF(DABS(BB(I)).LT.BMIN) THEN
            WRITE(6,5051) I,BB(I)
5051         FORMAT('MASS BALENCE ERROR ',I5,2X,1PD12.3)
            STOP
        END IF
107     CONTINUE
        DO 600 ITR=1,NTR
        IF(ITR.EQ.1) ERROR=.TRUE.
        WRITE(*,*) 'CALLING PRIMAL'
        CALL PRIMAL(ITR,ERROR)
        IF(ERROR) THEN
```

```
          WRITE(6,5014) ITR,ITR,ITR
          STOP
        ELSE
          IF(FRAX.GT.0.0D0) THEN
            DO 28 K=1,NP0C
            IF(KTYP(K).EQ.1) THEN
              IF(JFRAX(K).EQ.0) THEN
                J0=JMIN(K)
                TAU=X(J0)*FRAX
                X(J0)=X(J0)-TAU
                DO 29 I=N0EP1,NC0
                B(I)=B(I)-TAU*Q(J0,I)

 29             CONTINUE
                IF(X(J0).LE.DEL2) KTYP(K)=-IABS(KTYP(K))
              END IF
            END IF
 28         CONTINUE
          END IF
          DO 36 J=1,NS
          DO 26 I=N0EP1,NC0
          B(I)=B(I)+ALPHA2*X(J)*PF(J)*Q(J,I)
 26       CONTINUE
 36       CONTINUE
          DO 27 I=N0EP1,NC0
          B(I)=B(I)+ALPHA*DEL(I)
 27       CONTINUE
        END IF
 600    CONTINUE
        WRITE(6,2500)
2500    FORMAT('NORMAL TERMINATION OF EQUIL')
        STOP

9995    WRITE(6,5025) FRAX
        STOP
5002    FORMAT( / 7X,I5 / 4X,I5 / 5X,I5 / 5X,I5 / 5X,I5)
5003    FORMAT(4F15.7)
5001    FORMAT( 5X,A50)
5000    FORMAT(3(15('-'),I5),15('-'))
5004    FORMAT('NTR,ALPHA,FRAX,ALPHA2= ',I5,3F10.5)
5006    FORMAT(F10.5)
5007    FORMAT( /////// 4X,I5/ 6X,F10.5/  5X,F10.5/ 7X,F10.5)
5008    FORMAT(/,2X,'NO.',10X,'B(INPUT)',6X,'B(PROJ)',17X,'DEL(INPUT)',
     1  4X,'DEL(PROJ)')
5009    FORMAT(I5,2(5X,1P2D13.5,0PF6.3))
5010    FORMAT(11X,'SPECIES        X            P          A ')
5017    FORMAT(11X,'SPECIES                X')
5012    FORMAT(2I5,1X,A8,1P4D12.5)
5013    FORMAT(' TEMPERATURE = ',F10.4,' PRESSURE= ',F10.4)
5014    FORMAT('****************',(3I5,'****************'))
5016    FORMAT(I3,4F15.7)
5025    FORMAT('FRAX  = ',1PD12.3)
5555    FORMAT(' SPECIES NO. ',I5,' WITHHELD')
        END
```

```
C-----------------------------------------------------------------
C
C       THIS SUBROUTINE READS IN THE INFORMATION ABOUT EACH
C       SPECIES (NAME, CHARGE, STANDARD CHEMICAL POTENTIAL
C       DIVIDED BY RT, AND THE SPECIFICATION OF COMPONENTS) AND
C       FACTORS THE INITIAL CONSTRAINT MATRIX (I.E. CHARGE AND
C       MASS BALANCE BUT NO SPECIES CONSTRAINTS).
C
C
C-----------------------------------------------------------------
      SUBROUTINE DATA(JFRAX,M0,JH)
C-----------------------------------------------------------------
C
C       M0: TOTAL NUMBER OF MASS AND CHARGE BALANCE
C          CONSTRAINTS INCLUDING DEGENERATE ONES (SET TO M00 IN
C          AMAIN2.F)
C
C-----------------------------------------------------------------
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 Z
      INTEGER*2 LC(100),JV(9)
      CHARACTER*24 A1
      LOGICAL BLANK,ELECT,POS,NEG,SOLN,IDEAL,NODEG,REDUN
C-----------------------------------------------------------------
C
C       BLANK: SET TO .TRUE. WHEN A BLANK LINE IS READ
C
C       ELECT: SET TO .TRUE. IF THE INPUT PHASE BEING READ IS AN
C            ELECTROLYTE PHASE
C
C       POS: SET TO .TRUE. IF SPECIES BEING READ IS POSITIVE
C
C       NEG: SET TO .TRUE. IF SPECIES BEING READ IS NEGATIVE
C
C       POS AND NEG ARE USED TO INSURE THAT FOR AN ELECTROLYTE
C       SOLUTION PHASE, BOTH  POSITIVE AND NEGATIVE SPECIES ARE
C       DEFINED IN THE INPUT FILE.
C
C       SOLN: SET TO.TRUE. IF INPUT PHASE BEING READ IS A SOLUTION
C       PHASE
C
C       IDEAL: SET TO .TRUE. IF PHASE BEING READ IS AN IDEAL
C       SOLUTION
C            PHASE (THE VARIABLE IS SET BUT NOT USED)
C
C       NODEG: SET TO .TRUE. IF DURING THE QR FACTORIZATION, A
C            CONSTRAINT IS FOUND TO BE NOT DEGENERATE
C
C       REDUN: SET TO .TRUE. IF A SPECIES IS FOUND TO BE
C       DEGENERATE
C
C-----------------------------------------------------------------
     COMMON/CONST2/RR(96,16)
```

```
      COMMON/CONST/Q(96,96),MACT(96),JA(96),JI(96),NSC,NCC,NDC,NC0C
     1,N0EC
      COMMON/LABEL/Z(96),U0RT(96),A1(96)
      COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1,NP0C,N00C
      COMMON/SEARCH/X(96),P(96),G(96),W0(96),H(64,64),B(96)
      COMMON/DBASE/NCAT(18),NANI(18),ICAT,IANI,NCD,NAD,NND,TEMP
        DIMENSION R(96,96),V(96),IAN(96),JH(96),JFRAX(96)
C-----------------------------------------------------------------
C
C  R: STORE UNFACTORED CONSTRAINT MATRIX. AS THE QR ALGORITHM
C  PROCEEDS IT BECOMES THE UPPER TRIANGULAR MATRIX. R IS A
C  LOCAL VARIABLE WHICH SHARES ITS SPACE WITH THE H MATRIX.
C  THE R MATRIX IS STORED GLOBALLY IN R11.
C
C  IAN: AN ARRAY THAT STORES THE CONSTRAINTS ACCORDING TO THE
C  NEW CONSTRAINT NUMBERS GIVEN BY LC. THIS ARRAY IS USED ONLY
C  TO PRINT THE CONSTRAINT TABLE ON THE OUTPUT
C
C-----------------------------------------------------------------
       PARAMETER(DEL4=1.0D-13)
C-----------------------------------------------------------------
C
C       DEL4: VALUE THAT DETERMINES CONSTRAIN DEGENERACY.
C            I.E. IF IN R WE HAVE:
C
C              X   X   X
C              0   X   X
C              0   0   0
C              0   0   0
C              0   0   0
C              0   0   0
C
C            CONSTRAINT 3 IS DEGENERATE SINCE THE DIAGONAL
C            ELEMENT IS EQUAL TO DEL4( VERY CLOSE TO ZERO)
C
C-----------------------------------------------------------------
       PARAMETER(NMAX=96,MMAX=16)
       EQUIVALENCE (H(1,1),R(1,1))
C
C       INITIALIZE Q MATRIX AND LC VECTOR
C
 2000   FORMAT(/,1X,'SOLUTION PHASE',I2,3X,'PHASE TYPE',I2,/
    1,5X,'SPECIES',16X,'CHARGE',6X,'U0/RT')
 2001   FORMAT(/,1X,'PURE PHASES',/,5X,'SPECIES',32X,'U0/RT')
 2004   FORMAT(/,1X,'IDEAL SOLUTION PHASE',I2,/
    1,5X,'SPECIES',8X,'CHARGE',10X,'U0/RT')
       DO 1 J=1,100
       LC(J)=0
   1   CONTINUE
       DO 2 I=1,NMAX
       DO 2 J=1,NMAX
       Q(J,I)=0.0D0
   2   CONTINUE
```

```
C
C -----------------------------------------------------------------
C
C      SYSTEM DATA INPUT LOOP
C
C      (A) READ SOLUTION PHASES
C            (1) IDENT LINE   FOR PHASE IDENTIFICATION
C                IDENT = 2 3 AND 4
C            (2) SPECIES LINES
C                  A) READ SPECIES ID NUMBER
C                  B) SEARCH DATABASE FOR COMPOSITION AND
C                     CHEMICAL POTENTIAL DATA
C                  C) REPEAT FOR NEXT SPECIES
C            (3) ZERO LINE
C            (4) NON-IDEAL SOLUTION DATA READ IN DATAP
C                IF IDENT > 2  (IDENT = 2 FOR IDEAL SOLUTION)
C                  A) SEARCH DATABASE FOR PITZER DATA
C      (B) READ PURE PHASES
C            (1) IDENT LINE   IDENT = 1
C            (2) PURE PHASE LINES
C                  A) READ SPECIES I.D. NUMBER
C                  B) SEARCH DATABASE ETC.
C                  C) REPEAT FOR NEXT SPECIES
C            (3) ZERO LINE
C
C      NCAT(ICAT)=SPECIES I.D. NUMBER.  VECTOR THAT MAPS THE Ith
C      CATION TO ITS DATABASE IDENTIFICATION NUMBER
C
C      NANI(I)=SPECIES I.D.NUMBER. VECTOR THAT MAPS THE Ith ANION
C      TO ITS DATABASE IDENTIFICATION NUMBER
C
C
C      WHEN IDENT IS GREATER THAN 2 THE SUBROUTINE DATAP  IS
C      CALLED TO READ IN NON-IDEAL SOLUTION MODEL PARAMETERS.
C      THE PARAMETER IDENT IS USED TO IDENTIFY THE TYPE OF
C      MODEL TO BE USED FOR THE PHASE. FOR NON-IDEAL SOLUTION
C      PHASES GRAD,HESS,AND DATAP USE IDENT.
C
       NCOMP=0
       NP0=0
       M0=0
       J=1
       SOLN=.TRUE.
C
C SAS
C      READ TOTAL NUMBER OF CATIONS, ANIONS, AND NEUTRALS IN
C      DATABASE
       READ(5,1002) NCD,NAD,NND

C      READ SYSTEM TEMPERATURE (CELCIUS)
       READ(4,1003) TEMP

C
       ICAT=0
```

```
      IANI=0

       READ(4,1008)
  100  READ(4,1000) IDENT
C-----------------------------------------------------------------
C
C      IDENT: PHASE TYPE:
C      1 PURE PHASE
C      2 IDEAL SOLUTION PHASE
C      3 AQUEOUS SOLUTION PHASE WITH DAVIES ACTIVITY
C         COEFFICIENTS
C      4 AQUEOUS SOLUTION PHASE WITH PITZER ACTIVITY
C         COEFFICIENTS
C      FORMAT(I5)
C SAS
C-----------------------------------------------------------------
 1000  FORMAT( I3,60X)
      IF(IDENT.LT.1) GO TO 103

  106  NP0=NP0+1
      KTYP(NP0)=-IDENT
C
C      KTYP IS INITIALIZED TO ALL NEGATIVE NUMBERS. THAT IS, ALL
C      PHASES ARE NOT PRESENT. WHEN MOLE NUMBERS FOR SPECIES
C      ARE READ IN IN AMAIN2.F, KTYP FOR THE CORRESPONDING PHASE
C      IS SET TO POSITIVE.
C
      JMIN(NP0)=J
      ELECT=.FALSE.
      NSPEC=0
      IF(IDENT.GT.1) THEN
         IF(.NOT.SOLN) THEN
            WRITE(6,990)
  990       FORMAT('SOLUTION PHASE CANNOT FOLLOW A PURE PHASE')
            STOP
         ENDIF
         NSOLNC=NP0
         IF(IDENT.EQ.2) THEN
            WRITE(6,2004) NP0
         ELSE
            WRITE(6,2000) NP0,IDENT
         END IF
      ELSE IF(IDENT.EQ.1.AND.SOLN) THEN
         SOLN=.FALSE.
         WRITE(6,2001)
         NP0=NP0-1
         JJ=J
      END IF
C
C SAS
C      SEARCH DATABASE FOR SPECIES INDICATED IN INPUT FILE
  99   READ(4,1004) NSPEC1
      BLANK=.FALSE.
      IF (NSPEC1.EQ.0) THEN
```

```
      BLANK=.TRUE.
      GO TO 101
    ELSE IF (NSPEC1.LT.99) THEN
      WRITE(*,1006) NSPEC1
      STOP
    ELSE IF (NSPEC1.EQ.99) THEN
      READ(5,1005) NSPEC2, A1(J), Z(J), (V(K),JV(K),K=1,7),
     1     U0RT0, U0RT1, U0RT2, U0RT3, U0RT4
          IF (NSPEC2.NE.99) THEN
          WRITE(*,1006) NSPEC1
          STOP
          END IF
      GO TO 101
    ELSE IF (NSPEC1.LT.200) THEN
50    READ(5,1005) NSPEC2, A1(J), Z(J),(V(K),JV(K),K=1,7),
     1     U0RT0, U0RT1, U0RT2, U0RT3, U0RT4
      IF (NSPEC2.EQ.NSPEC1) THEN
         ICAT=ICAT+1
         NCAT(ICAT)=NSPEC2
         GO TO 101
      ELSE IF (NSPEC2.GT.NSPEC1) THEN
         WRITE(*,1006) NSPEC1
         STOP
      ELSE
         GO TO 50
      END IF
    ELSE IF (NSPEC1.LT.300) THEN
51    READ(5,1005) NSPEC2, A1(J), Z(J), (V(K),JV(K),K=1,7),
     1     U0RT0, U0RT1, U0RT2, U0RT3, U0RT4
      IF (NSPEC2.EQ.NSPEC1) THEN
         IANI=IANI+1
         NANI(IANI)=NSPEC2
         GO TO 101
      ELSE IF (NSPEC2.GT.NSPEC1) THEN
         WRITE(*,1006) NSPEC1
         STOP
      ELSE
         GO TO 51
      END IF
    ELSE IF (NSPEC1.LT.400) THEN
52    READ(5,1005) NSPEC2, A1(J), Z(J), (V(K),JV(K),K=1,7),
     1     U0RT0, U0RT1, U0RT2, U0RT3, U0RT4
      IF (NSPEC2.EQ.NSPEC1) THEN
         GO TO 101
      ELSE IF (NSPEC2.GT.NSPEC1) THEN
         WRITE(*,1006) NSPEC1
         STOP
      ELSE
         GO TO 52
      END IF
    ELSE
53    READ(5,1007) NSPEC2, A1(J), (V(K),JV(K),K=1,7),
     1     U0RT0, U0RT1, U0RT2, U0RT3, U0RT4
      IF (NSPEC2.EQ.NSPEC1) THEN
```

```
            GO TO 101
          ELSE IF (NSPEC2.GT.NSPEC1) THEN
            WRITE(*,1006) NSPEC1
            STOP
          ELSE IF (NSPEC2.EQ.0) THEN
            WRITE(*,1006) NSPEC1
            STOP
          ELSE
            GO TO 53
          END IF
        END IF

 1002   FORMAT( ////// I4 / I4 / I4 /)
 1003   FORMAT( 23X, F10.5)
 1004   FORMAT(I4)
 1005   FORMAT(I4, A24, I2, 7(F4.3,I3) /14X, 5E13.5)
 1006   FORMAT(1X,'SPECIES NUMBER ',I3,'NOT FOUND IN DATABASE OR',
     1 ' INPUT NOT IN ASCENDING ORDER')
 1007FORMAT(I4,A24,2X,7(F4.3,I3) /14X,5E13.5)
 1008FORMAT( //////)
  101   CONTINUE

C SAS

        IF(.NOT.SOLN) THEN
          IF(BLANK) GO TO 103
          JFRAX(NP0+1)=0
          IF(Z(J).GT.0) THEN
              IF(Z(J).EQ.1) THEN
                  WRITE(6,5999) A1(J)
                  JH(JJ)=-1
                  JJ=JJ+1
                  GO TO 101
              ELSE IF(Z(J).EQ.2) THEN
                  JFRAX(NP0+1)=1
              ELSE
                  WRITE(6,991) Z(J)
  991             FORMAT('ILLEGAL PURE PHASE OPTION',I5)
              END IF
          END IF
          NP0=NP0+1
          KTYP(NP0)=-1
          JMIN(NP0)=J
          JMAX(NP0)=J
          KSPEC(J)=NP0
          WRITE(*,*) 'CALLING DATAP @ LN 341'
          CALL DATAP(NP0)
          X(J)=0.0D0
          P(J)=0.0D0
C SAS
     U0RT(J)=U0RT0 +U0RT1*(TEMP-25.0D0) +U0RT2*(TEMP-25.0D0)**2
      1+U0RT3*(TEMP-25.0D0)**3 +U0RT4*(TEMP-25.0D0)**4
C SAS
          G(J)=U0RT(J)
```

```
            W0(J)=1.0D0
            Z(J)=0
            JH(JJ)=J
            JJ=JJ+1
            WRITE(6,2007) J,A1(J),U0RT(J)
 2007        FORMAT(1X,I3,1X,A24,7X,F17.8)
        ELSE
        IF(BLANK) GO TO 102
            JH(J)=J
            KSPEC(J)=NP0
            X(J)=0.0D0
            P(J)=0.0D0
C SAS
       U0RT(J)=U0RT0 +U0RT1*(TEMP-25.0D0) +U0RT2*(TEMP-25.0D0)**2
     1+U0RT3*(TEMP-25.0D0)**3 +U0RT4*(TEMP-25.0D0)**4
C SAS
            G(J)=U0RT(J)
            W0(J)=.01D0
            IF(Z(J).NE.0) ELECT=.TRUE.
            NSPEC=NSPEC+1
            WRITE(6,2003) J,A1(J),Z(J),U0RT(J)
 2003        FORMAT(1X,I3,1X,A24,I2,F17.8)
        END IF
        DO 4 K=1,9
        IF(JV(K).GT.0) THEN
            IF(LC(JV(K)).LE.0) THEN
                NCOMP=NCOMP+1
                LC(JV(K))=NCOMP
            END IF
            Q(J,LC(JV(K)))=V(K)
        END IF
   4    CONTINUE
        J=J+1
        GO TO 99
  102   JMAX(NP0)=J-1
        W0(JMIN(NP0))=1.0D0

        IF(NSPEC.LT.2) THEN
             WRITE(6,992)
  992         FORMAT('A SOLUTION PHASE MUST HAVE MORE THEN ONE
SPECIES')
             STOP
        ENDIF
        IF(ELECT) THEN
            M0=M0+1
            IAN(M0)=0
            JM=JMIN(NP0)
            JX=JMAX(NP0)
            POS=.TRUE.
            NEG=.TRUE.
            DO 5 I=1,NMAX
            R(I,M0)=0.0D0
            RR(I,M0)=R(I,M0)
    5       CONTINUE
```

```
            DO 7 I=JM,JX
            R(I,M0)=Z(I)
            RR(I,M0)=R(I,M0)
            IF(Z(I).LT.0) NEG=.FALSE.
            IF(Z(I).GT.0) POS=.FALSE.
    7       CONTINUE
            IF(POS.OR.NEG) THEN
                WRITE(6,993)
  993           FORMAT('YOU MUST HAVE BOTH POSITIVE AND NEGATIVE
IONS')
                STOP
            ENDIF
        END IF
C
C       IDENT=1 PURE PHASE
C       IDENT=2 IDEAL SOLUTION
C       IDENT=3 AQUEOUS DAVIES EQN
C       IDENT=4 AQUEOUS PITZER EQN
C
        WRITE(*,*) 'CALLING DATAP @ LN 460'
        CALL DATAP(NP0)
        GO TO 100
C
C -------------------------------------------------------------------
C
C       FORM CONSTRAINT MATRIX IN R
C
  103   NS=J-1
        MX=M0
        DO 11 I=1,100
        IF(LC(I).EQ.0) GO TO 11
        M0=M0+1
        IF(M0.GT.MMAX) THEN
            WRITE(6,5500) M0
 5500       FORMAT(I5,' TOO MANY CONSTRAINTS,ENLARGE RR AND R11
ARRAY')
        END IF
        IAN(M0)=I
        TAU=1.0D+20
        DO 10 J=1,NS
        R(J,M0)=Q(J,LC(I))
        RR(J,M0)=R(J,M0)
        IF(R(J,M0).GT.1.0D-05.AND.R(J,M0).LT.TAU) TAU=R(J,M0)
   10   CONTINUE
C
C       STORE SPECIES THAT HAS THE LEAST AMOUNT OF A GIVEN
C       COMPONENT (USED TO CALCULATE THE SCALING FACTOR IN
C       PRIMAL.F) SINCE WE WISH TO FIND THE MINIMUM VALUE, TAU IS
C       INITIALIZED AT A VERY LARGE NUMBER.
C
C       EXAMPLE: THE FOLLOWING SPECIES ALL CONTAIN HYDROGEN:
C
C       H20:  2 HYDROGENS
C
```

```
C       MIRABALITE : 10 HYDROGENS
C
C       GYPSUM : 4 HYDROGENS
C
C       FOR THE HYDROGEN COMPONENT, TAU WOULD BE EQUAL TO "2"
C       BECAUSE H2O HAS THE LEAST NUMBER OF HYDROGENS PER
C       MOLE OF SPECIES.
C
C
C
        B(M0)=TAU
   11   CONTINUE
        WRITE(6,2002) (IAN(K),K=1,M0)
 2002   FORMAT(///,1X,'CONSTRAINT EQUATIONS',/,5X,'SPECIES',9X,20I5)
        DO 12 J=1,NS
        WRITE(6,2005) J,A1(J),(R(J,K),K=1,M0)
 2005   FORMAT(1X,I3,1X,A16,20F5.1)
   12   CONTINUE
C
C ---------------------------------------------------------------------
C       INITIALIZE ORTHOGONAL CONSTRAINT MATRIX
C
C       INITIALIZE THE Q MATRIX TO THE IDENTITY MATRIX
C       AND ALL VALUES OF THE MACT ARRAY EQUAL TO NEGATIVE
C       NUMBERS (I.E. NO SPECIES CONSTRAINTS; SEE AMAIN2.F
C       AND THE DICTIONARY). THUS ALL SPECIES ARE PRESENT
C       (INACTIVE) AND THEY ALL HAVE CORRESPONDING ELEMENTS IN
C       THE JI ARRAY. THEREFORE THERE ARE NO ELEMENTS
C       (INITIALLY) IN THE JA ARRAY.
C
C
        DO 14 I=1,NS
        DO 13 J=1,NS
        Q(J,I)=0.0D0
   13   CONTINUE
        JA(I)=0
        JI(I)=I
        MACT(I)=-I
        Q(I,I)=1.0D0
   14   CONTINUE
C
C ---------------------------------------------------------------------
C       QR FACTOR CONSTRAINT MATRIX (SEE STRANG (1980))
C
        NC=0
        DO 22 MM=1,M0
        MP1=NC+1
        SIG=0.0D0
        NODEG=.FALSE.
        DO 15 I=MP1,NS
        V(I)=R(I,MM)
        R(I,MM)=0.0D0
        IF(DABS(V(I)).GT.DEL4) NODEG=.TRUE.
        SIG=SIG+V(I)*V(I)
```

```
   15   CONTINUE
        IF(NODEG) THEN
           NC=MP1
           SIG=DSIGN(DSQRT(SIG),V(NC))
           IF(NC.EQ.NS) THEN
              R(NC,MM)=V(NC)
           ELSE
              V(NC)=V(NC)+SIG
              PI=1.0D0/(V(NC)*SIG)
              R(NC,MM)=-SIG
              R11(NC,MM)=-SIG
              DO 18 J=1,NS
              TAU=0.0D0
              DO 16 I=NC,NS
              TAU=TAU+Q(J,I)*V(I)
   16         CONTINUE
              TAU=TAU*PI
              DO 17 I=NC,NS
              Q(J,I)=Q(J,I)-TAU*V(I)
              IF(DABS(Q(J,I)).LT.DEL4) Q(J,I)=0.0D0
   17         CONTINUE
   18         CONTINUE
              IF(MM.LT.M0) THEN
                 MMP1=MM+1
                 DO 21 J=MMP1,M0
                 TAU=0.0D0
                 DO 19 I=NC,NS
                 TAU=TAU+R(I,J)*V(I)
   19            CONTINUE
                 TAU=TAU*PI
                 DO 20 I=NC,NS
                 R(I,J)=R(I,J)-TAU*V(I)
   20            CONTINUE
                 R11(NC,J)=R(NC,J)
   21            CONTINUE
              END IF
           END IF
        END IF
   22   CONTINUE
C
        NCP1=NC+1
        DO 27 I=1,M0
        IF(I.GT.MX) THEN
           R11(NCP1,I) = B(I)
        ELSE
C          R11(NCP1,I)=1.0D+20
        END IF
   27   CONTINUE
        NINACT=NS
        ND=0
C
C       FIND SPECIES WHICH ARE REDUNDANT WITH CHARGE AND MASS
C       BALANCE CONSTRAINTS.
C
```

```
C
      DO 28 J=1,NS
      REDUN=.TRUE.
C
C      IF THE COLUMNS OF THE Q3 MATRIX (COLUMNS NCP1 THROUGH
C      NS OF THE FULL Q MATRIX) ARE ALL VERY CLOSE TO ZERO FOR
C      A GIVEN SPECIES  (I.E. FOR A GIVEN ROW OF THE Q3 MATRIX),
C      THAT SPECIES IS DEGENERATE.
C
C
C
      DO 29 I=NCP1,NS
      IF(DABS(Q(J,I)).GT.DEL4) REDUN=.FALSE.
   29 CONTINUE
C
C     IF THE NUMBER OF CHARGE AND MASS BALANCE CONSTRAINTS
C     EQUALS THE NUMBER OF SPECIES, ALL SPECIES ARE DEGENERATE
C     (THERE IS NO Q3 MATRIX).
C
C
      IF(NS.EQ.NC) REDUN=.TRUE.
      IF(REDUN) THEN
        ND=ND+1
        JA(ND)=J
        IT=-MACT(J)
        JI(IT)=JI(NINACT)
        MACT(JI(IT))=-IT
        NINACT=NINACT-1
        MACT(J)=ND
      END IF
   28 CONTINUE
      NSC=NS
      NCC=NC
      NDC=ND
      NC0C=NC
      N00C=M0
      N0EC=MX
      NP0C=NP0
      NPMAX=NC-MX
      WRITE(6,2008) NC,NPMAX
 2008 FORMAT(5X,'RANK=',I3,5X,'MAXIMUM PHASES=',I3)
      RETURN
C
 5999 FORMAT(80('-'),/,A24,' WITHHELD',/,80('-'))
      END
```

```
C-----------------------------------------------------------------------
C
C       THIS SUBROUTINE READS IN THE NONIDEAL SOLUTION
C       PARAMETERS.
C
C       THE INPUT ORDER IS: H2O,CATIONS,ANIONS,NEUTRALS.
C
C-----------------------------------------------------------------------
      SUBROUTINE DATAP(KPP)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      DOUBLE PRECISION IONIC,MOLAL
      INTEGER*2 Z,ZMAX
      CHARACTER*24 A1
      DOUBLE PRECISION DUMPSI(100)
      COMMON/LABEL/Z(96),U0RT(96),A1(96)
      COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1,NP0C,N00C
      COMMON/AQSOL/JP(46),JMC,JXC,JMA,JXA,JMB,JXB,ALPHA(3)
     1,BETA,B0(18,18),B1(18,18),B2(18,18),CCA(18,18),TA(153)
     2,PSIA(153,18),TC(153),PSIC(153,18),MOLAL(46)
     3,F1(3),F2(3),F3(3),ZSUM
C-----------------------------------------------------------------------
C
C    JP (J)=I: SPECIES J IN THE AQUEOUS SOLUTION IS THE ITH CATION,
C            ANION, OR NEUTRAL SPECIES. THIS IS USED TO ACCESS THE
C            SOLUTION PARAMETER ARRAYS AND MATRICES.
C
C            EXAMPLE:
C            SPECIES (J)    JP(J)
C            H2O       1      0
C            NA        2      1
C            K         3      2
C            CL        4      1
C            SO4       5      2
C            HCO3      6      3
C            CO2(AQ)   7      1
C
C    JMC: MINIMUM CATION SPECIES NUMBER
C
C    JXC: MAXIMUM CATION SPECIES NUMBER
C
C    JMA: MINIMUM ANION SPECIES NUMBER
C
C    JXA: MAXIMUM ANION SPECIES NUMBER
C
C    JMB: MINIMUM NEUTRAL SPECIES NUMBER
C
C    JXB: MAXIMUM NEUTRAL SPECIES NUMBER
C
C    ALPHA (J): THE THREE PITZER ALPHAS (ALPHA FOR 1-2 AND LESS,
C                ALPHA FOR GREATER THEN 1-2, AND ALPHA FOR THE
C                BETA2 TERM.
C
```

```
C     BETA: B VALUE IN PITZER LIMITING LAW.
C
C     B0 (I,J): PITZER'S B0 VALUE FOR CATION I AND ANION J
C             (REFERENCED BY JP)
C
C     B1 (I,J): PITZER'S B1 VALUE FOR CATION I AND ANION J
C             (REFERENCED BY JP)
C
C     B2 (I,J): PITZER'S B2 VALUE FOR CATION I AND ANION J
C             (REFERENCED BY JP)
C
C     CCA (I,J): PITZER'S CPHI VALUE FOR CATION I AND ANION J
C              (REFERENCED BY JP)
C
C     TA (J) : PITZER'S THETA VALUE FOR THE JTH ANION-ANION THETA
C            REFERENCED BY JP.
C
C            EXAMPLE: ANIONS CL,SO4, AND HCO3-:
C            TA(1) = THETA CL-SO4
C            TA(2) = THETA CL-HCO3
C            TA(3) = THETA SO4-HCO3
C
C      PSIA (J,I): PITZER'S PSI VALUE FOR THE JTH ANION-ANION AND ITH
C                CATION PSI REFERENCED BY JP.
C                EXAMPLE: CATIONS NA AND K, ANIONS CL, SO4, HCO3
C
C                PSIA(1,1) = PSI CL-SO4-NA
C                PSIA(1,2) = PSI CL-SO4-K
C                PSIA(2,1) = PSI CL-HCO3-NA
C                PSIA(2,2) = PSI CL-HCO3-K
C                PSIA(3,1) = PSI SO4-HCO3-NA
C                PSIA(3,2) = PSI SO4-HCO3-K
C
C     TC (J) : PITZER'S THETA VALUE FOR THE JTH CATION-CATION
C            THETA REFERENCED BY JPA (ANALAGOUS TO TA ABOVE).
C
C     PSIC (J,I) : PITZER'S PSI VALUE FOR THE JTH CATION-CATION AND
C                ITH ANION PSI REFERENCED BY JP (ANALAGOUS TO PSIA
C                ABOVE).
C
C     MOLAL (J) : THE MOLALITY OF SPECIES J.
C
C     F1 (J) : THE EXPONENTIAL TERM IN PITZER'S OSMOTIC
C              COEFFICIENT EXPRESSION FOR EACH VALUE OF ALPHA
C            (E.G. EXP(-ALPHA*SQRT(I))).
C
C     F2 (J) : THE "G(X)" TERM IN PITZER'S LOG OF THE ACTIVITY
C            COEFFICIENT EXPERSSIONS FOR EACH VALUE OF ALPHA.
C            (X = -ALPHA*SQRT(I)).
C
C     F3 (J) : THE DERIVATIVE OF G(X) WITH RESPECT TO IONIC
C              STRENGTH.
C
C     ZSUM :  THE SUM OF THE ABSOLUTE VALUES OF THE CHARGE ON
```

```
C           EACH SPECIES TIMES ITS MOLALITY. (THIS TERM
C           MULTIPLIES THE CPHI IN SOME INSTANCES IN THE PITZER
C           EQUATIONS)
C
C-----------------------------------------------------------------
      COMMON/ELCT/APHI,IONIC,ETH(8,8),ETHP(8,8),ETHP2(8,8),ZMAX
C-----------------------------------------------------------------
C
C    APHI : A PHI.
C
C    IONIC : THE IONIC STRENGTH.
C
C    ETH(I,J) : THE ELECTROSTATIC CONTRIBUTION TO THETA IN THE
C               PITZER EQUATIONS. (THETA = ETH(J,J) + THETA). I AND J
C               ARE THE REPRESENT THE ABSOLUTE VALUES OF THE
C               LIKE CHARGES.
C
C    ETHP (I,J) : THE FIRST DERIVATIVE OF ETH WITH RESPECT TO
C                 IONIC STRENGTH.
C
C    ETHP2 (I,J) : THE SECOND DERIVATIVE OF ETH WITH RESPECT TO
C                  IONIC STRENGTH.
C
C    ZMAX : THE ABSOLUTE VALUE OF THE HIGHEST CHARGE FOUND
C         ON ANY ONE THE AQUEOUS SPECIES.
C
C-----------------------------------------------------------------
C SAS
      COMMON/DBASE/NCAT(18),NANI(18),ICAT,IANI,NCD,NAD,NND,TEMP
C-----------------------------------------------------------------
C
C    NCAT : MATRIX OF DATABASE SPECIES NUMBERS OF CATIONS IN
C             THE SYSTEM
C
C    NANI : ANALOGOUS TO NCAT
C
C    ICAT : TOTAL NUMBER OF CATIONS IN THE CURRENT SYSTEM
C
C    IANI : TOTAL NUMBER OF ANIONS IN THE CURRENT SYSTEM
C
C    NCD : TOTAL NUMBER OF CATIONS IN THE DATABASE (READ FROM
C          DATABASE)
C
C    NAD : TOTAL NUMBER OF ANIONS IN THE DATABASE
C
C    NND : TOTAL NUMBER OF NEUTRALS IN THE DATABASE
C
C    TEMP : SYSTEM TEMPERATURE
C
C-----------------------------------------------------------------
C SAS

 1000   FORMAT(I3,1X,I3,3X,4F14.8)
 1003   FORMAT(I3,1X,I3,8X, F9.5, 2(/ 7X,9F8.4))
```

```
 1002   FORMAT(1X,'DATABASE ERROR IN PITZER SECTION. MISSING
    1 LINES',2I4)
 1004   FORMAT(2X,5F14.6)
 1005   FORMAT(/////////)
 1006   FORMAT()
 1007   FORMAT(I3)
 2001   FORMAT(//,10X,'NON-IDEAL AQUEOUS ELECTROLYTE SOLUTION
DATA')
 3000   FORMAT(/,5X,'APHI=',F8.5,5X,'BETA=',F8.3,5X,'ALPHA=',3F8.3)
 3001   FORMAT(/,24X,'B0',8X,'B1',8X,'B2',7X,'CPHI')
 3002   FORMAT(1X,A8,1X,A8,4F10.5)
 3003   FORMAT(/,19X,'THETA',4X,6(A8),(/ 32X,6A8))
 3004   FORMAT(2A8,7(F8.4),(/ 28X,6F8.4))
 3011   FORMAT(/,5X,'DAVIES CONSTANT A=',F8.6)
      JTYPE=IABS(KTYP(KPP))
      IF(JTYPE.LE.1.OR.JTYPE.EQ.2.AND.KPP.GT.1) RETURN
C
C      IF THE PHASE IS A PURE MINERAL OR AN IDEAL SOLUTION NON-
C      AQUEOUS  PHASE, EXIT.
C
      JM=JMIN(KPP)
      JX=JMAX(KPP)
      JMC=JM+1
      JXC=0
      JXA=0
      JXB=0
      ZMAX=0
      IF(JX-JM.GT.46) THEN
           WRITE(6,9988) JX-JM+1
           GO TO 600
      END IF
 9988   FORMAT('TOO MANY AQUEOUS SPECIES ',I5)
      DO 5 J=JMC,JX
      IF(Z(J)) 2,3,4
   2  JXA=J
      IF(JXB.GT.0) GO TO 600
      IF(ZMAX.LT.-Z(J)) ZMAX=-Z(J)
      JP(J)=J-JXC
      IF(JP(J).GT.18) THEN
           WRITE(6,9989) J,JP(J)
           GO TO 600
      END IF
 9989   FORMAT('TOO MANY ANIONS ',2I5)
      GO TO 5
   3  JXB=J
      JP(J)=J-JXA
      IF(JP(J).GT.8) THEN
           WRITE(6,9990) J,JP(J)
           GO TO 600
      END IF
 9990   FORMAT('TOO MANY NEUTRALS ',2I5)
      GO TO 5
   4  JXC=J
      IF(JXA.GT.0) GO TO 600
```

```
      IF(JXB.GT.0) GO TO 600
      IF(ZMAX.LT.Z(J)) ZMAX=Z(J)
      JP(J)=J-JM
      IF(JP(J).GT.18) THEN
           WRITE(6,9991) J,JP(J)
           GO TO 600
      END IF
 9991  FORMAT('TOO MANY CATIONS ',2I5)
    5  CONTINUE
      JMA=JXC+1
      JMB=JXA+1
      IF(JXA*JXC.EQ.0) GO TO 600
      IF(JTYPE.LE.2) RETURN
C
C      IF THE PHASE IS AN AQUEOUS IDEAL SOLUTION PHASE, EXIT
C SAS
    9  READ(5,1003) NSPEC
      IF(NSPEC.NE.900) GO TO 9
C      ADVANCING DATA FILE TO PITZER SECTION
C
C
C      READ APHI POLYNOMIAL
C
      READ(5,1004) APHI0,APHI1,APHI2,APHI3,APHI4
      APHI=APHI0 + APHI1*TEMP + APHI2*TEMP**2 + APHI3*TEMP**3
     1 +APHI4*TEMP**4

C      IF DAVIES MODEL IS USED REWIND DATA FILE AND RETURN
      IF(JTYPE.EQ.3) THEN
         REWIND(5)
         READ(5,1005)
         WRITE(6,3011) APHI*3.0D0/DLOG(10.0D0)
         RETURN
      END IF


C      THIS SECTION READS IN PITZER DATA
C      READ B, ALPHA0, ALPHA1, AND ALPHA2
      READ(5,1004) BETA,(ALPHA(I),I=1,3)
      READ(5,1006)
      WRITE(6,2001)
      WRITE(6,3000) APHI,BETA,(ALPHA(I),I=1,3)
      WRITE(6,3001)
      NC=JP(JXC)
      NA=JP(JXA)
C
C
C      READ BETA0, BETA1, BETA2, AND CPHI
C      AND CONVERT CPHI TO C.
C
      II=0
      DO 10 I=JMC,JXC
      JJ=0
      II=II+1
```

```
      DO 10 J=JMA,JXA
      JJ=JJ+1
 11   READ(5,1000) NSPECC,NSPECA,B0(JP(I),JP(J)),B1(JP(I),JP(J))
     1,B2(JP(I),JP(J)),CPHI
      IF (NSPECC.EQ.NCAT(II)) THEN
         IF (NSPECA.EQ.NANI(JJ)) THEN
            WRITE(6,3002) A1(I),A1(J)
     1      ,B0(JP(I),JP(J)),B1(JP(I),JP(J)),B2(JP(I)
     2      ,JP(J)),CPHI
            CCA(JP(I),JP(J))=CPHI/SQRT(-4.0E0*Z(I)*Z(J))
            GO TO 10
         ELSE IF (NSPECA.GT.NANI(JJ)) THEN
            WRITE(*,1002)
            STOP
         ELSE
            GO TO 11
         END IF
      ELSE IF (NSPECC.GT.NCAT(II)) THEN
         WRITE(*,1002)
      ELSE
         GO TO 11
      END IF
  10  CONTINUE

C MOVE POINTER TO BEGINING OF THETA/PSI DATA
  12  READ(5,1007) NSPECC
      IF (NSPECC.NE.901) GO TO 12

      L=0
      JXCM1=JXC-1
      IF(JXCM1.LT.JMC) GO TO 701
      WRITE(6,3003)(A1(K),K=JMA,JXA)
C
C      READ CATION-CATION THETAS AND CATION-CATION-ANION PSI'S
C
      DO 30 I=1,ICAT-1
      DO 30 J=I+1,ICAT
      L=L+1

  31  READ(5,1003) NSPEC1,NSPEC2,TC(L),(DUMPSI(K),K=1,NAD)
      IF (NSPEC1.EQ.NCAT(I)) THEN
         IF(NSPEC2.EQ.NCAT(J)) THEN
            DO 32 K=1,IANI
            PSIC(L,K)=DUMPSI(NANI(K)-199)
 32         CONTINUE
            WRITE(6,3004) A1(I+1),A1(J+1)
     1         ,TC(L),(PSIC(L,K),K=1,NA)
            GO TO 30
         ELSE IF (NSPEC2.GT.NCAT(J)) THEN
            WRITE(*,1002)
            STOP
         ELSE
            GO TO 31
         END IF
```

```
      ELSE IF (NSPEC1.GT.NCAT(I)) THEN
         WRITE(*,1002)
         STOP
      ELSE
         GO TO 31
      END IF
   30 CONTINUE

  701 CONTINUE

C  MOVE POINTER TO NEXT SECTION
   33 READ(5,1007) NSPEC1
      IF (NSPEC1.NE.902) GO TO 33

      JXAM1=JXA-1
      IF(JXAM1.LT.JMA) GO TO 700
      WRITE(6,3003)(A1(K),K=JMC,JXC)
      L=0
C
C     READ ANION-ANION THETAS AND ANION-ANION-CATION PSI'S
C
      DO 40 I=1,IANI-1
      DO 40 J=I+1,IANI
      L=L+1

   41 READ(5,1003) NSPEC1,NSPEC2,TA(L),(DUMPSI(K),K=1,NCD)
      IF (NSPEC1.EQ.NANI(I)) THEN
         IF(NSPEC2.EQ.NANI(J)) THEN
            DO 42 K=1,ICAT
            PSIA(L,K)=DUMPSI(NCAT(K)-99)
   42       CONTINUE
            WRITE(6,3004) A1(I+1+ICAT)
     1           ,A1(J+1+ICAT),TA(L)
     2           ,(PSIA(L,K),K=1,NC)
            GO TO 40
         ELSE IF (NSPEC2.GT.NANI(J)) THEN
            WRITE(*,1002)
            STOP
         ELSE
            GO TO 41
         END IF
      ELSE IF (NSPEC1.GT.NANI(I)) THEN
         WRITE(*,1002)
         STOP
      ELSE
         GO TO 41
      END IF
   40 CONTINUE
      L=0

  700 CONTINUE

C     REWIND DATA FILE AND ADVANCE
      REWIND(5)
```

```
      READ(5,1005)

      RETURN
C SAS

  600   WRITE(6,1001)
 1001   FORMAT(' DATAP READ ERROR  ')
      STOP
      END
```

```
C-----------------------------------------------------------------
C
C     THIS SUBROUTINE IS THE MAIN PROJECTED NEWTON MINMIZATION
C     PROCEDURE
C
C-----------------------------------------------------------------
      SUBROUTINE PRIMAL(IRUN,ERROR)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 Z
      CHARACTER*24 A1
      COMMON/CONST/Q(96,96),MACT(96),JA(96),JI(96),NSC,NCC,NDC,NC0C
     1,N0EC
      COMMON/LABEL/Z(96),U0RT(96),A1(96)
      COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1,NP0C,N00C
      COMMON/SEARCH/X(96),P(96),G(96),W0(96),H(64,64),B(96)
      COMMON/PRINT/PP,PP2,PD
      DIMENSION YS(96),JDG(96),KPH(96),KREM(96)
      LOGICAL SUBMIN,ZOUTEN(96),POSDEF,REMPHS,ADDPHS,SENS
      LOGICAL NORMAL,ERROR,REFIN,REMP,PREM,GRADF,GRADP
      LOGICAL REMDEG
      LOGICAL RR
      LOGICAL PP,PP2,PD,FIRST
      PARAMETER(DELMP=0.5D0**55)
      PARAMETER(DELLB=1.0D-20)
      PARAMETER(DEL4=1.0D-13,DEL8=4.0D0*DELMP*(2.0D0**5))
      PARAMETER(DEL2=1.0D-12)
      PARAMETER(DEL2P=DEL2*(1.0D0+DEL2),DEL2M=DEL2*(1.0D0-DEL2))
      PARAMETER(DEL1=1.0D-07,DEL5=0.1D0,DEL6=1.0D-04,DEL7=1.0D-07)
      PARAMETER(NFUN=1000)
C
C ----------------------------------------------------------------
C       INITALIZATION FOR MINIMIZATION
C
        FIRST=.FALSE.
        IF(ERROR) FIRST=.TRUE.
        ERROR=.FALSE.
        NS=NSC
        NC0=NC0C
        NSP1=NS+1
        NC0P1=NC0+1
        NPHAS0=NP0C
        N00=N00C
C
C       CALCULATE SCALE FACTOR
C
        SCALE=1.0D+20
        DO 2 I=1,N00
        TAU=0.0D0
        DO 1 J=1,NC0
        TAU=TAU+B(J)*R11(J,I)
    1   CONTINUE
        IF(TAU*SCALE.GT.R11(NC0P1,I)) SCALE=R11(NC0P1,I)/TAU
```

```
   2  CONTINUE
      WRITE(6,5000) IRUN,SCALE
5000  FORMAT(//,'   STEP',I5,'  SCALE FACTOR=',1PD12.4)
      IF(SCALE.EQ.1.0D+20) THEN
         ERROR=.TRUE.
         RETURN
      END IF
C
C     FIND INITIAL PHASE ASSEMBLAGE, INITIALIZE MOLE NUMBER AND
C     DIRECTION VECTORS, EVALUATE INITIAL FREE ENERGY, AND
C     DELETE OR ADD SPECIES CONSTRAINTS
C
      ITER=0
      ISENS=0
      IFUN=0
      ISUB=0
      IREFIN=0
      NPHAS=0
      GFE0=0.0D0
      PTGA=0.0D0
      ALPHA=0.0D0
      NDG=0
      DO 5 K=1,NPHAS0
      IF(KTYP(K).GT.1) THEN
         NPHAS=NPHAS+1
         KPH(NPHAS)=K
         JM=JMIN(K)
         JX=JMAX(K)
         DO 3 J=JM,JX
         ZOUTEN(J)=.FALSE.
         X(J)=X(J)*SCALE
2323      FORMAT(I5,1PD25.16)
         P(J)=0.0D0
         Q(J,NSP1)=0.0D0
         IF(X(J).GT.DEL2P) THEN
            IF(MACT(J).GT.NC0) THEN
              WRITE(*,*) 'CALLING CONDEL @ LN 91'
              CALL CONDEL(J)
            END IF
         ELSE
            IF(X(J).LT.DELLB) X(J)=DELLB
            IF(MACT(J).LT.0) THEN
              WRITE(*,*) 'CALLING CONADD @ LN 97'
              CALL CONADD(J)
            ELSE IF(MACT(J).LT.NC0P1) THEN
              NDG=NDG+1
              JDG(NDG)=J
            END IF
         END IF
   3     CONTINUE
         WRITE(*,*) 'CALLING GRAD FROM PRIMAL @ LN 105'
         CALL GRAD(K,ALPHA,PTGA,GFE0)
      ELSE IF(KTYP(K).EQ.1) THEN
         NPHAS=NPHAS+1
```

```
            KPH(NPHAS)=K
            J=JMIN(K)
            X(J)=X(J)*SCALE
            P(J)=0.0D0
            Q(J,NSP1)=0.0D0
            ZOUTEN(J)=.FALSE.
            IF(X(J).LT.DELLB) X(J)=DELLB
            IF(MACT(J).GT.NC0) THEN
              WRITE(*,*) 'CALLING CONDEL @ LN 117'
              CALL CONDEL(J)
            END IF
            GFE0=GFE0+X(J)*G(J)
         ELSE
            JM=JMIN(K)
            JX=JMAX(K)
            DO 4 J=JM,JX
            ZOUTEN(J)=.FALSE.
            X(J)=0.0D0
            P(J)=0.0D0
            Q(J,NSP1)=0.0D0
            IF(MACT(J).LT.0) THEN
              WRITE(*,*) 'CALLING CONADD @ LN 130'
              CALL CONADD(J)
            ELSE IF(MACT(J).LT.NC0P1) THEN
              NDG=NDG+1
              JDG(NDG)=J
            END IF
    4       CONTINUE
         END IF
    5    CONTINUE
         IF(NDG.GT.0) THEN
            DO 6 JD=1,NDG
            J=JDG(JD)
            IF(MACT(J).LT.0) THEN
               WRITE(*,*) 'CALLING CONADD @ LN 143'
               CALL CONADD(J)
            END IF
    6       CONTINUE
         END IF
         NORMAL=.FALSE.
         ADDPHS=.FALSE.
         REMPHS=.FALSE.
         REFIN=.FALSE.
C
C ---------------------------------------------------------------
C        SENSITIVITY STEP CALCULATION  - TEST FOR FEASIBILITY
C
 1000    SENS=.FALSE.
         DO 9 I=1,NC0
         TAU=B(I)*SCALE
         DO 8 K=1,NPHAS
         JM=JMIN(KPH(K))
         JX=JMAX(KPH(K))
         DO 7 J=JM,JX
```

```
      TAU=TAU-X(J)*Q(J,I)
    7 CONTINUE
    8 CONTINUE
      YS(I)=TAU
      IF(DABS(TAU).GT.DEL4) SENS=.TRUE.
    9 CONTINUE
C
C     IF FEASIBLE COMPUTE NORMAL DESCENT DIRECTION, OTHERWISE
C     COMPUTE A SENSITIVITY DIRECTION TO FEASIBLE DOMAIN
C
      IF(SENS) THEN
        ISENS=ISENS+1
        NCMIN=NC0P1
        NJMIN=1
        IF(ISENS.GT.100) THEN
            WRITE(6,9991) ISENS
 9991        FORMAT(' PRIMAL 9991 (ISENS)=',I5)
            ERROR=.TRUE.
            RETURN
        END IF
      ELSE
        NORMAL=.TRUE.
        GO TO 1002
      END IF
C
C -------------------------------------------------------------------
C     MODIFIED STEP DIRECTION CALCULATION
C
 1001  NC=NCC
      ND=NDC
      NU=NS-NC
      NUP1=NU+1
      NUT=NUP1
      NUP2=NU+2
      NINACT=NU-ND+NC0
      PTG=0.0D0
      AMAX=1.0D0
      J0=0
      IF(NC.GE.NCMIN) THEN
        DO 11 I=NCMIN,NC
        JJ=JA(I)
        IF(DABS(P(JJ)).GT.DELLB) THEN
          TAU=P(JJ)
          Q(JJ,NSP1)=TAU
          PTG=PTG+TAU*G(JJ)
        ELSE
          TAU=0.0D0
        END IF
        IF(I.GT.NJMIN) THEN
          IM1=I-1
          DO 10 J=NJMIN,IM1
          TAU=TAU-Q(JJ,J)*YS(J)
   10     CONTINUE
        END IF
```

```
      YS(I)=TAU/Q(JJ,I)
11    CONTINUE
    END IF
    IF(ND.GT.0) THEN
      DO 14 JD=1,ND
      J=JA(JD)
      TAU=0.0D0
      DO 12 I=NJMIN,NC
      TAU=TAU+Q(J,I)*YS(I)
12    CONTINUE
      IF(KTYP(KSPEC(J)).GT.1) THEN
        PI=X(J)+AMAX*TAU
        IF(X(J)+AMAX*TAU.LT.DELLB) THEN
              IF(PP) THEN
          WRITE(6,3867) J,X(J),TAU
3867       FORMAT(' DEG SPEC BOUNDARY',I5,2F25.20)
              END IF
          IF(X(J).GT.DELLB) THEN
            IF(REMPHS) GO TO 1012
            AMAX=(DELLB-X(J))/TAU
            J0=J
          ELSE IF(DABS(TAU).LT.DEL4.AND..NOT.REFIN) THEN
            TAU=0.0D0
          ELSE IF(TAU.LT.0.0D0) THEN
            IF(REMPHS) GO TO 1012
              IF(PP) THEN
               WRITE(6,3876) J,TAU,X(J)
3876            FORMAT('  ND ZERO STEP',I5,1P2D25.16)
              END IF
            AMAX=0.0D0
            J0=J
          END IF
        END IF
      ELSE IF(KTYP(KSPEC(J)).EQ.1) THEN
        IF(X(J)+AMAX*TAU.LT.0.0D0) THEN
          AMAX=-X(J)/TAU
          J0=J
        END IF
      ELSE IF(DABS(TAU-P(J)).LT.DEL4) THEN
        TAU=P(J)
      ELSE IF(REMPHS) THEN
        GO TO 1012
      ELSE IF(SENS) THEN
        IF(TAU.LT.0) THEN
          J0=J
        ELSE
          K0=KSPEC(J)
          NPHAS=NPHAS+1
          KTYP(K0)=-KTYP(K0)
          KPH(NPHAS)=K0
          IF(KTYP(K0).GT.1) THEN
            JM=JMIN(K0)
            JX=JMAX(K0)
            DO 13 J2=JM,JX
```

```
                X(J2)=DELLB
 13             CONTINUE
                ALPHA=0.0D0
                WRITE(*,*) 'CALLING GRAD FROM PRIMAL @ LN 274'
                CALL GRAD(K0,ALPHA,PTG,GFE0)
              END IF
            END IF
          ELSE
            WRITE(6,9393) ADDPHS,REFIN,J
9393        FORMAT(' ADDPHS MOD STEP ERROR',3I5)
            ERROR=.TRUE.
            RETURN
          END IF
          Q(J,NSP1)=TAU
          P(J)=TAU
          PTG=PTG+TAU*G(J)
 14       CONTINUE
          IF(.NOT.SENS) THEN
            IF(J0.GT.0) THEN
              IF(X(J0)+AMAX*P(J0).LE.X(J0)*DEL2) THEN
                AMAX=X(J0)*(DEL2-1.0D0)/P(J0)
                IF(X(J0)+AMAX*P(J0).GT.DELLB) J0=0
              END IF
            END IF
          ELSE IF(J0.GT.0) THEN
            J1=0
            DO 43 II=NC0P1,NC
            I=(NC0P1+NC)-II
            J2=JA(I)
            TAU=Q(J0,I)
            IF(I.LT.NC) THEN
              IP1=I+1
              DO 63 I2=IP1,NC
              TAU=TAU-YS(I2)*Q(JA(I2),I)
 63           CONTINUE
            END IF
            YS(I)=TAU/Q(J2,I)
            IF(YS(I).GT.DEL4.AND.J2.GT.J1) THEN
              J1=J2
            END IF
 43         CONTINUE
            IF(J1.EQ.0) THEN
              WRITE(6,9999) J0
9999          FORMAT(I5,' PRIMAL 9999  (NO FEASIBLE SOLUTION)')
              ERROR=.TRUE.
              RETURN
            END IF
            WRITE(*,*) 'CALLING CONDEL @ LN 318'
            CALL CONDEL(J1)
            WRITE(*,*) 'CALLING CONADD @ LN 320'
            CALL CONADD(J0)
            P(J0)=0.0D0
            Q(J0,NSP1)=0.0D0
                  IF(PP) THEN
```

```
                    WRITE(6,5006) J0,J1
 5006               FORMAT(' PRIMAL SENSITIVITY PIVOT',2I5)
                       END IF
                 GO TO 1000
              END IF
           END IF
           IF(NINACT.EQ.0) GO TO 1006
           DO 62 JJ=1,NINACT
           J=JI(JJ)
           TAU=0.0D0
           DO 61 I=NJMIN,NC
           TAU=TAU+Q(J,I)*YS(I)
   61      CONTINUE
           Q(J,NSP1)=TAU
   62      CONTINUE
           DO 16 I1=2,NUP2
           IM1=I1-1
           DO 15 I2=1,IM1
           H(I2,I1)=0.0D0
   15      CONTINUE
   16      CONTINUE
           H(NUP1,NUP2)=-1.0D0
           GO TO 1005
C
C ---------------------------------------------------------------------
C          NEWTON STEP DIRECTION CALCULATION
C
 1002      NC=NCC
           ND=NDC
           NU=NS-NC
           NUT=NU
           NUP1=NU+1
           NUP2=NU+2
           NINACT=NU-ND+NC0
           IF(NU.EQ.0) THEN
              Q3MAX=0.0D0
              WMIN=-DEL7
              SUBMIN=.TRUE.
              ISUB=ISUB+1
                 IF(PP.OR.PD) WRITE(6,6050) ISUB,ITER,IFUN,NU
 6050             FORMAT('SUBSPACE MINIMUM ',4I5)
              POSDEF=.TRUE.
              GO TO 1004
           END IF
           POSDEF=.FALSE.
C
C          REDUCED GRADIENT VECTOR CALCULATION
C
 1003      SIG=0.0D0
           PTPMAX=0.0D0
           Q3MAX=0.0D0
           PTG=0.0D0
           AMAX=0.999D0
           J0=0
```

```
      SIG=0.0D0
      DO 19 L=1,NU
      TAU=0.0D0
      LQ=L+NC
      DO 18  JJ=1,NINACT
      TAU=TAU+G(JI(JJ))*Q(JI(JJ),LQ)
   18 CONTINUE
      YS(L)=TAU
      IF(DABS(TAU).GT.Q3MAX) Q3MAX=DABS(TAU)
   19 CONTINUE
      IF(Q3MAX.LT.DEL1)  THEN
         SUBMIN=.TRUE.
         ISUB=ISUB+1
                  IF(PP.OR.PD) THEN
            WRITE(6,6050) ISUB,ITER,IFUN,NU
                  END IF
         WMIN=-DEL7
      ELSE
         SUBMIN=.FALSE.
         WMIN=-DMAX1(Q3MAX,DEL7)
      END IF
C
C     EVALUATE MULTIPLIERS  (IF POSDEF)
C
 1004 IF(POSDEF) THEN
      NP=NPHAS
      WRITE(*,*) 'CALLING DUAL @ LN 405'
      CALL DUAL(SUBMIN,ADDPHS,ZOUTEN,ERROR,WMIN,NP,NCMIN,KPH)
         IF(ERROR) THEN
            RETURN
         ELSE IF(ADDPHS) THEN
            NPSAV=NPHAS
            NPHAS=NP
            NJMIN=NCMIN
            NORMAL=.FALSE.
            GO TO 1001
         ELSE IF(SUBMIN) THEN
            IREFIN=IREFIN+1
            WRITE(*,*) 'CALLING REFINE @ LN 417'
            CALL REFINE(Q2MAX,NCMIN)
            IF(Q2MAX.GT.DEL1) THEN
               NJMIN=NCMIN
               NORMAL=.FALSE.
               REFIN=.TRUE.
               GO TO 1001
            ELSE
C
C              OPTIMAL SOLUTION FOUND WITHIN TOLERANCE OF DEL1
C
             IF(ITER.EQ.0) GFEA = GFE0
             IF(PP2) THEN
                WRITE(*,*) 'CALLING PHSPRT @ LN 430'
                CALL PHSPRT(SCALE,GFEA,ITER,IFUN,IREFIN,
     1                      Q2MAX,Q3MAX)
```

```
                END IF
                RETURN
              END IF
            END IF
          END IF
          DO 59 L=1,NU
          LP1=L+1
          DO 17 L2=1,L
          H(L2,LP1)=0.0D0
   17     CONTINUE
          H(L,NUP2)=YS(L)
   59     CONTINUE
C
C         PROJECTED HESSIAN MATRIX CALCULATION
C
 1005     DO 25 KP=1,NPHAS
          K=KPH(KP)
          IF(KTYP(K).GT.1) THEN
            JM=JMIN(K)
            JX=JMAX(K)
            WRITE(*,*) 'CALLING HESS FROM PRIMAL @ LN 452'
            CALL HESS(K)
            DO 24 L1=1,NU
            LQ1=L1+NC
            DO 21 J1=JM,JX
            TAU=0.0D0
            DO 20 J2=JM,JX
            JP=MAX0(J1,J2)
            TAU=TAU+H(JP,J1+J2-JP)*Q(J2,LQ1)
   20       CONTINUE
            YS(J1)=TAU
   21       CONTINUE
            DO 23 L2=L1,NUT
            LQ2=L2+NC
            TAU=H(L1,L2+1)
            DO 22 J1=JM,JX
            TAU=TAU+YS(J1)*Q(J1,LQ2)
   22       CONTINUE
            H(L1,L2+1)=TAU
   23       CONTINUE
   24       CONTINUE
          END IF
   25     CONTINUE
C
C         REDUCED DIRECTION VECTOR CALCULATION
C
          POSDEF=.TRUE.
          GAMMA=0.0D0
          ZETA=0.0D0
          THETA=0.0D0
          DO 27 I=1,NU
          IP1=I+1
          IF(DABS(H(I,IP1)).GT.GAMMA) GAMMA=DABS(H(I,IP1))
          IF(I.LT.NU) THEN
```

```
            DO 26 J=IP1,NU
            IF(DABS(H(I,J+1)).GT.ZETA) ZETA=DABS(H(I,J+1))
   26       CONTINUE
            IF(I.EQ.1) THETA=ZETA
         END IF
   27    CONTINUE
         DELTA=DEL8*DMAX1(1.0D0,GAMMA,ZETA)
         BETA2=1.0D0/DMAX1(GAMMA,DELTA,ZETA/NU)
         TAU=DMAX1(DABS(H(1,2)), DELTA, THETA*THETA*BETA2)
         IF(TAU.GT.H(1,2)) POSDEF=.FALSE.
         H(1,2)=1.0D0/TAU
         IF(NU.EQ.1) THEN
            H(1,3)=H(1,3)*H(1,2)
         ELSE
            DO 31 J=2,NU
            JM1=J-1
            JP1=J+1
            JP2=J+2
            PHI=H(J,JP1)
            DO 28 K=1,JM1
            PI=H(K,JP1)*H(K,K+1)
            PHI=PHI-PI*H(K,JP1)
            H(K,JP1)=PI
   28       CONTINUE
            THETA=0.0D0
            DO 30 I=JP2,NUP2
            TAU=H(J,I)
            DO 29 K=1,JM1
            TAU=TAU-H(K,JP1)*H(K,I)
   29       CONTINUE
            H(J,I)=TAU
            IF(I.LT.NUP2) THEN
               IF(DABS(TAU).GT.THETA) THETA=DABS(TAU)
            END IF
   30       CONTINUE
            SIG=DMAX1( DABS(PHI), DELTA, THETA*THETA*BETA2)
            IF(SIG.GT.PHI) POSDEF=.FALSE.
            H(J,JP1)=1.0D0/SIG
   31       CONTINUE
            H(NU,NUP2)=H(NU,NUP2)*H(NU,NUP1)
            DO 33 JMN=2,NU
            J=NUP1-JMN
            JP1=J+1
            TAU=H(J,NUP2)*H(J,JP1)
            DO 32 K=JP1,NU
            TAU=TAU-H(J,K+1)*H(K,NUP2)
   32       CONTINUE
            H(J,NUP2)=TAU
   33       CONTINUE
         END IF
C
C        TEST FOR ACCIDENTAL MAXIMUM
C
         IF(NORMAL.AND.SUBMIN) THEN
```

```
          IF(POSDEF) GO TO 1004
C
C              LOCAL MAXIMUM
C
          WRITE(6,9992)
 9992      FORMAT(' PRIMAL 9992  (LOCAL MAXIMUM)')
          ERROR=.TRUE.
          RETURN
        END IF
C
C       FEASIBLE DIRECTION VECTOR CALCULATION AND MAXIMUM STEP
C       LENGTH
C
        DO 35 JJ=1,NINACT
        J=JI(JJ)
        TAU=0.0D0
        DO 34 I=1,NUT
        TAU=TAU-Q(J,I+NC)*H(I,NUP2)
   34   CONTINUE
        P(J)=TAU
        PTG=PTG+TAU*G(J)
        IF(KTYP(KSPEC(J)).GT.1) THEN
          IF(X(J)+AMAX*TAU.LT.DEL2M) THEN
            AMAX=DMAX1( (DEL2-X(J))/TAU ,0.0D0)
            J0=J
          END IF
        ELSE
          IF(X(J)+AMAX*TAU.LT.0.0D0) THEN
            AMAX=DMAX1( -X(J)/TAU ,0.0D0)
            J0=J
          END IF
        END IF
   35   CONTINUE
C
C       TEST FOR DESCENT DIRECTIONS   (IERR=1)
C
 1006       SIG=0.0D0
            PI=0.0D0
            IF(PP) THEN
C            WRITE(*,*) 'CALLING QTEST @ LN 578'
C            CALL QTEST
            DO 890 I=1,NC0
            TAU=B(I)*SCALE
            TAU2=0.0D0
            DO 891 J=1,NS
            TAU=TAU-Q(J,I)*X(J)
            TAU2=TAU2+P(J)*Q(J,I)
  891       CONTINUE
            IF(DABS(TAU).GT.SIG) SIG=DABS(TAU)
            IF(DABS(TAU2).GT.PI) PI=DABS(TAU2)
  890       CONTINUE
            WRITE(6,894) SIG,PI
  894       FORMAT( 'FEASIBILITY TEST ',1P2D27.19)
            PI=-PTG
```

```
            TAU=-GFE0
            SIG=-WMIN
            DO 888 J=1,NS
            TAU=TAU+X(J)*G(J)
            PI=PI+P(J)*G(J)
            SIG=SIG+Q(J,NSP1)*G(J)
C           WRITE(6,8738) J,P(J),G(J)
 8738       FORMAT(I5,1P3D16.8)
  888       CONTINUE
            IF(.NOT.ADDPHS) SIG=0.0D0
            WRITE(6,8738) J0,PI,TAU,SIG
                 SIG=0.0D0
                 PHI=0.0D0
                 DO 896 KP=1,NPHAS
                 K=KPH(KP)
                 JM=JMIN(K)
                 JX=JMAX(K)
                    DO 960 J=JM,JX
                    IF(NORMAL) THEN
                       TAU=G(J)
                       IF(DABS(TAU).GT.PHI) PHI=DABS(TAU)
                    ELSE
                       TAU=0.0D0
                    END IF
                    IF(KTYP(K).GT.1) THEN
                        DO 961 J2=JM,JX
                        JP=MAX0(J,J2)
                        PI=H(JP,J+J2-JP)*P(J2)
                        TAU=TAU+PI
                        IF(DABS(PI).GT.PHI) PHI=DABS(PI)
  961                    CONTINUE
                    END IF
                    YS(J)=TAU
  960               CONTINUE
  896            CONTINUE
                 SIG=0.0D0
                 IF(NC.LT.NS) THEN
                    NCP1=NC+1
                    DO 963 I=NCP1,NS
                    TAU=0.0D0
                    DO 964 J=1,NS
                    TAU=TAU+Q(J,I)*YS(J)
  964               CONTINUE
                    IF(DABS(TAU).GT.SIG) SIG=DABS(TAU)
  963               CONTINUE
                 END IF
                 WRITE(6,962) SIG,PHI,POSDEF
  962            FORMAT(' P TEST ',1P2D25.17,I5)
            END IF
C
C
      IF(SENS) GO TO 1013
      IF(PTG.GT.0.0D0) THEN
         IF(NORMAL.OR.REFIN) THEN
```

```
            WRITE(6,9994) NORMAL,REFIN,PTG,Q3MAX,PTPMAX
 9994        FORMAT(' PRIMAL 9994   PTG > 0',2I5,1P3D12.4)
            ERROR=.TRUE.
            RETURN
         ELSE
            IERR=1
            GO TO 1012
         END IF
      END IF
C
C     CONSTRAINT ENCOUNTERED FOR FULL STEP
C       LIMIT MAXIMUM DECREASE OF SOLUTION PHASE VARIABLE TO
C         FACTOR OF DEL6
C
      IF(J0.GT.0.AND.NORMAL) THEN
         IF(KTYP(KSPEC(J0)).GT.1) THEN
            IF(MACT(J0).LT.0) THEN
               DELTA=DEL6
            ELSE
               DELTA=DEL4
            END IF
            IF((DELTA-1.0D0)*X(J0).GT.AMAX*P(J0)) THEN
               AMAX=(DELTA-1.0D0)*X(J0)/P(J0)
               IF(X(J0)+AMAX*P(J0).GT.DEL2P) J0=0
            END IF
         END IF
      END IF
C
C     EVALUATE FUNCTION VALUE AT MAXIMUM STEP
C
 1013 IFUN=IFUN+1
      IF(IFUN.GT.NFUN) THEN
            WRITE(6,9995) IFUN
 9995        FORMAT(' PRIMAL 9995 (NFUN EXCEEDED)',I6)
            ERROR=.TRUE.
            RETURN
      END IF
      ALPHA=AMAX
      PTGA=0.0D0
      GFEA=0.0D0
      DO 36 K=1,NPHAS
      KP=KPH(K)
      RR=.FALSE.
      IF(REMPHS) THEN
         DO 89 KK=1,NPREM
         IF(KP.EQ.KREM(KK)) RR=.TRUE.
   89    CONTINUE
      END IF
      IF(IABS(KTYP(KP)).GT.1.AND..NOT.RR) THEN
         WRITE(*,*) 'CALLING GRAD FROM PRIMAL @ LN 696'
         CALL GRAD(KP,ALPHA,PTGA,GFEA)
      ELSE IF(KTYP(KP).LT.-1) THEN
          JM=JMIN(KP)
          JX=JMAX(KP)
```

```
            DO 83 J=JM,JX
            GFEA=GFEA+(X(J)+ALPHA*P(J))*G(J)
            PTGA=PTGA+P(J)*G(J)
   83       CONTINUE
         ELSE
            J=JMIN(KP)
            GFEA=GFEA+(X(J)+ALPHA*P(J))*G(J)
            PTGA=PTGA+P(J)*G(J)
         END IF
   36    CONTINUE
         IF(SENS) GO TO 1009
C
C        IF FREE ENERGY IS REDUCED TAKE MAXIMUM STEP
C
         IF(GFEA.LT.GFE0) GO TO 1009
C
C        IF FREE ENERGY DECREASING AND MAXIMUM POSSIBLE
C        IMPROVEMENT IS INSIGNIFICANT COMPARED TO GFE0
C        TAKE MAXIMUM STEP
C
         IF(PTGA.LT.DELMP) THEN
           IF(DABS(AMAX*PTG).LT.DEL8*DABS(GFE0)) GO TO 1009
                DELG=(GFEA-GFE0)/DABS(GFE0)
                WRITE(6,9996) AMAX,PTG,PTGA,GFEA,DELG
 9996            FORMAT('PRIMAL 9996  (NON-CONVEXITY)',1P5D12.5)
                GO TO 1009
         END IF
C
C        LINESEARCH
C
         AMIN=0.0D0
         ETA=DEL5*DABS(PTG)
                        IF(PP) THEN
         WRITE(6,9638) ALPHA,PTG,PTGA,GFE0,GFEA
                        END IF
 1007    AMAX=ALPHA
         PTG2=PTGA
         IF(DABS(GFEA-GFE0).LT.DELMP*DABS(GFE0).AND.PTGA.LT.0.0D0)
     1     GO TO 1008
         IF(DABS(PTGA).LT.ETA) THEN
            IF(GFEA.LT.GFE0) GO TO 1008
            IF(PTGA.LT.DELMP)  THEN
                DELG=(GFEA-GFE0)/DABS(GFE0)
                IF(DABS(ALPHA*PTG).GT.DEL8*DABS(GFE0)) THEN
                       WRITE(6,9980) AMAX,PTGA,GFEA,DELG
                END IF
 9980            FORMAT('PRIMAL 9980  (NON-CONVEXITY)',1P4D12.5)
                GO TO 1008
            END IF
         END IF
         ALPHA=AMIN-(AMAX-AMIN)*PTG/(PTG2-PTG)
         IFUN=IFUN+1
         IF(IFUN.GT.NFUN) THEN
                WRITE(6,9995) IFUN
```

```
            ERROR=.TRUE.
            RETURN
         END IF
         PTGA=0.0D0
         GFEA=0.0D0
         DO 37 K=1,NPHAS
         KP=KPH(K)
         RR=.FALSE.
         IF(REMPHS) THEN
            DO 88 KK=1,NPREM
            IF(KP.EQ.KREM(KK)) RR=.TRUE.
   88       CONTINUE
         END IF
         IF(IABS(KTYP(KP)).GT.1.AND..NOT.RR) THEN
            WRITE(*,*) 'CALLING GRAD FROM PRIMAL @ LN 769'
            CALL GRAD(KP,ALPHA,PTGA,GFEA)
         ELSE IF(KTYP(KP).LT.-1) THEN
            JM=JMIN(KP)
            JX=JMAX(KP)
            DO 84 J=JM,JX
            GFEA=GFEA+(X(J)+ALPHA*P(J))*G(J)
            PTGA=PTGA+P(J)*G(J)
   84       CONTINUE
         ELSE
            J=JMIN(KP)
            GFEA=GFEA+(X(J)+ALPHA*P(J))*G(J)
            PTGA=PTGA+P(J)*G(J)
         END IF
   37    CONTINUE
                        IF(PP) THEN
         WRITE(6,9638) ALPHA,PTG,PTGA,GFE0,GFEA
 9638    FORMAT(' LINESEARCH',1P5D24.16,/)
                        END IF
         IF(PTGA*PTG2.LT.0.0D0) THEN
            PTG=PTG2
            AMIN=AMAX
         ELSE
            PTG=PTG*0.5D0
         END IF
         GO TO 1007
 1008    IF(J0.GT.0) THEN
            IF(KTYP(KSPEC(J0)).GT.1) THEN
               IF(X(J0)+ALPHA*P(J0).GT.DEL2P) J0=0
            ELSE
               IF(X(J0)+ALPHA*P(J0).GT.DELMP) J0=0
            END IF
         END IF
C
C        UPDATE X = X + ALPHA*P
C
 1009    GFE0=GFEA
         ITER=ITER+1
C
C        UPDATE NORMAL STEP
```

```
C
      IF(NORMAL) THEN
          DO 38 JJ=1,NINACT
          J=JI(JJ)
          X(J)=X(J)+ALPHA*P(J)
   38     CONTINUE
                  IF(PP) THEN
              WRITE(6,5001) ITER,IFUN,J0,GFE0,ALPHA
              WRITE(6,5107)
 5001          FORMAT(' PRIMAL NORMAL',3I5,1PD25.17,1P2D20.8)
              DO 6000 J=1,NS
              KK=(KSPEC(J))
              WRITE(6,5002) J,KK,KTYP(KK),MACT(J),X(J),G(J),P(J)
 6000         CONTINUE
 5002         FORMAT(4I5,F25.21,1P3D13.4)
                  END IF
          IF(J0.EQ.0) GO TO 1003
          WRITE(*,*) 'CALLING CONADD @ LN 826'
          CALL CONADD(J0)
          IF(NDC.GT.ND) THEN
              NDP1=ND+1
              DO 85 JD=NDP1,NDC
              P(JA(JD))=0.0D0
              Q(JA(JD),NSP1)=0.0D0
   85         CONTINUE
          END IF
          K0=KSPEC(J0)
          IF(KTYP(K0).EQ.1) THEN
              X(J0)=0.0D0
              P(J0) = 0.0D0
              Q(J0,NSP1)=0.0D0
              KTYP(K0)=-KTYP(K0)
              NPHAS=NPHAS-1
              DO 46 K=1,NPHAS
              IF(KPH(K).EQ.K0) KPH(K)=KPH(NPHAS+1)
   46         CONTINUE
              GO TO 1002
          ELSE
              X(J0)=DEL2
              JM=JMIN(K0)
              JX=JMAX(K0)
              TAU=0.0D0
              PI=0.0D0
              DO 47 J=JM,JX
              TAU=TAU+X(J)
              PI=PI+P(J)
   47         CONTINUE
              P(J0)=0.0D0
              Q(J0,NSP1)=0.0D0
              IF(TAU+(1.0D0-ALPHA)*PI.LT.1.0D-05) THEN
                  KCON=NCC+1
                  NCMIN=KCON
                  KTYP(K0)=-KTYP(K0)
                  TAU=1.0D0/TAU
```

```
                DO 49 J=JM,JX
                W0(J)=X(J)*TAU
                P(J)=-X(J)
                II=MACT(J)
                IF(II.LT.0) THEN
                   WRITE(*,*) 'CALLING CONADD @ LN 868'
                   CALL CONADD(J)
                ELSE IF(II.GT.NC0) THEN
                   IF(II.LT.NCMIN) NCMIN=II
                END IF
 49             CONTINUE
                IF(NDC.GT.ND) THEN
                   NDP1=ND+1
                   DO 99 JD=NDP1,NDC
                   P(JA(JD))=0.0D0
                   Q(JA(JD),NSP1)=0.0D0
 99                CONTINUE
                END IF
                NPREM=1
                KREM(1)=K0
                NJMIN=NCMIN
                AMAX=1.0D0
                NORMAL=.FALSE.
                REMPHS=.TRUE.
                GO TO 1001
             ELSE
                GO TO 1002
             END IF
          END IF
C
C -----------------------------------------------------------------
C      UPDATE MODIFIED STEP
C
       ELSE
          NDG=0
          PREM=.FALSE.
          REMP=.FALSE.
          GRADF=.FALSE.
          IF(REFIN) REMP=.TRUE.
          NPREM=0
          IF(PP) THEN
            DO 1112 J=1,NS
            YS(J)=0.0D0
1112         CONTINUE
          END IF
          DO 39 KP=1,NPHAS
          K0=KPH(KP)
          IF(KTYP(K0).LT.0) KTYP(K0)=-KTYP(K0)
          REMDEG=.TRUE.
          JPH=0
          IF(KTYP(K0).EQ.1) THEN
             J=JMIN(K0)
             X(J)=X(J)+ALPHA*P(J)
                   YS(J)=P(J)
```

```
            P(J)=0.0D0
            Q(J,NSP1)=0.0D0
            IF(X(J).GT.DEL4) THEN
               IF(MACT(J).GT.NC0) THEN
                  WRITE(*,*) 'CALLING CONDEL @ LN 921'
                  CALL CONDEL(J)
                  REMP=.FALSE.
                  REMDEG=.FALSE.
               END IF
            ELSE
               PREM=.TRUE.
               KTYP(K0)=-1
               IF(MACT(J).LT.0) THEN
                  WRITE(*,*) 'CALLING CONADD @ LN 930'
                  CALL CONADD(J)
                  JPH=JPH+1
               ELSE IF(MACT(J).LT.NC0P1) THEN
                  NDG=NDG+1
                  JDG(NDG)=J
               END IF
            END IF
         ELSE
            GRADP=.FALSE.
            JM=JMIN(K0)
            JX=JMAX(K0)
            TAU=0.0D0
            DO 40 J=JM,JX
            X(J)=X(J)+ALPHA*P(J)
            TAU=TAU+X(J)
                 YS(J)=P(J)
            P(J)=0.0D0
            Q(J,NSP1)=0.0D0
            IF(X(J).GT.DEL2P) THEN
               IF(MACT(J).GT.NC0) THEN
                  WRITE(*,*) 'CALLING CONDEL @ LN 951'
                  CALL CONDEL(J)
                  REMP=.FALSE.
                  REMDEG=.FALSE.
               END IF
            ELSE
               IF(X(J).LT.DELLB) THEN
                  X(J)=DELLB
                  GRADP=.TRUE.
               END IF
               IF(MACT(J).LT.0) THEN
                  WRITE(*,*) 'CALLING CONADD @ LN 962'
                  CALL CONADD(J)
                  JPH=JPH+1
               ELSE IF(MACT(J).LT.NC0P1) THEN
                  NDG=NDG+1
                  JDG(NDG)=J
               END IF
            END IF
40          CONTINUE
```

```
            IF(REMPHS.AND.TAU.LT.DEL4*10.0D0) THEN
               DO 44 J=JM,JX
               X(J)=0.0D0
 44            CONTINUE
               KTYP(K0)=-KTYP(K0)
               PREM=.TRUE.
               GRADP=.FALSE.
            ELSE IF(REMP.AND.TAU.LT.100.D0*DEL2.OR.REMDEG
     1            .AND.TAU.LT.100.0D0*DEL2.AND.JPH.GT.0) THEN
               TAU=1.0D0/TAU
               DO 45 J=JM,JX
               W0(J)=X(J)*TAU
 45            CONTINUE
               NPREM=NPREM+1
               KREM(NPREM)=K0
            END IF
            IF(GRADP) GRADF=.TRUE.
         END IF
 39      CONTINUE
         IF(NDG.GT.0) THEN
            DO 41 JD=1,NDG
            J=JDG(JD)
            IF(MACT(J).LT.0) THEN
               WRITE(*,*) 'CALLING CONADD @ LN 994'
               CALL CONADD(J)
            END IF
 41         CONTINUE
         END IF
         IF(J0.GT.0) THEN
            IF(MACT(J0).LT.0) THEN
               WRITE(*,*) 'CALLING CONADD @ LN 1001'
               CALL CONADD(J0)
                  IF(PP) THEN
                  WRITE(6,7878) J0,X(J0)
7878               FORMAT(' MOD STEP 7878',I5,1PD12.4)
                  END IF
            END IF
         END IF
         IF(PREM) THEN
            NP=NPHAS
            NPHAS=0
            DO 42 K=1,NP
            IF(KTYP(KPH(K)).GT.0) THEN
               NPHAS=NPHAS+1
               KPH(NPHAS)=KPH(K)
            END IF
 42         CONTINUE
         END IF
         IF(GRADF) THEN
           IF(PP) WRITE(6,3444)
3444        FORMAT(' SOL. VAR. HIT ZERO')
           GFE0=0.0D0
           PTG=0.0D0
           ALP=0.0D0
```

```
            DO 110  KP=1,NPHAS
            K=KPH(KP)
            IF(KTYP(K).GT.1) THEN
              WRITE(*,*) 'CALLING GRAD FROM PRIMAL @ LN 1028'
              CALL GRAD(K,ALP,PTG,GFE0)
            ELSE
              J=JMIN(K)
              GFE0=GFE0+X(J)*G(J)
            END IF
110         CONTINUE
          END IF
                IF(PP) THEN
            IF(ADDPHS) THEN
              WRITE(6,5003) ITER,IFUN,J0,GFE0,ALPHA
            ELSE IF(REMPHS) THEN
              WRITE(6,5004) ITER,IFUN,J0,GFE0,ALPHA
            ELSE IF(REFIN) THEN
              WRITE(6,5005) ITER,IFUN,J0,GFE0,ALPHA
            ELSE IF(SENS) THEN
              WRITE(6,5106) ITER,IFUN,J0,GFE0,ALPHA
            END IF
5003         FORMAT(' PRIMAL ADDPHS',3I5,1PD25.17,1PD20.8)
5004         FORMAT(' PRIMAL REMPHS',3I5,1PD25.17,1PD20.8)
5005         FORMAT(' PRIMAL REFIN ',3I5,1PD25.17,1PD20.8)
5106         FORMAT(' PRIMAL SENS  ',3I5,1PD25.17,1PD20.8)
5107         FORMAT('   J   K0 KTYP MACT',24X,'X',12X,'G',12X,'P')
            WRITE(6,5107)
            DO 6001 J=1,NS
            KK=(KSPEC(J))
            WRITE(6,5002) J,KK,KTYP(KK),MACT(J),X(J),G(J),YS(J)
6001         CONTINUE
            END IF
          IF(SENS) THEN
            GO TO 1000
          ELSE IF(NPREM.EQ.0) THEN
            ADDPHS=.FALSE.
            REMPHS=.FALSE.
            REFIN=.FALSE.
            NORMAL=.TRUE.
            GO TO 1002
          ELSE
            ADDPHS=.FALSE.
            REMPHS=.TRUE.
            REFIN=.FALSE.
            NORMAL=.FALSE.
            KCON=NCC+1
            NCMIN=NCC+1
            DO 78 KR=1,NPREM
            K0=KREM(KR)
            KTYP(K0)=-KTYP(K0)
            JM=JMIN(K0)
            JX=JMAX(K0)
            DO 82 J=JM,JX
            P(J)=-X(J)
```

```
                II=MACT(J)
                IF(II.GT.NC0.AND.II.LT.NCMIN) NCMIN=II
   82           CONTINUE
   78           CONTINUE
                NJMIN=NCMIN
                AMAX=1.0D0
                GO TO 1001
             END IF
          END IF
C ---------------------------------------------------------------------
C      MODIFIED STEP FAILED
C
C          IERR= -J, P(J) NOT SET CORRECTLY
C          IERR=  1, PTG > 0
C          IERR=  2, AMAX NOT FEASIBLE
C
 1012  IF(REMPHS) THEN
          REMPHS=.FALSE.
          DO 79 K=1,NPREM
          K0=KREM(K)
          KTYP(K0)=-KTYP(K0)
   79     CONTINUE
          IF(KCON.LE.NC) THEN
             DO 51 II=KCON,NC
             I=(KCON+NC)-II
             JR=JA(I)
             WRITE(*,*) ' CALLING CONDEL @ LN 1105'
             CALL CONDEL(JR)
   51        CONTINUE
          END IF
                      IF(PP) THEN
                   WRITE(6,5007) IERR,(KREM(I),I=1,NPREM)
 5007        FORMAT(' PRIMAL SOLUTION PHASE REMOVAL FAILED ',2015)
                      END IF
       ELSE IF(SUBMIN) THEN
          WRITE(6,9998)
 9998      FORMAT(' PRIMAL 9998  (CONSTRAINT DELETION STEP)')
          ERROR=.TRUE.
          RETURN
       ELSE
          NPHAS=NPSAV
          ADDPHS=.FALSE.
                      IF(PP) THEN
                   WRITE(6,9998)
                      END IF
       END IF
       DO 52 J=1,NS
       P(J)=0.0D0
       Q(J,NSP1)=0.0D0
   52  CONTINUE
       NORMAL=.TRUE.
       GO TO 1002
       END
```

```
C-----------------------------------------------------------------
C
C       THIS SUBROUTINE CALCULATES THE CHEMICAL POTENTIALS FOR
THE
C       SOLUTION SPECIES IN PHASE KPP AS WELL AS THE SOLUTION'S
C       CONTRIBUTION TO THE FREE ENERGY, AND ITS CONTRIBUTION
TO THE
C       DERIVATIVE OF THE FREE ENERGY WITH RESPECT TO THE STEP
C       DIRECTION (P).
C
C-----------------------------------------------------------------
      SUBROUTINE GRAD(KPP,STSZ,ZETA,GFE)
C
C       COMPUTE CHEMICAL POTENTIALS FOR AQUEOUS ELECTROLYTE
C       SOLUTIONS USING PITZER AND KIM (1974) MODEL WITH
C       ELECTROSTATIC MIXING TERMS FROM PITZER(1975).
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      DOUBLE PRECISION IONIC,MOLAL
      INTEGER*2 Z,ZMAX
      CHARACTER*24 A1
      COMMON/LABEL/Z(96),U0RT(96),A1(96)
      COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1,NP0C,N00C
      COMMON/SEARCH/X(96),P(96),G(96),W(96),H(64,64),B(96)
      COMMON/AQSOL/JP(46),JMC,JXC,JMA,JXA,JMB,JXB,ALPHA(3)
     1,BETA,B0(18,18),B1(18,18),B2(18,18),CCA(18,18),TA(153)
     2,PSIA(153,18),TC(153),PSIC(153,18),MOLAL(46),F1(3),F2(3),F3(3)
     3,ZSUM
      COMMON/ELCT/APHI,IONIC,ETH(8,8),ETHP(8,8),ETHP2(8,8),ZMAX
      COMMON/PH/LACT
      DATA WH2O/0.0180153D0/
      STEP=STSZ
      IF(IABS(KTYP(KPP)).EQ.2) GO TO 500
C SAS
      IF(IABS(KTYP(KPP)).EQ.3) GO TO 600
C SAS
      JM=JMIN(KPP)
      JX=JMAX(KPP)
      IONIC=0.0D0
      ZSUM=0.0D0
      PHI=0.0D0
      CONVR=1.0D0/(WH2O*(X(JM)+STEP*P(JM)) )
      DO 2 J=JMC,JX
      S=(X(J)+STEP*P(J))*CONVR
      G(J)=U0RT(J)+DLOG(S)
      PHI=PHI+S
      ZSUM=ZSUM+IABS(Z(J))*S
      IONIC=IONIC+S*Z(J)*Z(J)
      MOLAL(J)=S
    2 CONTINUE
      IONIC=IONIC*0.5D0
      S=DSQRT(IONIC)
```

```
      SH=S
      T=1.0D0+BETA*S
      PHI=PHI-2.0D0*APHI*S*IONIC/T
      FZ2=-APHI*(S/T+2.0*DLOG(T)/BETA)
      FZ=0.0D0
      DO 3 I=1,3
      T=ALPHA(I)*S
      IF(T.LT.1.0D-07) GO TO 4
      F1(I)=DEXP(-T)
      F2(I)=2.0D0*(1.0D0-(1.0D0+T)*F1(I))/(T*T)
      F3(I)=(F1(I)-F2(I))/IONIC
      GO TO 3
    4 F1(I)=1.0D0-T*(1.0D0-T*(.5D0-T*(.166666666666667D0
     1 -T*4.166666666666D-02)))
      F2(I)=1.D0-T*(.666666666666667D0-T*(0.25D0-T*(
     1 6.66666666666666D-02-T*(1.388888888888D-02-T*2.380952381D-03))))
      IF(T.LT.1.0D-21)T=1.0D-21
      F3(I)=(-.333333333333333D0/T+.25D0-T*(.1D0
     1 -T*(2.77777777777778D-02-T*5.952380952D-03)))*ALPHA(I)*ALPHA(I)
    3 CONTINUE
      GAMCL=FZ2+IONIC*(.0967D0+.2122D0*(F2(1)+F1(1))-IONIC*.00126D0)
      WRITE(*,*) 'CALLING ELECT @ LN 74'
      CALL ELECT
      DO 13 I=JMC,JXC
      KK=MIN0(Z(I),2)
      DO 12 J=JMA,JXA
      S=MOLAL(I)*MOLAL(J)
      K=MIN0(KK,-Z(J))
      PHI=PHI+2.0D0*S*(B0(JP(I),JP(J))+B1(JP(I),JP(J))*F1(K)
     1    +B2(JP(I),JP(J))*F1(3)+ZSUM*CCA(JP(I),JP(J)) )
      T=2.0D0*(B0(JP(I),JP(J))+B1(JP(I),JP(J))*F2(K)
     1  +B2(JP(I),JP(J))*F2(3) )+ZSUM*CCA(JP(I),JP(J))
      G(I)=G(I)+MOLAL(J)*T
      G(J)=G(J)+MOLAL(I)*T
      FZ2=FZ2+S*(B1(JP(I),JP(J))*F3(K)+B2(JP(I),JP(J))*F3(3) )
      FZ=FZ+S*CCA(JP(I),JP(J))
   12 CONTINUE
   13 CONTINUE
      L=0
      JXAM1=JXA-1
      IF(JMA.GT.JXAM1) GO TO 100
      DO 30 I=JMA,JXAM1
      IP1=I+1
      DO 36 J=IP1,JXA
      L=L+1
      THETA=TA(L)+ETH(-Z(I),-Z(J))
      T=THETA
      S=MOLAL(I)*MOLAL(J)
      FZ2=FZ2+S*ETHP(-Z(I),-Z(J))
      DO 28 K=JMC,JXC
      T=T+MOLAL(K)*PSIA(L,JP(K))
      G(K)=G(K)+S*PSIA(L,JP(K))
   28 CONTINUE
      PHI=PHI+2.0D0*S*(T+IONIC*ETHP(-Z(I),-Z(J)))
```

```
      T=T+THETA
      G(I)=G(I)+MOLAL(J)*T
      G(J)=G(J)+MOLAL(I)*T
   36 CONTINUE
   30 CONTINUE
  100 JXCM1=JXC-1
      IF(JMC.GT.JXCM1) GO TO 102
      L=0
      DO 34 I=JMC,JXCM1
      IP1=I+1
      DO 33 J=IP1,JXC
      L=L+1
      THETA=TC(L)+ETH(Z(I),Z(J))
      T=THETA
      S=MOLAL(I)*MOLAL(J)
      FZ2=FZ2+S*ETHP(Z(I),Z(J))
      DO 32 K=JMA,JXA
      T=T+MOLAL(K)*PSIC(L,JP(K))
      G(K)=G(K)+S*PSIC(L,JP(K))
   32 CONTINUE
      PHI=PHI+2.0D0*S*(T+IONIC*ETHP(Z(I),Z(J)))
      T=T+THETA
      G(I)=G(I)+MOLAL(J)*T
      G(J)=G(J)+MOLAL(I)*T
   33 CONTINUE
   34 CONTINUE
C************************
C************************
  102 G(JM)=U0RT(JM)-WH2O*PHI
      T=G(JM)*P(JM)
      DO 9 I=JMC,JX
      G(I)=G(I)+IABS(Z(I))*(FZ+IABS(Z(I))*FZ2)
   80 T=T+G(I)*P(I)
    9 CONTINUE
      ZETA=ZETA+T
      DO 169 J=JM,JX
      GFE=GFE+(X(J)+STEP*P(J))*G(J)
  169 CONTINUE
      IF(LACT.EQ.0) THEN
         AL1=G(JMA)-DLOG(MOLAL(JMA))-U0RT(JMA)
         PHI=AL1-GAMCL
   67    DO 68 J=JMC,JXA
         G(J)=G(J)+Z(J)*PHI
   68    CONTINUE
      END IF
      RETURN
  500 JM=JMIN(KPP)
      JX=JMAX(KPP)
      S=0.0D0
      DO 70 J=JM,JX
   70 S=S+X(J)+STEP*P(J)
      S=1.0D0/S
      T=0.0D0
      DO 71 J=JM,JX
```

```
      G(J)=U0RT(J)+DLOG(S*(X(J)+STEP*P(J)))
      GFE=GFE+(X(J)+STEP*P(J))*G(J)
   71 T=T+G(J)*P(J)
      ZETA=ZETA+T
      RETURN
C SAS
  600 JM=JMIN(KPP)
      JX=JMAX(KPP)
      IONIC=0.0D0
      CONVR=1.0D0/(WH2O*(X(JM)+STEP*P(JM)))
      DO 72 J=JMC,JX
      S=(X(J)+STEP*P(J))*CONVR
      IONIC=IONIC+S*Z(J)*Z(J)
   72 CONTINUE
      IONIC=0.5D0*IONIC
      S=0.0D0
      DO 73 J=JM,JX
      S=S+X(J)+STEP*P(J)
   73 CONTINUE
      S=1.0D0/S
      T=0.0D0
      DAVIES=(DSQRT(IONIC)/(DSQRT(IONIC)+1.0D0)-0.3D0*IONIC)
      DO 74 J=JM,JX
      G(J)=U0RT(J)+DLOG(S*(X(J)+STEP*P(J)))-3*APHI*Z(J)*Z(J)*DAVIES
      GFE=GFE+(X(J)+STEP*P(J))*G(J)
      T=T+G(J)*P(J)
   74 CONTINUE
      ZETA=ZETA+T
      RETURN
C SAS
      END
```

```
C-----------------------------------------------------------------
C
C     THIS SUBROUTINE CALCULATES THE IONIC STRENGTH AND
C     CHARGE DEPENDENT ELECTROSTATIC TERMS IN THE EXPRESSION
C     FOR THETA AS WELL AS THEIR FIRST AND SECOND DERIVATIVES
C     USING A CHEBYCHEV EXPANSION.
C-----------------------------------------------------------------
      SUBROUTINE ELECT
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      COMMON/ELCT/APHI,ST,ETH(8,8),ETHP(8,8),ETHP2(8,8),NZ
      DIMENSION A1(21),A2(21)
      DATA A1/-.000000000010991D0,-.000000000002563D0
     1,0.000000000001943D0,0.000000000046333D0,-.000000000050847D0
     1,-.000000000821969D0,0.000000001229405D0,0.000000013522610D0
     1,-.000000025267769D0,-.000000202099617D0,0.000000396566462D0
     1,0.000002937706971D0,-.000004537895710D0,-.000045036975204D0
     1,0.000036583601823D0,0.000636874599598D0,0.000388260636404D0
     1,-.007299499690937D0,-.029779077456514D0,-.060076477753119D0
     1,1.925154014814667D0/
      DATA A2/0.000000000237816D0,-.000000002849257D0
     1,-.000000006944757D0,0.000000004558555D0,0.000000080779570D0
     1,0.000000216991779D0,-.000000250453880D0,-.000003548684306D0
     1,-.000004583768938D0,0.000034682122751D0,0.000087294451594D0
     1,-.000242107641309D0,-.000887171310131D0,0.001130378079086D0
     1,0.006519840398744D0,-.001668087945272D0,-.036552745910311D0
     1,-.028796057604906D0,0.150044637187895D0,0.462762985338493D0
     1,0.628023320520852D0/
      NZM1=NZ-1
      IF(NZM1.EQ.0) GO TO 6
      X0=6.0D0*APHI*DSQRT(ST)
      DO 2 II=1,NZ
      DO 1 JJ=II,NZ
      X=II*JJ*X0
      IF(X.GT.1.0D0) GO TO 20
      Y=4.0D0*(X**0.2D0)
      G=0.2D0*Y/X
      G2=-0.8D0*G/X
      Y=Y-2.0D0
      BI=0.0D0
      BIP1=0.0D0
      DBI=0.0D0
      DBIP1=0.0D0
      D2BI=0.0D0
      D2BIP1=0.0D0
      DO 10 I=1,21
      BIP2=BIP1
      BIP1=BI
      BI=Y*BIP1-BIP2+A1(I)
      DBIP2=DBIP1
      DBIP1=DBI
      DBI=BIP1+Y*DBIP1-DBIP2
      D2BIP2=D2BIP1
      D2BIP1=D2BI
```

```
      D2BI=2.0D0*DBIP1+Y*D2BIP1-D2BIP2
10    CONTINUE
      GO TO 40
20    Y=4.444444444444444D0/(X**0.1D0)
      G=-0.1D0*Y/X
      G2=-1.1D0*G/X
      Y=Y-2.44444444444444D0
      BI=0.0D0
      BIP1=0.0D0
      DBI=0.0D0
      DBIP1=0.0D0
      D2BI=0.0D0
      D2BIP1=0.0D0
      DO 25 I=1,21
      BIP2=BIP1
      BIP1=BI
      BI=Y*BIP1-BIP2+A2(I)
      DBIP2=DBIP1
      DBIP1=DBI
      DBI=BIP1+Y*DBIP1-DBIP2
      D2BIP2=D2BIP1
      D2BIP1=D2BI
      D2BI=2.0D0*DBIP1+Y*D2BIP1-D2BIP2
25    CONTINUE
40    ETH(II,JJ)=0.25D0*X+0.5D0*(BI-BIP2)-1.0D0
      THP=0.5D0*(DBI-DBIP2)
      ETHP2(II,JJ)=X*X*(G2*THP+0.5D0*G*G*(D2BI-D2BIP2))
      ETHP(II,JJ)=X*(G*THP+0.25D0)
 1    CONTINUE
 2    CONTINUE
      C=0.5D0/ST
      C1=0.5D0*C
      C2=C*C1
      C3=C*C2
      DO 4 I=1,NZM1
      IP1=I+1
      DO 3 J=IP1,NZ
      TH=I*J*C1*(ETH(I,J)-0.5D0*(ETH(I,I)+ETH(J,J)))
      THP=-TH/ST+I*J*C2*(ETHP(I,J)-0.5D0*(ETHP(I,I)+ETHP(J,J)))
      THP2=C*(-TH/ST-5.0D0*THP)
      THP2=THP2+I*J*C3*(ETHP2(I,J)-0.5D0*(ETHP2(I,I)+ETHP2(J,J)))
      ETH(I,J)=TH
      ETH(J,I)=TH
      ETHP(I,J)=THP
      ETHP(J,I)=THP
      ETHP2(I,J)=THP2
 3    ETHP2(J,I)=THP2
 4    CONTINUE
 6    DO 5 I=1,NZ
      ETH(I,I)=0.0D0
      ETHP(I,I)=0.0D0
 5    ETHP2(I,I)=0.0D0
      RETURN
      END
```

```
C-----------------------------------------------------------------------
C
C       THIS SUBROUTINE CALCULATES THE LOWER TRIANGLE OF THE
C       SECOND DERIVATIVE (SYMMETRIC) MATRIX OF THE FREE
C       ENERGY.
C
C
C-----------------------------------------------------------------------
      SUBROUTINE HESS(KPP)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      DOUBLE PRECISION IONIC,MOLAL
      INTEGER*2 Z,ZMAX,ZI,ZJ
      CHARACTER*24 A1
      COMMON/LABEL/Z(96),U0RT(96),A1(96)
      COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1,NP0C,N00C
      COMMON/SEARCH/X(96),P(96),G(96),W(96),H(64,64),B(96)
      COMMON/AQSOL/JP(46),JMC,JXC,JMA,JXA,JMB,JXB,ALPHA(3)
     1,BETA,B0(18,18),B1(18,18),B2(18,18),CCA(18,18),TA(153)
     2,PSIA(153,18),TC(153),PSIC(153,18),MOLAL(46),F1(3),F2(3),F3(3)
     3,ZSUM
      COMMON/ELCT/APHI,IONIC,ETH(8,8),ETHP(8,8),ETHP2(8,8),ZMAX
      COMMON/PRINT/PP,PP2,PD
      LOGICAL PP,PP2,PD
      DIMENSION GT(64),HT(64)
      DATA WH2O/0.0180153D0/
      IF(IABS(KTYP(KPP)).EQ.2) GO TO 500
      IF(IABS(KTYP(KPP)).EQ.3) GO TO 700
      JM=JMIN(KPP)
      JX=JMAX(KPP)
      DO 2 J=JM,JX
      GT(J)=0.0D0
      HT(J)=0.0D0
      DO 1 I=J,JX
      H(I,J)=0.0D0
    1 CONTINUE
    2 CONTINUE
      S=DSQRT(IONIC)
      T=1.0D0+BETA*S
      FZ2=-APHI*(0.5D0+T)/(S*T*T)
      DO 3 I=1,3
      F1(I)=-(2.0D0*F3(I)+0.5D0*ALPHA(I)*F1(I)/S)/IONIC
    3 CONTINUE
      DO 13 I=JMC,JXC
      KK=MIN0(Z(I),2)
      DO 12 J=JMA,JXA
      S=MOLAL(I)*MOLAL(J)
      K=MIN0(KK,-Z(J))
      H(J,I)=2.0D0*(B0(JP(I),JP(J))+B1(JP(I),JP(J))*F2(K)
     1  +B2(JP(I),JP(J))*F2(3) )+ZSUM*CCA(JP(I),JP(J))
      T=B1(JP(I),JP(J))*F3(K)+B2(JP(I),JP(J))*F3(3)
      GT(I)=GT(I)+MOLAL(J)*T
      GT(J)=GT(J)+MOLAL(I)*T
```

```
      FZ2=FZ2+MOLAL(I)*MOLAL(J)*(B1(JP(I),JP(J))*F1(K)
     1       +B2(JP(I),JP(J))*F1(3))
      HT(I)=HT(I)+MOLAL(J)*CCA(JP(I),JP(J))
      HT(J)=HT(J)+MOLAL(I)*CCA(JP(I),JP(J))
   12 CONTINUE
   13 CONTINUE
      L=0
      JXAM1=JXA-1
      IF(JMA.GT.JXAM1) GO TO 100
      DO 30 I=JMA,JXAM1
      IP1=I+1
      DO 36 J=IP1,JXA
      L=L+1
      H(J,I)=H(J,I)+2.0D0*(TA(L)+ETH(-Z(I),-Z(J)))
      DO 28 K=JMC,JXC
      H(J,I)=H(J,I)+MOLAL(K)*PSIA(L,JP(K))
      H(I,K)=H(I,K)+MOLAL(J)*PSIA(L,JP(K))
      H(J,K)=H(J,K)+MOLAL(I)*PSIA(L,JP(K))
   28 CONTINUE
      GT(I)=GT(I)+MOLAL(J)*ETHP(-Z(I),-Z(J))
      GT(J)=GT(J)+MOLAL(I)*ETHP(-Z(I),-Z(J))
      FZ2=FZ2+MOLAL(I)*MOLAL(J)*ETHP2(-Z(I),-Z(J))
   36 CONTINUE
   30 CONTINUE
  100 JXCM1=JXC-1
      IF(JMC.GT.JXCM1) GO TO 101
      L=0
      DO 34 I=JMC,JXCM1
      IP1=I+1
      DO 33 J=IP1,JXC
      L=L+1
      H(J,I)=H(J,I)+2.0D0*(TC(L)+ETH(Z(I),Z(J)))
      DO 32 K=JMA,JXA
      H(J,I)=H(J,I)+MOLAL(K)*PSIC(L,JP(K))
      H(K,I)=H(K,I)+MOLAL(J)*PSIC(L,JP(K))
      H(K,J)=H(K,J)+MOLAL(I)*PSIC(L,JP(K))
   32 CONTINUE
      GT(I)=GT(I)+MOLAL(J)*ETHP(Z(I),Z(J))
      GT(J)=GT(J)+MOLAL(I)*ETHP(Z(I),Z(J))
      FZ2=FZ2+MOLAL(I)*MOLAL(J)*ETHP2(Z(I),Z(J))
   33 CONTINUE
   34 CONTINUE
  101 CONTINUE

  102 CONVR=1.0D0/(WH2O*X(JM))
      FZ2=0.5D0*FZ2
      DO 91 I=JMC,JX
      ZI=IABS(Z(I))
      H(I,I)=CONVR*(H(I,I)+2.0D0*ZI*(HT(I)+ZI*GT(I))+1.0D0/MOLAL(I)
     1       +ZI*ZI*ZI*ZI*FZ2)
      H(I,JM)=H(I,JM)-MOLAL(I)*H(I,I)
      IM1=I-1
      IF(JMC.GT.IM1) GO TO 91
      DO 90 J=JMC,IM1
```

```
      ZJ=IABS(Z(J))
      H(I,J)=CONVR*( H(I,J)+ZI*(HT(J)+ZI*GT(J))+ZJ*(HT(I)+ZJ*GT(I))
     1        +ZI*ZI*ZJ*ZJ*FZ2)
      H(J,JM)=H(J,JM)-MOLAL(I)*H(I,J)
      H(I,JM)=H(I,JM)-MOLAL(J)*H(I,J)
   90 CONTINUE
   91 CONTINUE

  600 T=0.0D0
      DO 92 J=JMC,JX
      H(J,JM)=WH2O*H(J,JM)
   92 T=T-MOLAL(J)*H(J,JM)
      H(JM,JM)=T*WH2O
      RETURN

C  IDEAL SOLUTION HESSIAN ELEMENTS
  500 JM=JMIN(KPP)
      JX=JMAX(KPP)
      S=0.0D0
      DO 70 J=JM,JX
   70 S=S+X(J)
      S=1.0D0/S
      DO 71 J=JM,JX
      DO 72 I=J,JX
      H(I,J)=-S
   72 CONTINUE
      H(J,J)=1.0D0/X(J)-S
   71 CONTINUE
      RETURN
C   SAS
C  DAVIES SOLUTION HESSIAN ELEMENTS
  700 JM=JMIN(KPP)
      JX=JMAX(KPP)
      S=0.0D0
      DO 80 J=JM,JX
      S=S+X(J)
   80 CONTINUE
      S=1.0D0/S
      IONIC=0.0D0
      CONVR=1.0D0/(WH2O*X(JM))
      DO 81 J=JMC,JX
      IONIC=IONIC+CONVR*X(J)*Z(J)*Z(J)
   81 CONTINUE
      IONIC=IONIC*0.5D0
      DAVIES=1.0D0/(DSQRT(IONIC)+2.0D0*IONIC+IONIC**1.5D0)-0.6D0
      DO 82 J=JM,JX
      DO 83 I=J,JX
      H(I,J)=-0.75D0*APHI*Z(I)*Z(I)*Z(J)*Z(J)*DAVIES-S
   83 CONTINUE
      H(J,J)=H(J,J)+1.0D0/X(J)
   82 CONTINUE
      RETURN
C SAS
      END
```

```
C-----------------------------------------------------------------
C
C       THIS ROUTINE CALCULATES A REFINEMENT STEP FOR SOLUTION
C       SPECIES WHOSE MOLE NUMBERS ARE AT OR BELOW THE
C       BOUNDARY VALUE. FOR DEGENERATE SPECIES A NEWTON
C       ITERATION PROCEDURE IS USED TO SOLVE FOR THE REFINEMENT
C       DIRECTION.
C
C
C-----------------------------------------------------------------
        SUBROUTINE REFINE(Q2MAX,NCM)
C
C
C       SUBROUTINE ESTIMATES EQUILIBRIUM CONCENTRATIONS FOR
C       SMALL CONCENTRATION SOLUTION PHASE VARIABLES AND
C       COMPUTES A DESCENT DIRECTION
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        IMPLICIT INTEGER*2 (I-N)
        LOGICAL NEWCON,NOVAR
        COMMON/CONST/Q(96,96),MACT(96),JA(96),JI(96),NSC,NCC,NDC,NC0C
        COMMON/DUAL2/DEG(96,96)
        COMMON/DUALB/QC(96,96),IR(96)
        COMMON/SEARCH/X(96),P(96),G(96),W0(96),H(64,64),B(96)
        COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1  ,NP0C,N00C
        COMMON/PRINT/PP,PP2,PD
        LOGICAL PP,PP2,PD
        DIMENSION W1(96),YS(96),V(20),DD(20),JDX(96)
        DIMENSION KBCON(96)
        PARAMETER(DELMP=0.5D0**55)
        PARAMETER(DEL2=1.0D-12)
        PARAMETER(DEL2P=DEL2*(1.0D0+DEL2),DEL2M=DEL2*(1.0D0-DEL2))
        PARAMETER(DEL4=1.0D-13)
        PARAMETER(DEL1=1.0D-07, DELLB=1.0D-20)
        NC=NCC
        NC0=NC0C
        Q2MAX=0.0D0
        IF(NC.EQ.NC0) RETURN
        NS=NSC
        ND=NDC
        NINACT=NS-NC-ND+NC0
        NC0P1=NC0+1
        NOVAR=.TRUE.
C
C       IDENTIFY CONSTRAINED SOLUTION SPECIES (KBCON=-1)
C
              DO 962 J=1,96
              DO 962 I=1,96
  962          DEG(I,J)=0.0D0
        DO 1 I=NC0P1,NC
        IF(PP) THEN
           WRITE(6,8324) I,JA(I),Q(JA(I),I)
 8324      FORMAT(2I5,1PD20.4)
```

```
      END IF
      IF(KTYP(KSPEC(JA(I))).GT.1) THEN
         KBCON(I)=-1
         NOVAR=.FALSE.
      ELSE
         KBCON(I)=0
      END IF
    1 CONTINUE
      IF(NOVAR) RETURN
      NU=0
      NSS=0
C
C     IDENTIFY CONSTRAINTS ON SMALL SOLUTION SPECIES DIRECTION
C     VECTOR
      IF(ND.GT.0) THEN
               NUX=0
         DO 8 JJ=1,ND
         J1=JA(JJ)
         KBCON(JJ)=0
         IF(X(J1).LT.DEL2*100.0D0) THEN
               NUX=NUX+1
            IF(NSS.GT.0) THEN
               DO 2 K=1,NSS
               V(K)=0.0D0
    2          CONTINUE
            END IF
            IF(KTYP(KSPEC(J1)).GT.1) THEN
               NM1=NSS
               NSS=NSS+1
               JDX(NSS)=J1
               W1(NSS)=0.0D0
               V(NSS)=-1.0D0
               DEG(NSS,NUX)=-1.0D0
               KBCON(JJ)=NSS
               IF(NM1.GT.0) THEN
                  DO 3 K=1,NM1
                  QC(NSS,K)=0.0D0
                  QC(K,NSS)=0.0D0
    3             CONTINUE
               END IF
               QC(NSS,NSS)=1.0D0
            END IF
            DO 7 II=NC0P1,NC
            I=(NC0P1+NC)-II
            TAU=Q(J1,I)
            IF(I.LT.NC) THEN
               IP1=I+1
               DO 4 I2=IP1,NC
               TAU=TAU-YS(I2)*Q(JA(I2),I)
    4          CONTINUE
            END IF
            YS(I)=TAU/Q(JA(I),I)
            IF(DABS(YS(I)).GT.DEL4) THEN
               KK=KBCON(I)
```

```
                IF(KK.GT.0) THEN
                    V(KK)=YS(I)
                        DEG(KK,NUX)=YS(I)
                ELSE IF(KK.EQ.-1) THEN
                    NM1=NSS
                    NSS=NSS+1
                    JDX(NSS)=JA(I)
                    V(NSS)=YS(I)
                         DEG(NSS,NUX)=YS(I)
                    KBCON(I)=NSS
                    IF(NM1.GT.0) THEN
                        DO 5 K=1,NM1
                        QC(NSS,K)=0.0D0
                        QC(K,NSS)=0.0D0
 5                      CONTINUE
                    END IF
                    QC(NSS,NSS)=1.0D0
                END IF
            END IF
 7          CONTINUE
C
            IF(NU.LT.NSS) THEN
                NEWCON=.FALSE.
                SIG=0.0D0
                NUP1=NU+1
                DO 84 I=NUP1,NSS
                TAU=0.0D0
                DO 85 J=1,NSS
                TAU=TAU+V(J)*QC(J,I)
 85             CONTINUE
                IF(DABS(TAU).GT.DEL4) NEWCON=.TRUE.
                SIG=SIG+TAU*TAU
                YS(I)=TAU
 84             CONTINUE
                IF(NEWCON) THEN
                    NU=NUP1
                    IF(NU.LT.NSS) THEN
                        SIG=DSIGN(DSQRT(SIG),YS(NU))
                        YS(NU)=YS(NU)+SIG
                        PI=1.0D0/(YS(NU)*SIG)
                        DO 81 J=1,NSS
                        TAU=0.0D0
                        DO 82 I=NU,NSS
                        TAU=TAU+QC(J,I)*YS(I)
 82                     CONTINUE
                        TAU=TAU*PI
                        DO 83 I=NU,NSS
                        QC(J,I)=QC(J,I)-YS(I)*TAU
 83                     CONTINUE
 81                     CONTINUE
                    END IF
                END IF
            END IF
C
```

```
          END IF
    8     CONTINUE
        END IF
          IF(PP) THEN
           WRITE(6,8567) NU
           IF(NU.GT.0) THEN
                DO 901 JS=1,NSS
                WRITE(6,777) JDX(JS),(DEG(JS,I),I=1,NUX)
  777            FORMAT(I5,7F10.6,/,5X,7F10.6,/,5X,7F10.6)
  901            CONTINUE
 8567           FORMAT(' CONSTRAINT EQUATIONS  NU=',I5)
 8568           FORMAT(7F10.6)
           END IF
          END IF
C
C       ESTIMATE EQUILIBRIUM MOLE NUMBER USING MULTIPLIER AND
C       ASSUMING TRACE CONCENTRATION
C
        DO 12 II=NC0P1,NC
        I=(NC0P1+NC)-II
        J1=JA(I)
        TAU=0.0D0
        IF(NINACT.GT.0) THEN
          DO 9 J=1,NINACT
          TAU=TAU+G(JI(J))*Q(JI(J),I)
    9     CONTINUE
        END IF
        IF(ND.GT.0) THEN
          DO 10 J=1,ND
          TAU=TAU+G(JA(J))*Q(JA(J),I)
   10     CONTINUE
        END IF
        IF(I.LT.NC) THEN
          IP1=I+1
          DO 11 J=IP1,NC
          TAU=TAU+YS(JA(J))*Q(JA(J),I)
   11     CONTINUE
        END IF
        YS(J1)=-TAU/Q(J1,I)
        IF(KBCON(I).EQ.-1) THEN
          PI=DMAX1(X(J1)*DEXP(YS(J1)-G(J1)), DELLB)
          P(J1)=PI-X(J1)
          IF(DABS(P(J1)).GT.Q2MAX*X(J1)) Q2MAX=DABS(P(J1))/X(J1)
          NCM=I
              TAU3=G(J1)-YS(J1)
              TAU2=X(J1)+P(J1)
           IF(PP) THEN
              WRITE(6,3232) J1,TAU3,TAU2,P(J1),X(J1)
           END IF
 3232          FORMAT(I4,1P4D16.8)
        ELSE IF(KBCON(I).GT.0) THEN
          W1(KBCON(I))=YS(J1)-G(J1)
           IF(PP) THEN
              WRITE(6,3232) J1,-W1(KBCON(I))
```

```
            END IF
           NCM=I
         END IF
      12 CONTINUE
         IF(PP) THEN
         IF(NU.EQ.0)  WRITE(6,3845) Q2MAX
 3845       FORMAT('  Q2MAX =',1PD12.5)
         END IF
         IF(NU.EQ.0) RETURN
C
C ---------------------------------------------------------------------
C        NEWTON ITERATION FOR A DEGENERATE OPTIMAL SOLUTION
C
C        INITIALIZATION  (LEAST SQUARE ESTIMATE FOR V(I) I=1,NU)
C
         FUNC0=1.0D+30
         NUP1=NU+1
         NUP2=NU+2
         DO 16 I=1,NU
         IP1=I+1
         DO 14 K=1,NU
         TAU=0.0D0
         DO 13 J=1,NSS
         TAU=TAU+X(JDX(J))*QC(J,I)*QC(J,K)
      13 CONTINUE
         H(K,IP1)=TAU
      14 CONTINUE
         TAU=0.0D0
         PI=0.0D0
         DO 15 J=1,NSS
         SIG=X(JDX(J))*QC(J,I)
         TAU=TAU+SIG
         PI=PI+SIG*W1(J)
      15 CONTINUE
         V(I)=0.0D0
         DD(I)=TAU
         H(I,NUP2)=-PI
      16 CONTINUE
         ITER=0
         IFUN=0
C
C        SOLVE LINEAR EQUATIONS FOR NEWTON DIRECTION (OR
C         INITIALIZATION)
 1002    GAMMA=0.0D0
         ZETA=0.0D0
         THETA=0.0D0
         DO 27 I=1,NU
         IP1=I+1
         IF(DABS(H(I,IP1)).GT.GAMMA) GAMMA=DABS(H(I,IP1))
         IF(I.LT.NU) THEN
            DO 26 J=IP1,NU
            IF(DABS(H(I,J+1)).GT.ZETA) ZETA=DABS(H(I,J+1))
      26    CONTINUE
            IF(I.EQ.1) THETA=ZETA
```

```
      END IF
   27 CONTINUE
      DELTA=(0.5D0**45)*DMAX1(GAMMA,ZETA,DELLB)
      BETA2=1.0D0/DMAX1(GAMMA,DELTA,ZETA/NU)
      TAU=DMAX1(DABS(H(1,2)), DELTA, THETA*THETA*BETA2)
      H(1,2)=1.0D0/TAU
      IF(NU.EQ.1) THEN
         TAU=H(1,3)
         H(1,3)=H(1,3)*H(1,2)
      ELSE
         DO 31 J=2,NU
         JM1=J-1
         JP1=J+1
         JP2=J+2
         PHI=H(J,JP1)
         DO 28 K=1,JM1
         PI=H(K,JP1)*H(K,K+1)
         PHI=PHI-PI*H(K,JP1)
         H(K,JP1)=PI
   28    CONTINUE
         THETA=0.0D0
         DO 30 I=JP2,NUP2
         TAU=H(J,I)
         DO 29 K=1,JM1
         TAU=TAU-H(K,JP1)*H(K,I)
   29    CONTINUE
         H(J,I)=TAU
         IF(I.LT.NUP2) THEN
            IF(DABS(TAU).GT.THETA) THETA=DABS(TAU)
         END IF
   30    CONTINUE
         SIG=DMAX1( DABS(PHI), DELTA, THETA*THETA*BETA2)
         H(J,JP1)=1.0D0/SIG
   31    CONTINUE
         H(NU,NUP2)=H(NU,NUP2)*H(NU,NUP1)
         DO 33 JMN=2,NU
         J=NUP1-JMN
         JP1=J+1
         TAU=H(J,NUP2)*H(J,JP1)
         DO 32 K=JP1,NU
         TAU=TAU-H(J,K+1)*H(K,NUP2)
   32    CONTINUE
         H(J,NUP2)=TAU
   33    CONTINUE
      END IF
C
C     COMPUTE MOLE NUMBER ESTIMATES, SHORTEN STEP LENGTH TO
C     INSURE DECREASE OF OBJECTIVE FUNCTION
C
      ALPHA=1.0D0
 1001 FUNC=0.0D0
      DO 50 I=1,NSS
      J=JDX(I)
      TAU0=W1(I)
```

```
      PI=0.0D0
      DO 51 K=1,NU
      TAU0=TAU0+QC(I,K)*V(K)
      PI=PI+QC(I,K)*H(K,NUP2)
   51 CONTINUE
      TAU=TAU0+ALPHA*PI
      IF(TAU.GT.60.0D0) THEN
         ALPHA=(59.9D0-TAU0)/PI
         GO TO 1001
      END IF
      YS(I)=DMAX1( X(J)*DEXP(TAU), DELLB)
      FUNC=FUNC+YS(I)
      P(J)=YS(I)-X(J)
   50 CONTINUE
      DO 52 I=1,NU
      FUNC=FUNC-DD(I)*(V(I)+ALPHA*H(I,NUP2))
   52 CONTINUE
      IFUN=IFUN+1
      IF(FUNC.GE.FUNC0) THEN
         IF(ITER.EQ.0) GO TO 1000
         PTG=0.0D0
         DO 53 J=1,NSS
         TAU=0.0D0
         DO 54 I=1,NU
         TAU=TAU+QC(J,I)*H(I,NUP2)
   54    CONTINUE
         PTG=PTG+P(JDX(J))*TAU
   53    CONTINUE
         IF(PTG.LE.0.0D0) GO TO 1000
         ALPHA=ALPHA*0.5D0
         GO TO 1001
      END IF
C
C     TEST FOR FEASIBILITY (ZERO GRADIENT) AND COMPUTE HESSIAN
C     MATRIX FOR NEWTON ITERATION
C
 1000 FUNC0=FUNC
      ITER=ITER+1
      SIG=0.0D0
      DO 64 K=1,NU
      V(K)=V(K)+ALPHA*H(K,NUP2)
      DO 65 I=K,NU
      TAU=0.0D0
      DO 66 J=1,NSS
      TAU=TAU+YS(J)*QC(J,K)*QC(J,I)
   66 CONTINUE
      H(K,I+1)=TAU
   65 CONTINUE
      TAU=0.0D0
      DO 67 J=1,NSS
      TAU=TAU+P(JDX(J))*QC(J,K)
   67 CONTINUE
      H(K,NUP2)=-TAU
      IF(DABS(TAU).GT.SIG) SIG=DABS(TAU)
```

```
   64   CONTINUE
        IF(SIG.GT.DELMP) GO TO 1002
C
C       CONVERGED WITHIN DELMP OF FEASIBILITY
C       COMPUTE REMAINING PART OF Q2MAX
C
        DO 68 I=1,NSS
        J=JDX(I)
        IF(DABS(P(J)).GT.Q2MAX*X(J)) Q2MAX=DABS(P(J))/X(J)
            IF(PP) THEN
                TAU=X(J)+P(J)
                WRITE(6,3234) J,-W1(I),TAU,P(J),X(J)
 3234            FORMAT(I4,1P4D16.8)
            END IF
   68   CONTINUE
        IF(PP) WRITE(6,3845) Q2MAX
        RETURN
        END
```

```
C-----------------------------------------------------------------------
C
C    THIS SUBTOUTINE ADDS A SPECIES CONSTRAINT (SPECIES J0) TO
C    THE CONSTRAINT MATRIX. IF THE NEW CONSTRAINT IS
C    DEGENERATE THE CONSTRAINT SET IS NOT ALTERED, BUT A NEW
C    ELEMENT TO JA IS ADDED TO NOTE THE DEGENERATE
C    CONSTRAINT.
C
C-----------------------------------------------------------------------
      SUBROUTINE CONADD(J0)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      LOGICAL REDUN,NODEG
      COMMON/CONST/Q(96,96),MACT(96),JA(96),JI(96),NS,NC,ND,NC0
      DIMENSION V(96)
      PARAMETER(DEL4=1.0D-13)
         IF(NS.EQ.NC) GO TO 99
      NC=NC+1
      SIG=0.0D0
      NODEG=.TRUE.
      DO 1 I=NC,NS
      V(I)=Q(J0,I)
      IF(DABS(V(I)).GT.DEL4) NODEG=.FALSE.
      Q(J0,I)=0.0D0
      SIG=SIG+V(I)*V(I)
1     CONTINUE
         IF(NODEG) GO TO 99
      SIG=DSIGN(DSQRT(SIG),V(NC))
      NINACT=(NS+NC0)-(NC+ND)
      JI(-MACT(J0))=JI(NINACT+1)
      MACT(JI(NINACT+1))=MACT(J0)
      JA(NC)=J0
      MACT(J0)=NC
      IF(NS.EQ.NC) THEN
         Q(J0,NC)=V(NC)
         IF(NINACT.GT.0) THEN
           DO 2 J=1,NINACT
           ND=ND+1
           JA(ND)=JI(J)
           MACT(JI(J))=ND
2          CONTINUE
         END IF
      ELSE
         Q(J0,NC)=-SIG
         IF(NINACT.GT.0) THEN
            NDS=ND
            V(NC)=V(NC)+SIG
            PI=1.0D0/(V(NC)*SIG)
            DO 5 MI=1,NINACT
            J=JI(MI)
            TAU=0.0D0
            DO 3 I=NC,NS
            TAU=TAU+V(I)*Q(J,I)
3           CONTINUE
```

```
                TAU=TAU*PI
                REDUN=.TRUE.
                DO 4 I=NC,NS
                T=Q(J,I)-V(I)*TAU
                IF(DABS(T).GT.DEL4) THEN
                   Q(J,I)=T
                   IF(I.GT.NC) REDUN=.FALSE.
                ELSE
                   Q(J,I)=0.0D0
                END IF
    4           CONTINUE
                IF(REDUN) THEN
                   ND=ND+1
                   JA(ND)=J
                END IF
    5           CONTINUE
                IF(ND.GT.NDS) THEN
                   NDS=NDS+1
                   DO 6 JD=NDS,ND
                   J=JA(JD)
                   IF(NINACT.GT.0) THEN
                      JI(-MACT(J))=JI(NINACT)
                      MACT(JI(NINACT))=MACT(J)
                      NINACT=NINACT-1
                   END IF
                   MACT(J)=JD
    6              CONTINUE
                END IF
             END IF
          END IF
          RETURN
C
C -------------------------------------------------------------------
C       ATTEMPT TO ADD A CONSTRAINED VARIABLE
C
   99   WRITE(6,9999) J0
 9999   FORMAT(' CONADD ERROR',I5)
        STOP
        END
```

```
C-----------------------------------------------------------------------
C
C   THIS SUBROUTINE DELETES A SPECIES CONSTRAINT (SPECIES JR).
C
C-----------------------------------------------------------------------
      SUBROUTINE CONDEL(JR)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      COMMON/CONST/Q(96,96),MACT(96),JA(96),JI(96),NS,NC,ND,NC0
      PARAMETER(DEL4=1.0D-13)
      MREM=MACT(JR)
           IF(MREM.LE.NC0) GO TO 99
      NC=NC-1
      NINACT=(NS+NC0)-(NC+ND)
      JI(NINACT)=JR
      MACT(JR)=-NINACT
      IF(MREM.LE.NC) THEN
        DO 4 MM=MREM,NC
        MP1=MM+1
        J0=JA(MP1)
        JA(MM)=J0
        MACT(J0)=MM
        CMM=Q(J0,MM)
        CMP1=Q(J0,MP1)
        SIG=DSIGN(DSQRT(CMM*CMM+CMP1*CMP1),CMM)
        CMM=CMM+SIG
        Q(J0,MM)=-SIG
        Q(J0,MP1)=0.0D0
        V1=1.0D0/SIG
        V2=CMP1/(SIG*CMM)
        IF(NINACT.GT.0) THEN
          DO 1 I=1,NINACT
          J=JI(I)
          TAU=Q(J,MM)*V1+Q(J,MP1)*V2
          Q(J,MM)=Q(J,MM)-CMM*TAU
          Q(J,MP1)=Q(J,MP1)-CMP1*TAU
1         CONTINUE
        END IF
        IF(ND.GT.0) THEN
          DO 2 I=1,ND
          J=JA(I)
          TAU=Q(J,MM)*V1+Q(J,MP1)*V2
          Q(J,MM)=Q(J,MM)-CMM*TAU
          Q(J,MP1)=Q(J,MP1)-CMP1*TAU
2         CONTINUE
        END IF
        IF(MM.LT.NC) THEN
          DO 3 I=MP1,NC
          J=JA(I+1)
          TAU=Q(J,MM)*V1+Q(J,MP1)*V2
          Q(J,MM)=Q(J,MM)-CMM*TAU
          Q(J,MP1)=Q(J,MP1)-CMP1*TAU
3         CONTINUE
        END IF
```

```
    4     CONTINUE
        END IF
        IF(ND.GT.0) THEN
          NDS=ND
          NCP1=NC+1
          ND=0
          DO 5 JD=1,NDS
          JX=JA(JD)
          JA(JD)=0
          T=Q(JX,NCP1)
          IF(DABS(Q(JX,NCP1)).LT.DEL4) THEN
            ND=ND+1
            JA(ND)=JX
            MACT(JX)=ND
            Q(JX,NCP1)=0.0D0
          ELSE
            NINACT=NINACT+1
            JI(NINACT)=JX
            MACT(JX)=-NINACT
          END IF
    5     CONTINUE
        END IF
        RETURN
C
C ---------------------------------------------------------------
C       ATTEMPT TO DELETE A NON-EXPLICITLY CONSTRAINED SPECIES
C
   99   WRITE(6,9999) JR
 9999   FORMAT(' CONDEL ERROR',I5)
        STOP
        END
```

```
C-----------------------------------------------------------------------
C
C     THIS SUBROUTINE SOLVES THE SUBMINIZATION PROBLEM FOR
C     SOLUTION PHASES THAT ARE NOT PRESENT IN ORDER TO
C     CALCULATE THE PHASE MULTIPLIER.  IT ALSO CALCULATES THE
C     PHASE MULTIPLIER FOR PURE PHASES. IT THEN PICKS THE
C     LOWEST NEGATIVE MULTIPLIER.
C
C-----------------------------------------------------------------------
C
      SUBROUTINE DUAL(SUBM,ADDPHS,ZOUTEN,ERROR,WMIN,NPP,NCM,KPH)
C
C
C     SUBROUTINE DETERMINES MINIMUM DESCENT DIRECTION WHEN
C     CONSTRAINTS ARE TO BE RELAXED
C
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        IMPLICIT INTEGER*2 (I-N)
        INTEGER*2 Z
        CHARACTER*24 A1
        LOGICAL ZDUAL(96),ZOUTEN(96),VTYP(96),ERROR,NODEG,SVAR
        LOGICAL TEST,REDUN
        LOGICAL NORMAL,ADDCON,SENS,SUBMIN, SUBM,POSDEF,ADDPHS,
     1,NOMIN,CONCAV
        COMMON/CONST/Q(96,96),MACT(96),JA(96),JI(96),NSC,NCC,NDC,NC0C
        COMMON/DUALB/QB(96,96),MACB(96),JAQ(96),JIQ(96),NSBC,NCBC,NDBC
     1,NCB0C
        COMMON/LABEL/Z(96),U0RT(96),A1(96)
        COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1,NP0C,N00C
        COMMON/SEARCH/X(96),P(96),G(96),PS(96),H(64,64),B(96),S(32)
        COMMON/PRINT/PP,PP2,PD
        COMMON/MMM/MUL,WWW(96)
        LOGICAL PP,PP2,PD,MUL
        DIMENSION JQX(96),JXQ(96),JBMIN(96),KPH(96),KBLK(96),KBCON(96)
        DIMENSION YS(96),V(96),BB(96),GSAV(96),XSAV(96),W1(96)
        DIMENSION SL(96),SLP(96)
        PARAMETER(DEL1=1.0D-07,DEL2=1.0D-12,DEL3=1.0D-12,DEL4=1.0D-13)
        PARAMETER(DEL5=1.0D-01,DEL6=1.0D-04)
        PARAMETER(DEL3P=DEL3*(1.0D0+DEL4),DEL3M=DEL3*(1.0D0-DEL4))
        PARAMETER(DEL2P=DEL2*(1.0D0+DEL2))
        PARAMETER(DELLB=1.0D-20)
        PARAMETER(NFUN=2000)
C
C       AT A SUBSPACE MINIMUM RESET ALL ZOUTENDIJK FLAGS
C
        NS=NSC
        NC=NCC
        NC0=NC0C
        IF(SUBM) THEN
            DO 1 J=1,NS
            ZOUTEN(J)=.FALSE.
    1       CONTINUE
        END IF
```

```
      IF(NC.EQ.NC0) RETURN
      NSP1=NS+1
      ND=NDC
      NC0P1=NC0+1
      NINACT=NS-NC-ND+NC0
C
C     IDENTIFY BLOCKS OF ACTIVE CONSTRAINTS
C
C        KBCON(I) =    0, SINGLE SPECIES BLOCK
C                     K0, SPECIES IN NON-DEG. ZERO SOLUTION PHASE K0
C                    -JJ, DEG. BLOCK ASSOCIATED WITH JA(JJ)
C                   NSP1, CANNOT BE DELETED  (TOO LARGE TO BE
C                         CONSIDERED DEGENERATE OR ZOUTEN=TRUE)
C
      DO 2 I=1,NC
      IF(I.GT.NC0) THEN
         J=JA(I)
         IF(ZOUTEN(J)) THEN
            KBCON(I)=NSP1
         ELSE IF(KTYP(KSPEC(J)).LT.-1) THEN
            KBCON(I)=KSPEC(J)
         ELSE
            KBCON(I)=0
         END IF
      ELSE
            KBCON(I)=NSP1
      END IF
    2 CONTINUE
      IF(ND.GT.0) THEN
         DO 7 JJ=1,ND
         J1=JA(JJ)
         IF(X(J1).LT.DEL2P) THEN
            W1(J1)=G(J1)
            KBCON(JJ)=-JJ
            NOMIN=.FALSE.
            IF(ZOUTEN(JJ)) NOMIN=.TRUE.
            DO 5 II=NC0P1,NC
            I=(NC0P1+NC)-II
            TAU=Q(J1,I)
            IF(I.LT.NC) THEN
               IP1=I+1
               DO 3 I2=IP1,NC
               TAU=TAU-YS(I2)*Q(JA(I2),I)
    3          CONTINUE
            END IF
            YS(I)=TAU/Q(JA(I),I)
            IF(DABS(YS(I)).GT.DEL4) THEN
               NBLK=KBCON(I)
               IF(NBLK.EQ.NSP1) NOMIN=.TRUE.
               IF(NBLK.EQ.0) THEN
                  KBCON(I)=-JJ
               ELSE IF(NBLK.NE.-JJ) THEN
                  DO 4 K=1,NC
                  IF(KBCON(K).EQ.NBLK) KBCON(K)=-JJ
```

```
    4               CONTINUE
                  END IF
                END IF
    5           CONTINUE
                IF(NOMIN) THEN
                  DO 6 K=1,NC
                  IF(KBCON(K).EQ.-JJ) KBCON(K)=NSP1
    6             CONTINUE
                END IF
              END IF
    7         CONTINUE
            END IF
C
C     LEAST SQUARE MULTIPLIER CALCULATION (GRADIENT ASSUMED
C     FOR DEGENERATE SPECIES)
C
            NOMIN=.TRUE.
            DO 11 II=NC0P1,NC
            I=(NC0P1+NC)-II
            J1=JA(I)
            TAU=0.0D0
            IF(NINACT.GT.0) THEN
              DO 8 J=1,NINACT
              TAU=TAU+G(JI(J))*Q(JI(J),I)
    8         CONTINUE
            END IF
            IF(ND.GT.0) THEN
              DO 9 J=1,ND
              TAU=TAU+G(JA(J))*Q(JA(J),I)
    9         CONTINUE
            END IF
            IF(I.LT.NC) THEN
              IP1=I+1
              DO 10 J=IP1,NC
              TAU=TAU+W1(JA(J))*Q(JA(J),I)
   10         CONTINUE
            END IF
            W1(J1)=-TAU/Q(J1,I)
            K0=KBCON(I)
            IF(K0.EQ.0) THEN
              KBCON(I)=NSP1
              GFE0=G(J1)-W1(J1)
              WWW(KSPEC(J1))=GFE0
              IF(PP.OR.PD) THEN
                WRITE(6,5656) GFE0,J1
 5656            FORMAT(' DUAL SIMPLE ',1PD20.12,I5)
              END IF
              IF(GFE0.LT.WMIN) THEN
                NBMIN=1
                JBMIN(1)=-J1
                WMIN=GFE0
                ADDPHS=.TRUE.
              END IF
            ELSE
```

```
            IF(K0.NE.NSP1) NOMIN=.FALSE.
          END IF
   11     CONTINUE
C            IF NOMIN=TRUE, THERE ARE NO MULTIPLE SPECIES BLOCKS
C            FOR WHICH THE MULTIPLIER NEED BE EVALUATED.  IF
C            ADDPHS IS SET INITIALIZE FOR ADDPHS MODIFIED DIRECTION.
C            RETURN
C
          IF(NOMIN) GO TO 3000
C
C ---------------------------------------------------------------------
C         EVALUATION OF MULTIPLIERS FOR MULTIPLE SPECIES BLOCKS
C
          DO 100 ICON=NC0P1,NC
          NBLK=KBCON(ICON)
          IF(NBLK.EQ.NSP1) GO TO 100
          NSB=0
          NBLOCK=0
          NOMIN=.FALSE.
          CONCAV=.FALSE.
          NCON=0
          DO 13 I=1,NC
          IF(I.EQ.NC0P1) NCON=NSB+1
          IF(KBCON(I).EQ.NBLK) THEN
             KBCON(I)=NSP1
             J=JA(I)
              NSB=NSB+1
             JQX(NSB)=J
             JXQ(J)=NSB
             JIQ(NSB)=NSB
             MACB(NSB)=-NSB
             ZDUAL(NSB)=.FALSE.
             KK=KTYP(KSPEC(J))
             IF(KK.LT.-1) THEN
                IF(-KK.EQ.3.OR.-KK.EQ.5.OR.-KK.EQ.7) CONCAV=.TRUE.
                NB=NBLOCK
                NBLOCK=NBLOCK+1
                K0=KSPEC(J)
                KBLK(NBLOCK)=K0
                IF(NB.GT.0) THEN
                   DO 12 K=1,NB
                   IF(KBLK(K).EQ.KSPEC(J)) NBLOCK=NBLOCK-1
   12              CONTINUE
                END IF
                VTYP(NSB)=.TRUE.
                SL(NSB)=DEL3
                SLP(NSB)=DEL3P
                X(J)=PS(J)
                IF(X(J).LT.DEL3P) X(J)=DEL3
                GSAV(NSB)=U0RT(J)
                U0RT(J)=U0RT(J)-W1(J)
             ELSE
                VTYP(NSB)=.FALSE.
                XSAV(NSB)=X(J)
```

```
                X(J)=.01D0
                GSAV(NSB)=G(J)
                G(J)=G(J)-W1(J)
                NBLOCK=NBLOCK+1
                KBLK(NBLOCK)=-J
             END IF
          END IF
       13 CONTINUE
          GFE0=0.0D0
          ALPHA=0.0D0
          PTGA=0.0D0
          DO 14 KP=1,NBLOCK
          K=KBLK(KP)
          IF(K.GT.0) THEN
             WRITE(*,*) 'CALLING GRAD FROM DUAL @ LN 231'
             CALL GRAD(K,ALPHA,PTGA,GFE0)
          ELSE
             GFE0=GFE0+X(-K)*G(-K)
          END IF
       14 CONTINUE
          SVAR=.FALSE.
          KK=KBLK(1)
          IF(NBLOCK.EQ.1.AND.KK.GT.0) THEN
             IF(CONCAV) THEN
                JM=JMIN(KK)
                JX=JMAX(KK)
                DO 123 J=JM,JX
                IF(DABS(S(J)).GT.DEL3P)  THEN
                   SL(JXQ(J))=S(J)
                   SLP(JXQ(J))=S(J)*(1.0D0+DEL4)
                   IF(X(J).LT.SLP(JXQ(J))) X(J)=SL(JXQ(J))
                   SVAR=.TRUE.
                END IF
      123        CONTINUE
             END IF
          ELSE
          END IF
          GLOBLB=-1.0D+20
          ITER=0
          IFUN=1
          ISENS=0
      C
      C     INITIALIZE ORTHOGONAL Q MATRIX FOR BLOCK MINIMIZATION
      C
          NSBP1=NSB+1
          AN=NSB
          SQN=DSQRT(AN)
          AN=-1.0D0/(AN+SQN)
          BN=AN*(1.0D0+SQN)
          QB(1,1)=1.0D0+BN*(1.0D0+SQN)
          QB(1,NSBP1)=0.0D0
          DO 16 I=2,NSB
          DO 15 J=2,NSB
          QB(J,I)=AN
```

```
   15 CONTINUE
      QB(I,1)=BN
      QB(1,I)=BN
      QB(I,I)=1.0D0+AN
      QB(I,NSBP1)=0.0D0
   16 CONTINUE
      BB(1)=-1.0D0/SQN
      NCB=1
C
C     ADD ADDITIONAL CONSTRAINTS DUE TO DEGENERACY, IF ANY
C
      NINACT=NSB
      NDB=0
      IF(NCON.GT.1) THEN
         NCONM1=NCON-1
         DO 24 JQD=1,NCONM1
         JXD=JQX(JQD)
         SIG=0.0D0
         PI=0.0D0
         NCBP1=NCB+1
         DO 18 II=NCON,NSB
         JQ=(NCON+NSB)-II
         JX=JQX(JQ)
         I=MACT(JX)
         TAU=Q(JXD,I)
         IF(JQ.LT.NSB) THEN
            JQP1=JQ+1
            DO 17 JQ2=JQP1,NSB
            TAU=TAU-YS(JQ2)*Q(JQX(JQ2),I)
   17       CONTINUE
         END IF
         YS(JQ)=TAU/Q(JX,I)
   18    CONTINUE
         NODEG=.FALSE.
         DO 20 I=1,NSB
         TAU=-QB(JQD,I)
         DO 19 J=NCON,NSB
         TAU=TAU+YS(J)*QB(J,I)
   19    CONTINUE
         IF(I.LT.NCBP1) THEN
            PI=PI-BB(I)*TAU
         ELSE
            IF(DABS(TAU).GT.DEL4) NODEG=.TRUE.
            V(I)=TAU
            SIG=SIG+TAU*TAU
         END IF
   20    CONTINUE
         IF(NODEG) THEN
            NCB=NCBP1
            SIG=DSIGN(DSQRT(SIG),V(NCB))
            IF(NCB.EQ.NSB) THEN
               BB(NCB)=PI/V(NCB)
            ELSE
               V(NCB)=V(NCB)+SIG
```

```
                BB(NCB)=-PI/SIG
                PI=1.0D0/(V(NCB)*SIG)
                DO 23 J=1,NSB
                TAU=0.0D0
                DO 21 I=NCB,NSB
                TAU=TAU+QB(J,I)*V(I)
 21             CONTINUE
                TAU=TAU*PI
                DO 22 I=NCB,NSB
                QB(J,I)=QB(J,I)-TAU*V(I)
 22             CONTINUE
 23             CONTINUE
             END IF
          END IF
 24       CONTINUE
          DO 26 J=1,NSB
          REDUN=.TRUE.
          DO 25 I=NCBP1,NSB
          IF(DABS(QB(J,I)).GT.DEL4) REDUN=.FALSE.
 25       CONTINUE
          IF(NSB.EQ.NCB) REDUN=.TRUE.
          IF(REDUN) THEN
             NDB=NDB+1
             MACB(JIQ(NINACT))=MACB(J)
             JIQ(-MACB(J))=JIQ(NINACT)
             NINACT=NINACT-1
             JAQ(NDB)=J
             MACB(J)=NDB
          END IF
 26       CONTINUE
       END IF
       NCB0=NCB
       NDBC=NDB
       NCB0P1=NCB0+1
       NCBC=NCB
       NSBC=NSB
       NCB0C=NCB0
C
C      ADD CONSTRAINTS FOR SMALL SPECIES, IF ANY
C
       DO 27 JQ=1,NSB
       IF(X(JQX(JQ)).LT.SLP(JQ)) THEN
          IF(MACB(JQ).LT.0) THEN
             WRITE(*,*) 'CALLING DADD @ LN 368'
             CALL DADD(JQ)
          END IF
       END IF
 27    CONTINUE
       NORMAL=.FALSE.
       ADDCON=.FALSE.
C
C      TEST FOR FEASIBILITY
C
 1000  SENS=.FALSE.
```

```
      DO 81 I=1,NCB0
      TAU=BB(I)
      DO 80 JQ=1,NSB
      TAU=TAU-X(JQX(JQ))*QB(JQ,I)
   80 CONTINUE
      IF(DABS(TAU).GT.DEL4) SENS=.TRUE.
      YS(I)=TAU
   81 CONTINUE
C
C     IF FEASIBLE COMPUTE NORMAL DESCENT DIRECTION, OTHERWISE
C     COMPUTE A SENSITIVITY DIRECTION TO FEASIBLE DOMAIN
      IF(SENS) THEN
         ISENS=ISENS+1
         IF(ISENS.GT.100) THEN
            WRITE(6,9973) ISENS
 9973        FORMAT(' SUBROUTINE DUAL #9973, ISENS=',I5)
            ERROR=.TRUE.
            RETURN
         END IF
         NCMIN=NCB0P1
         NJMIN=1
         AMAX=1.0D0
      ELSE
         NORMAL=.TRUE.
         GO TO 1002
      END IF
C
C     MODIFIED STEP DIRECTION CALCULATION
C
 1001 NCB=NCBC
      NDB=NDBC
      NU=NSB-NCB
      NUP1=NU+1
      NUT=NUP1
      NUP2=NU+2
      NINACT=NU-NDB+NCB0
      PTG=0.0D0
      J0Q=0
      AMAX=1.0D0
      IF(NCB.GE.NCMIN) THEN
         DO 28 I=NCMIN,NCB
         JQ=JAQ(I)
         JX=JQX(JQ)
         IF(DABS(P(JX)).GT.DEL4) THEN
            TAU=P(JX)
            PTG=PTG+TAU*G(JX)
            QB(JQ,NSBP1)=TAU
         ELSE
            TAU=0.0D0
         END IF
         IF(I.GT.NJMIN) THEN
            IM1=I-1
            DO 82 J=NJMIN,IM1
            TAU=TAU-QB(JQ,J)*YS(J)
```

```
 82          CONTINUE
          END IF
          YS(I)=TAU/QB(JQ,I)
 28       CONTINUE
       END IF
       IF(NDB.GT.0) THEN
          DO 30 J=1,NDB
          JQ=JAQ(J)
          JX=JQX(JQ)
          TAU=0.0D0
          DO 29 I=NJMIN,NCB
          TAU=TAU+QB(JQ,I)*YS(I)
 29       CONTINUE
          P(JX)=TAU
          PTG=PTG+TAU*G(JX)
          QB(JQ,NSBP1)=TAU
          IF(VTYP(JQ)) THEN
             IF(X(JX)+AMAX*TAU.LT.SL(JQ)*(1.0D0-DEL4)) THEN
                AMAX=DMAX1( (SL(JQ)-X(JX))/TAU, 0.0D0)
                J0Q=JQ
             END IF
          ELSE
             IF(X(JX)+AMAX*TAU.LT.0.0D0) THEN
                AMAX=DMAX1( -X(JX)/TAU, 0.0D0)
                J0Q=JQ
             END IF
          END IF
 30       CONTINUE
          IF(SENS.AND.J0Q.GT.0) THEN
             J1Q=0
             DO 32 II=NCB0P1,NCB
             I=(NCB0P1+NCB)-II
             J2Q=JAQ(I)
             TAU=QB(J0Q,I)
             IF(I.LT.NCB) THEN
                IP1=I+1
                DO 31 I2=IP1,NCB
                TAU=TAU-YS(I2)*QB(JAQ(I2),I)
 31             CONTINUE
             END IF
             YS(I)=TAU/QB(J2Q,I)
             IF(YS(I).GT.DEL4.AND.J2Q.GT.J1Q) THEN
               J1Q=J2Q
             END IF
 32          CONTINUE
             IF(J1Q.EQ.0) THEN
                WRITE(6,9999) J0Q
9999             FORMAT(I5,' SUBROUTINE DUAL NO FEASIBLE SOLUTION')
                ERROR=.TRUE.
                RETURN
             ELSE
                WRITE(*,*) 'CALLING DDEL @ LN 485'
                CALL DDEL(J1Q)
                WRITE(*,*) 'CALLING DADD @ LN 487'
```

```
                  CALL DADD(J0Q)
                  P(JQX(J0Q))=0.0D0
                  QB(J0Q,NSBP1)=0.0D0
               END IF
               GO TO 1001
            ELSE IF(NINACT.EQ.0) THEN
               GO TO 1006
            END IF
         END IF
         IF (NINACT.EQ.0) GO TO 1006
         DO 34 J=1,NINACT
         JQ=JIQ(J)
         TAU=0.0D0
         DO 33 I=NJMIN,NCB
         TAU=TAU+QB(JQ,I)*YS(I)
   33    CONTINUE
         QB(JQ,NSBP1)=TAU
   34    CONTINUE
         DO 36 I1=1,NUP2
         DO 35 I2=1,I1
         H(I2,I1)=0.0D0
   35    CONTINUE
   36    CONTINUE
         H(NUP1,NUP2)=-1.0D0
         GO TO 1005
C
C        NEWTON STEP DIRECTION CALCULATION
C
 1002    NCB=NCBC
         NDB=NDBC
         NU=NSB-NCB
         NUT=NU
         NUP1=NU+1
         NUP2=NU+2
         NINACT=NU-NDB+NCB0
         IF(NU.EQ.0) THEN
            SUBMIN=.TRUE.
            Q3DMAX=0.0D0
            POSDEF=.TRUE.
            GO TO 1004
         END IF
         POSDEF=.FALSE.
C
C        REDUCED GRADIENT VECTOR CALCULATION
C
 1003    Q3DMAX=0.0D0
         J0Q=0
         PTG=0.0D0
         AMAX=0.999D0
         DO 39 L=1,NU
         LP1=L+1
         DO 37 L2=1,L
         H(L2,LP1)=0.0D0
   37    CONTINUE
```

```
      LQ=L+NCB
      TAU=0.0D0
      DO 38  JJ=1,NINACT
      TAU=TAU+G(JQX(JIQ(JJ)))*QB(JIQ(JJ),LQ)
   38 CONTINUE
      H(L,NUP2)=TAU
      IF(DABS(TAU).GT.Q3DMAX) Q3DMAX=DABS(TAU)
   39 CONTINUE
      IF(Q3DMAX.LT.DEL1) THEN
         SUBMIN=.TRUE.
      ELSE
         SUBMIN=.FALSE.
      END IF
C
C     BLOCK SPECIES MULTIPLIER EVALUATION, IF ANY
C
 1004 IF(POSDEF) THEN
         GFELB=GFE0-NSB*Q3DMAX
         IF(SUBMIN) THEN
            DO 85 JQ=1,NSB
            ZDUAL(JQ)=.FALSE.
   85       CONTINUE
         END IF
         IF(NCB.GT.NCB0) THEN
            JADD=0
            WLB=1.0D+20
            WM=-DMAX1(Q3DMAX,1.0D-03)
            DO 43 II=NCB0P1,NCB
            I=(NCB0P1+NCB)-II
            JQ1=JAQ(I)
            JX1=JQX(JQ1)
            TAU=0.0D0
            IF(NINACT.GT.0) THEN
               DO 40 J=1,NINACT
               TAU=TAU+G(JQX(JIQ(J)))*QB(JIQ(J),I)
   40          CONTINUE
            END IF
            IF(NDB.GT.0) THEN
               DO 41 J=1,NDB
               TAU=TAU+G(JQX(JAQ(J)))*QB(JAQ(J),I)
   41          CONTINUE
            END IF
            IF(I.LT.NCB) THEN
               IP1=I+1
               DO 42 J=IP1,NCB
               TAU=TAU+YS(J)*QB(JAQ(J),I)
   42          CONTINUE
            END IF
            YS(I)=-TAU/QB(JQ1,I)
            SIG=G(JX1)-YS(I)
            IF(SIG.GT.0) THEN
               GFELB=GFELB-DEL3*SIG
            ELSE
               GFELB=GFELB+SIG
```

```
            END IF
            IF(SIG.LT.WM.AND..NOT.ZDUAL(JQ1)) THEN
               WM=SIG
               JADD=JQ1
            END IF
 43         CONTINUE
            IF(JADD.GT.0) THEN
               NCMIN=MACB(JADD)
               ZDUAL(JADD)=.TRUE.
               NJMIN=NCMIN
               ADDCON=.TRUE.
               NORMAL=.FALSE.
               JXADD=JQX(JADD)
               IF(VTYP(JADD)) THEN
                    K0=KSPEC(JXADD)
                  JM=JMIN(K0)
                  JX=JMAX(K0)
                  TAU=0.0D0
                  DO 44 J=JM,JX
                  TAU=TAU+X(J)
 44               CONTINUE
                  PI=DEXP(DMIN1(-WM,17.0D0))
                  P(JXADD)=DMIN1(X(JXADD)*PI, 0.1D0*TAU)
               ELSE
                    P(JXADD)=0.01D0
               END IF
               GO TO 1001
            END IF
         END IF
         IF(GFE0.LT.WMIN) THEN
             IF(SUBMIN) THEN
                WMIN=GFE0
                NBMIN=0
                DO 65 KP=1,NBLOCK
                NBMIN=NBMIN+1
                JBMIN(NBMIN)=KBLK(KP)
 65             CONTINUE
                ADDPHS=.TRUE.
                GO TO 2001
            END IF
         ELSE
            IF(GFELB.GT.WMIN.OR.SUBMIN) GO TO 2001
         END IF
      END IF
C
C     PROJECTED HESSIAN MATRIX CALCULATION
C
1005  DO 50 KP=1,NBLOCK
      K=KBLK(KP)
      IF(K.GT.0) THEN
         JM=JMIN(K)
         JX=JMAX(K)
         WRITE(*,*) 'CALLING HESS FROM DUAL @ LN 649'
         CALL HESS(K)
```

```
            DO 49 L1=1,NU
            LQ1=L1+NCB
            DO 46 J1=JM,JX
            TAU=0.0D0
            DO 45 J2=JM,JX
            JP=MAX0(J1,J2)
            TAU=TAU+H(JP,J1+J2-JP)*QB(JXQ(J2),LQ1)
   45       CONTINUE
            YS(JXQ(J1))=TAU
   46       CONTINUE
            DO 48 L2=L1,NUT
            LQ2=L2+NCB
            TAU=H(L1,L2+1)
            DO 47 J1=JM,JX
            JQ=JXQ(J1)
            TAU=TAU+YS(JQ)*QB(JQ,LQ2)
   47       CONTINUE
            H(L1,L2+1)=TAU
   48       CONTINUE
   49       CONTINUE
         END IF
   50    CONTINUE
C
C        REDUCED DIRECTION VECTOR CALCULATION
C
         POSDEF=.TRUE.
         GAMMA=0.0D0
         ZETA=0.0D0
         THETA=0.0D0
         DO 52 I=1,NU
         IP1=I+1
         IF(DABS(H(I,IP1)).GT.GAMMA) GAMMA=DABS(H(I,IP1))
         IF(I.LT.NU) THEN
            DO 51 J=IP1,NU
            IF(DABS(H(I,J+1)).GT.ZETA) ZETA=DABS(H(I,J+1))
   51       CONTINUE
            IF(I.EQ.1) THETA=ZETA
         END IF
   52    CONTINUE
         DELTA=DEL4*DMAX1(1.0D0,GAMMA,ZETA)
         BETA2=1.0D0/DMAX1(GAMMA,DELTA,ZETA/NU)
         TAU=DMAX1(DABS(H(1,2)),DELTA,THETA*THETA*BETA2)
         IF(TAU.GT.H(1,2)) POSDEF=.FALSE.
         H(1,2)=1.0D0/TAU
         IF(NU.EQ.1) THEN
            H(1,3)=H(1,3)*H(1,2)
         ELSE
            DO 56 J=2,NU
            JM1=J-1
            JP1=J+1
            JP2=J+2
            PHI=H(J,JP1)
            DO 53 K=1,JM1
            PI=H(K,JP1)*H(K,K+1)
```

```
            PHI=PHI-PI*H(K,JP1)
            H(K,JP1)=PI
   53       CONTINUE
            THETA=0.0D0
            DO 55 I=JP2,NUP2
            TAU=H(J,I)
            DO 54 K=1,JM1
            TAU=TAU-H(K,JP1)*H(K,I)
   54       CONTINUE
            H(J,I)=TAU
            IF(I.LT.NUP2) THEN
               IF(DABS(TAU).GT.THETA) THETA=DABS(TAU)
            END IF
   55       CONTINUE
            SIG=DMAX1(DABS(PHI),DELTA,THETA*THETA*BETA2)
            IF(SIG.GT.PHI) POSDEF=.FALSE.
            H(J,JP1)=1.0D0/SIG
   56       CONTINUE
            H(NU,NUP2)=H(NU,NUP2)*H(NU,NUP1)
            DO 58 JMN=2,NU
            J=NUP1-JMN
            JP1=J+1
            TAU=H(J,NUP2)*H(J,JP1)
            DO 57 K=JP1,NU
            TAU=TAU-H(J,K+1)*H(K,NUP2)
   57       CONTINUE
            H(J,NUP2)=TAU
   58       CONTINUE
         END IF
C
C        TEST FOR ACCIDENTAL MAXIMUM
C
         IF(SUBMIN.AND.NORMAL) THEN
            IF(POSDEF) GO TO 1004
            WRITE(6,9991)
 9991        FORMAT(' SUBROUTINE DUAL 9991: LOCAL MAXIMUM')
            ERROR=.TRUE.
            RETURN
         END IF
C
C        FEASIBLE DIRECTION VECTOR CALCULATION AND MAXIMUM STEP
C        LENGTH
C
         DO 60 JJ=1,NINACT
         JQ=JIQ(JJ)
         JX=JQX(JQ)
         TAU=0.0D0
         DO 59 I=1,NUT
         TAU=TAU-QB(JQ,I+NCB)*H(I,NUP2)
   59    CONTINUE
         P(JX)=TAU
         PTG=PTG+TAU*G(JX)
         IF(VTYP(JQ)) THEN
            IF(X(JX)+AMAX*TAU.LT.SL(JQ)*(1.0D0-DEL4)) THEN
```

```
                AMAX=DMAX1((SL(JQ)-X(JX))/TAU,0.0D0)
                J0Q=JQ
            END IF
        ELSE
            IF(X(JX)+AMAX*TAU.LT.0.0D0) THEN
                AMAX=DMAX1( -X(JX)/TAU,0.0D0)
                J0Q=JQ
            END IF
        END IF
   60   CONTINUE
C
C     TEST FOR NEGATIVE DERIVATIVE ON DESCENT DIRECTIONS (IERR=1)
C
 1006   IF(SENS) GO TO 1007
        IF(PTG.GT.0.0D0) THEN
            IERR=1
            IF(ADDCON.AND..NOT.SUBMIN) THEN
                P(JQX(JADD))=0.0D0
                QB(JADD,NSBP1)=0.0D0
                IF(NDB.GT.0) THEN
                    DO 76 JQ=1,NDB
                    P(JQX(JAQ(JQ)))=0.0D0
                    QB(JAQ(JQ),NSBP1)=0.0D0
   76               CONTINUE
                END IF
                ADDCON=.FALSE.
                NORMAL=.TRUE.
                GO TO 1002
            ELSE
                WRITE(6,9992) JADD,PTG
 9992            FORMAT(' SUBROUTINE DUAL 9992: PTG=',I2,1PD12.4)
                ERROR=.TRUE.
                RETURN
            END IF
        END IF
C
C       LIMIT MAXIMUM CHANGE IN SOLUTION PHASE VARIABLE
C
        IF(J0Q.GT.0) THEN
            IF(VTYP(J0Q)) THEN
                JX=JQX(J0Q)
                IF((DEL6-1.0D0)*X(JX).GT.AMAX*P(JX)) THEN
                        AMAX=(DEL6-1.0D0)*X(JX)/P(JX)
                        IF(X(JX)+AMAX*P(JX).GT.SL(J0Q))  J0Q=0
                END IF
            END IF
        END IF
C
C       GRADIENT AT MAXIMUM STEP LENGTH
C
 1007   IFUN=IFUN+1
        IF(IFUN.GT.NFUN) THEN
                    WRITE(6,9993) IFUN
 9993               FORMAT(' SUBROUTINE DUAL 9993: IFUN=',I6)
```

```
                ERROR=.TRUE.
                RETURN
        END IF
        ALPHA=AMAX
        PTGA=0.0D0
        GFEA=0.0D0
        DO 61 KP=1,NBLOCK
        K=KBLK(KP)
        IF(K.GT.0) THEN
          WRITE(*,*) 'CALLING GRAD FROM DUAL @ LN 821'
          CALL GRAD(K,ALPHA,PTGA,GFEA)
        ELSE
          GFEA=GFEA+(X(-K)+ALPHA*P(-K))*G(-K)
          PTGA=PTGA+P(-K)*G(-K)
        END IF
   61   CONTINUE
        IF(SENS) GO TO 1010
C
C       TEST FOR MAXIMUM STEP LENGTH ON DESCENT DIRECTIONS
C         IF FREE ENERGY DECREASE TAKE FULL STEP
C
        IF(GFEA.LT.GFE0) GO TO 1010
C
C         IF FUNCTION DECREASING AND MAXIMUM IMPROVEMENT
C         OF FREE ENERGY IS SMALL COMPARED TO GFE0 TAKE
C         FULL STEP
C
        IF(PTGA.LE.0.0D0) THEN
           IF(DABS(ALPHA*PTG).LT.DABS(GFE0)*DEL4) GO TO 1010
           WRITE(6,8811) J0Q,ADDCON,AMAX,PTG,PTGA,GFE0,GFEA
 8811       FORMAT(' DUAL 8811 ',2I5,1P3D12.3,1P2D24.16)
           GO TO 1010
        END IF
C
C       FUNCTION INCREASING AT MAXIMUM STEP   LINESEARCH
C
        AMIN=0.0D0
        ETA=DEL5*DABS(PTG)
 1008   AMAX=ALPHA
        PTG2=PTGA
        IF(DABS(PTGA).LT.ETA) THEN
           IF(GFEA.LT.GFE0) GO TO 1009
           IF(PTGA.LE.0.0D0) GO TO 1009
        END IF
        ALPHA=AMIN-(AMAX-AMIN)*PTG/(PTG2-PTG)
        IFUN=IFUN+1
        IF(IFUN.GT.NFUN) THEN
           WRITE(6,9993) IFUN
           ERROR=.TRUE.
           RETURN
        END IF
        PTGA=0.0D0
        GFEA=0.0D0
        DO 62 KP=1,NBLOCK
```

```
      K=KBLK(KP)
      IF(K.GT.0) THEN
        WRITE(*,*) 'CALLING GRAD FROM DUAL @ LN 868'
        CALL GRAD(K,ALPHA,PTGA,GFEA)
      ELSE
        GFEA=GFEA+(X(-K)+ALPHA*P(-K))*G(-K)
        PTGA=PTGA+P(-K)*G(-K)
      END IF
   62 CONTINUE
      IF(PTGA*PTG2.LT.0.0D0) THEN
         PTG=PTG2
         AMIN=AMAX
      ELSE
         PTG=PTG*0.5D0
      END IF
      GO TO 1008
 1009 IF(J0Q.GT.0) THEN
         TAU=X(JQX(J0Q))+ALPHA*P(JQX(J0Q))
         IF(VTYP(J0Q)) THEN
            IF(TAU.GT.SLP(J0Q)) J0Q=0
         ELSE
            IF(TAU.GT.DEL4) J0Q=0
         END IF
      END IF
C
C     UPDATE X = X + ALPHA*P
C
 1010 ITER=ITER+1
      DELGX=GFE0-GFEA
      GFE0=GFEA
      IF(NORMAL) THEN
         DO 63 J=1,NINACT
         JX=JQX(JIQ(J))
         X(JX)=X(JX)+ALPHA*P(JX)
   63    CONTINUE
         IF(PD) THEN
               IF(J0Q.GT.0) THEN
                 J0=JQX(J0Q)
               ELSE
                 J0=0
               END IF
               WRITE(6,5001) ITER,IFUN,J0,GFE0,ALPHA
 5001          FORMAT(' DUAL NORMAL',3I5,1PD25.17,1P2D20.8)
               DO 6000 JJ=1,NSB
               J=JQX(JJ)
               WRITE(6,5002) JJ,J,VTYP(JJ),MACB(JJ),X(J),G(J),P(J)
 6000          CONTINUE
 5002          FORMAT(4I5,F25.21,1P3D13.4)
          END IF
         IF(J0Q.EQ.0) THEN
            GO TO 1003
         ELSE
            WRITE(*,*) 'CALLING DADD @ LN 918'
            CALL DADD(J0Q)
```

```
            J0X=JQX(J0Q)
            P(J0X)=0.0D0
            QB(J0Q,NSBP1)=0.0D0
            IF(NDB.LT.NDBC) THEN
               NDBP1=NDB+1
               DO 71 JD=NDBP1,NDBC
               P(JQX(JAQ(JD)))=0.0D0
               QB(JAQ(JD),NSBP1)=0.0D0
71             CONTINUE
            END IF
            IF(VTYP(J0Q)) THEN
                X(J0X)=SL(J0Q)
            ELSE
                X(J0X)=0.0D0
            END IF
            GO TO 1002
         END IF
      ELSE
         DO 64 JQ=1,NSB
         JX=JQX(JQ)
         X(JX)=X(JX)+ALPHA*P(JX)
         YS(JX)=P(JX)
         P(JX)=0.0D0
         QB(JQ,NSBP1)=0.0D0
64       CONTINUE
         IF(ADDCON) THEN
            WRITE(*,*) 'CALLING DDEL @ LN 946'
            CALL DDEL(JADD)
         END IF
         IF(J0Q.GT.0) THEN
            IF(MACB(J0Q).LT.0) THEN
                WRITE(*,*)'CALLING DADD @ LN 951'
                CALL DADD(J0Q)
            END IF
         END IF
         IF(PD) THEN
               IF(J0Q.GT.0) THEN
                 J0=JQX(J0Q)
               ELSE
                 J0=0
               END IF
               IF(ADDCON) THEN
               WRITE(6,5003) ITER,IFUN,J0,GFE0,ALPHA
               ELSE
                    WRITE(6,5004) ITER,IFUN,J0,GFE0,ALPHA
               END IF
5003           FORMAT(' DUAL ADDCON',3I5,1PD25.17,1PD20.8)
5004           FORMAT(' DUAL SENS',3I5,1PD25.17,1PD20.8)
               DO 6001 JJ=1,NSB
               J=JQX(JJ)
               WRITE(6,5002) JJ,J,VTYP(JJ),MACB(JJ),X(J),G(J),YS(J)
6001           CONTINUE
         END IF
         IF(SENS) THEN
```

```
            GO TO 1000
          ELSE
            ADDCON=.FALSE.
            NORMAL=.TRUE.
            GO TO 1002
          END IF
        END IF
C
C       REPLACE X AND G VECTORS FOR PRIMAL
 2001   DO 67 JQ=1,NSB
        JX=JQX(JQ)
        IF(MACB(JQ).GT.0.AND.SUBM.AND..NOT.ADDPHS) THEN
          WRITE(6,3100) JX,MACB(JQ),X(JX),S(JX)
 3100      FORMAT('WARNING: DUAL SPECIES ON BOUNDARY AT GLOBAL
     1 MINIMUM',2I5,1P2D12.3)
        END IF
        IF(.NOT.SVAR.OR.NBLOCK.EQ.1) PS(JX)=X(JX)
        P(JX)=0.0D0
        IF(VTYP(JQ)) THEN
          U0RT(JX)=GSAV(JQ)
          G(JX)=G(JX)+W1(JX)
          X(JX)=0.0D0
        ELSE
          X(JX)=XSAV(JQ)
          G(JX)=GSAV(JQ)
        END IF
   67   CONTINUE
          IF(PP.OR.PD) THEN
            WRITE(6,5657) GFE0,(JQX(JQ),JQ=1,NSB)
 5657       FORMAT(' DUAL BLOCK ',1PD20.12,96I3)
          END IF
C
  100 CONTINUE
C -----------------------------------------------------------------
 3000 IF(ADDPHS) THEN
        IF(NBMIN.EQ.1.AND.JBMIN(1).LT.0) THEN
          J1=-JBMIN(1)
          NCM=MACT(J1)
          K0=KSPEC(J1)
          IF(KTYP(K0).GT.1) THEN
            JM=JMIN(K0)
            JX=JMAX(K0)
            TAU=0.0D0
            DO 97 J=JM,JX
            TAU=TAU+X(J)
   97       CONTINUE
            PI=DEXP( DMIN1(-WMIN,20.0D0) )
            SIG=X(J1)*PI
            IF(TAU.GE.DEL2) THEN
              TAU=DMIN1( SIG ,0.1D0*TAU,.01D0)
              IF((TAU-X(J1)).LE.0.0D0) TAU=SIG
              TAU=TAU-X(J1)
            ELSE
              TAU=DMAX1(SIG-X(J1)
```

```
     1                ,DMIN1(0.1D0*X(J1),DEL2*0.01D0))
                   END IF
                   P(J1)=DMAX1(TAU,0.0D0)
               ELSE
                   PS(J1)=DMIN1(-WMIN,1.0D0)
                   P(J1)=PS(J1)
                   IF(KTYP(K0).LT.0) THEN
                       NPP=NPP+1
                       KPH(NPP)=K0
                   END IF
               END IF
           ELSE
               NCM=NC
               TEST=.FALSE.
               ALPHA=DMIN1(-WMIN,1.0D0)
               SIG=0.0D0
               DO 68 KP=1,NBMIN
               K0=JBMIN(KP)
               IF(K0.GT.0) THEN
                   JM=JMIN(K0)
                   JX=JMAX(K0)
                   TEST=.TRUE.
               ELSE
                   JM=-K0
                   JX=-K0
                   K0=KSPEC(JM)
               END IF
               IF(KTYP(K0).LT.0) THEN
                   NPP=NPP+1
                   KPH(NPP)=K0
               END IF
               DO 66 J=JM,JX
               SIG=SIG+X(J)
               IF(PS(J).GT.DELLB) THEN
                   P(J)=ALPHA*PS(J)
                   II=MACT(J)
                   IF(II.GT.NC0.AND.II.LT.NCM) NCM=II
               END IF
    66         CONTINUE
    68         CONTINUE
               IF(.NOT.TEST) THEN
                   SIG=DMAX1(SIG,DEL2)
                   DO 160 KP=1,NBMIN
                   J=-JBMIN(KP)
                   IF(J.LT.0) STOP
                   IF(KTYP(KSPEC(J)).LT.2) GO TO 170
                   P(J)=P(J)*SIG/ALPHA
   160             CONTINUE
   170             CONTINUE
               END IF
           END IF
       END IF
       RETURN
       END
```

```
C-----------------------------------------------------------------------
C
C  THIS SUBROUTINE ADDS A SPECIES CONSTRAINT FOR SPECIES J0 IN
C  THE SUB-MINIZATION PROBLEM. DEGENERATE CONSTRAINTS ARE
C  HANDLED EXACTLY AS IN CONADD.F (CONSTRAINT ADDITION FOR THE
C  MAIN PROBLEM).
C
C-----------------------------------------------------------------------
      SUBROUTINE DADD(J0)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      LOGICAL REDUN,NODEG
      COMMON/DUALB/Q(96,96),MACT(96),JA(96),JI(96),NS,NC,ND,NC0
      DIMENSION V(96)
      PARAMETER(DEL4=1.0D-13)
         IF(NS.EQ.NC) GO TO 99
      NC=NC+1
      SIG=0.0D0
      NODEG=.TRUE.
      DO 1 I=NC,NS
      V(I)=Q(J0,I)
      IF(DABS(V(I)).GT.DEL4) NODEG=.FALSE.
      Q(J0,I)=0.0D0
      SIG=SIG+V(I)*V(I)
1     CONTINUE
         IF(NODEG) GO TO 99
      SIG=DSIGN(DSQRT(SIG),V(NC))
      NINACT=(NS+NC0)-(NC+ND)
      JI(-MACT(J0))=JI(NINACT+1)
      MACT(JI(NINACT+1))=MACT(J0)
      JA(NC)=J0
      MACT(J0)=NC
      IF(NS.EQ.NC) THEN
         Q(J0,NC)=V(NC)
         IF(NINACT.GT.0) THEN
           DO 2 J=1,NINACT
           ND=ND+1
           JA(ND)=JI(J)
           MACT(JI(J))=ND
2          CONTINUE
         END IF
      ELSE
         Q(J0,NC)=-SIG
         IF(NINACT.GT.0) THEN
            NDS=ND
            V(NC)=V(NC)+SIG
            PI=1.0D0/(V(NC)*SIG)
            DO 5 MI=1,NINACT
            J=JI(MI)
            TAU=0.0D0
            DO 3 I=NC,NS
            TAU=TAU+V(I)*Q(J,I)
3           CONTINUE
            TAU=TAU*PI
```

```
                REDUN=.TRUE.
                DO 4 I=NC,NS
                T=Q(J,I)-V(I)*TAU
                IF(DABS(T).GT.DEL4) THEN
                   Q(J,I)=T
                   IF(I.GT.NC) REDUN=.FALSE.
                ELSE
                   Q(J,I)=0.0D0
                END IF
    4           CONTINUE
                IF(REDUN) THEN
                   ND=ND+1
                   JA(ND)=J
                END IF
    5           CONTINUE
                IF(ND.GT.NDS) THEN
                   NDS=NDS+1
                   DO 6 JD=NDS,ND
                   J=JA(JD)
                   IF(NINACT.GT.0) THEN
                      JI(-MACT(J))=JI(NINACT)
                      MACT(JI(NINACT))=MACT(J)
                      NINACT=NINACT-1
                   END IF
                   MACT(J)=JD
    6              CONTINUE
                END IF
             END IF
          END IF
          RETURN
C
C -------------------------------------------------------------------
C       ATTEMPT TO ADD A CONSTRAINED VARIABLE
C
   99   WRITE(6,9999) J0
 9999   FORMAT(' DADD ERROR',I5)
        STOP
        END
```

```
C-----------------------------------------------------------------------
C
C   THIS SUBROUTINE DELETES A SPECIES CONSTRAINT IN THE SUB-
C   MINIMIZATION PROBLEM. IT IS VIRTUALLY IDENTICAL TO CONDEL.F
C   FOR THE MAIN PROBLEM.
C
C-----------------------------------------------------------------------
      SUBROUTINE DDEL(JR)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      COMMON/DUALB/Q(96,96),MACT(96),JA(96),JI(96),NS,NC,ND,NC0
      PARAMETER(DEL4=1.0D-13)
      MREM=MACT(JR)
           IF(MREM.LE.NC0) GO TO 99
      NC=NC-1
      NINACT=(NS+NC0)-(NC+ND)
      JI(NINACT)=JR
      MACT(JR)=-NINACT
      IF(MREM.LE.NC) THEN
        DO 4 MM=MREM,NC
        MP1=MM+1
        J0=JA(MP1)
        JA(MM)=J0
        MACT(J0)=MM
        CMM=Q(J0,MM)
        CMP1=Q(J0,MP1)
        SIG=DSIGN(DSQRT(CMM*CMM+CMP1*CMP1),CMM)
        CMM=CMM+SIG
        Q(J0,MM)=-SIG
        Q(J0,MP1)=0.0D0
        V1=1.0D0/SIG
        V2=CMP1/(SIG*CMM)
        IF(NINACT.GT.0) THEN
          DO 1 I=1,NINACT
          J=JI(I)
          TAU=Q(J,MM)*V1+Q(J,MP1)*V2
          Q(J,MM)=Q(J,MM)-CMM*TAU
          Q(J,MP1)=Q(J,MP1)-CMP1*TAU
1         CONTINUE
        END IF
        IF(ND.GT.0) THEN
          DO 2 I=1,ND
          J=JA(I)
          TAU=Q(J,MM)*V1+Q(J,MP1)*V2
          Q(J,MM)=Q(J,MM)-CMM*TAU
          Q(J,MP1)=Q(J,MP1)-CMP1*TAU
2         CONTINUE
        END IF
        IF(MM.LT.NC) THEN
          DO 3 I=MP1,NC
          J=JA(I+1)
          TAU=Q(J,MM)*V1+Q(J,MP1)*V2
          Q(J,MM)=Q(J,MM)-CMM*TAU
          Q(J,MP1)=Q(J,MP1)-CMP1*TAU
```

```
    3        CONTINUE
           END IF
    4      CONTINUE
         END IF
         IF(ND.GT.0) THEN
           NDS=ND
           NCP1=NC+1
           ND=0
           DO 5 JD=1,NDS
           JX=JA(JD)
           JA(JD)=0
           T=Q(JX,NCP1)
           IF(DABS(Q(JX,NCP1)).LT.DEL4) THEN
             ND=ND+1
             JA(ND)=JX
             MACT(JX)=ND
             Q(JX,NCP1)=0.0D0
           ELSE
             NINACT=NINACT+1
             JI(NINACT)=JX
             MACT(JX)=-NINACT
           END IF
    5      CONTINUE
         END IF
         RETURN
C
C -----------------------------------------------------------------------
C       ATTEMPT TO DELETE A NON-EXPLICITLY CONSTRAINED SPECIES
C
   99    WRITE(6,9999) JR
 9999    FORMAT(' DDEL ERROR',I5)
         STOP
         END
```

```
C-----------------------------------------------------------------------
C
C     THIS SUBROUTINE PRINTS THE SOLUTION TO EACH PROBLEM.
C
C-----------------------------------------------------------------------
      SUBROUTINE PHSPRT(SCALE,GFEA,ITER,IFUN,IREFIN,Q2MAX,Q3MAX)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 Z
      CHARACTER*24 A1
      COMMON/LABEL/Z(96),U0RT(96),A1(96)
      COMMON/PHASE/R11(96,16),JMIN(96),JMAX(96),KSPEC(96),KTYP(96)
     1,NP0C,N00C
      COMMON/CONST/Q(96,96),MACT(96),JA(96),JI(96),NSC,NCC,NDC,NC0C
      COMMON/SEARCH/X(96),P(96),G(96),W0(96),H(64,64),B(96)
      COMMON/AQSOL/JP(46),JMC,JXC,JMA,JXA,JMB,JXB
      COMMON/MMM/MUL,WWW(96)
      COMMON/PH/LACT,NFILE,P0
      COMMON/DBASE/NCAT(18),NANI(18),ICAT,IANI,NCD,NAD,NND,TEMP
      LOGICAL MUL
      PARAMETER(WH2O=1.80153D-02)
      NPHAS0=NP0C
      NC0=NC0C
      CONC2=0.0D0
      SCALEI=1.0D0/SCALE
      IF(MUL) THEN
          WRITE(6,2020)
 2020      FORMAT(/,'PURE PHASE MULTIPLIERS')
          DO 11 K=1,NPHAS0
          IF(KTYP(K).EQ.-1) THEN
              J=JMIN(K)
              WRITE(6,2021) J,A1(J),WWW(K)
 2021          FORMAT(I5,2X,A24,1PD20.12)
          END IF
   11     CONTINUE
      END IF
      GFE=0.0D0
      GFEA=GFEA*SCALEI
      WRITE(6,3002)
      DO 10 K=1,NPHAS0
      IF(KTYP(K).GT.0) THEN
          JM=JMIN(K)
          JX=JMAX(K)
          X(JM)=X(JM)*SCALEI
          GFE=GFE+X(JM)*G(JM)
          IF(KTYP(K).EQ.1) THEN
              CONC=1.0D0/X(JM)
          ELSE IF(K.EQ.1) THEN
              CONC=1.0D0/(X(JM)*WH2O)
              DO 20 J=JM+1,JX
              X(J)=X(J)*SCALEI
              GFE=GFE+X(J)*G(J)
   20         CONTINUE
          ELSE IF(KTYP(K).EQ.2) THEN
```

```
            CONC=X(JM)
            DO 21 J=JM+1,JX
            X(J)=X(J)*SCALEI
            CONC=CONC+X(J)
            GFE=GFE+X(J)*G(J)
 21         CONTINUE
            CONC=1.0D0/CONC
         END IF

         XI=0.0D0
         DO 30 J=JM,JX
         TAU=G(J)-U0RT(J)

         WRITE(6,3000) J,A1(J),X(J),TAU,X(J)*CONC,X(J)*CONC

         XI=XI+Z(J)*Z(J)*X(J)*CONC
 30      CONTINUE
         XI=0.5D0*XI

      IF(KTYP(K).GE.2) THEN
         AH2O=DEXP(G(1)-U0RT(1))
         WRITE(6,3009) AH2O, XI
      END IF

3009     FORMAT('  AH2O= ',F12.8,'  IONIC STRENGTH= ',F15.8,' mol/kg')
         WRITE(6,3003)
         END IF
 10      CONTINUE

         SIG=0.0D0
         DO 40 I=1,NC0
         TAU=B(I)
         DO 50 K=1,NPHAS0
         IF(KTYP(K).GT.0) THEN
              JM=JMIN(K)
              JX=JMAX(K)
              DO 60 J=JM,JX
              TAU=TAU-X(J)*Q(J,I)
 60           CONTINUE
         END IF
 50      CONTINUE
         IF(DABS(TAU).GT.SIG) SIG=DABS(TAU)
 40      CONTINUE
         SIG=SIG*SCALE
         WRITE(6,5010) SIG
5010     FORMAT(//,' CONVERGENCE CRITERIA ',1P1D12.3)
         WRITE(6,5011) ITER,IFUN,IREFIN
5011     FORMAT('ITER= ',I5,'  IFUN= ',I5,'  IREFIN= ',I5)
         RETURN
3000     FORMAT(I2,1X,A24,1P3D14.5,0PF10.5)
3002     FORMAT('SPECIES',26X,'MOLES',10X,'LN(A)',8X,'CONC',8X,'CONC')
3003     FORMAT(/)
         END
```

APPENDIX G

DBLIST source code

```
      PROGRAM DBLIST
      CHARACTER*24 A1,A2
      CHARACTER*8 R1,R2


      WRITE(*,999)
 999  FORMAT(1X,'THE FOLLOWING IS A LIST OF THE CHEMICAL
     1 SPECIES FOUND IN THE DATABASE'/1X,'"THERMO.DAT" ALONG
     2 WITH THEIR RESPECTIVE I.D. NUMBERS AND THE
     3  TEMPERATURE',/1X,'RANGES OVER WHICH THE FREE ENERGY
     4 POLYNOMIAL IS VALID.  AN ASTERISK INDICATES'/1X,'THAT THE
     5  FREE ENERGY DATA WERE CALCULATED USING STANDARD
     6  STATE DATA FOUND',/1X, 'IN THE LITERATURE.  THE ABSENSE
     7  OF AN ASTERISK INDICATES THAT THE  FREE' /1X,'ENERGY
     8  DATA WERE CALCULATED FROM PUBLISHED SOLUBILITY
     9  DATA.  CONSULT THE'/1X,'USERS MANUAL FOR MORE
     0  INFORMATION.'///)

      OPEN(UNIT=5,FILE='THERMO.DAT',STATUS='OLD')
      REWIND(5)
      READ(5,1000)
1000  FORMAT(/////////)

    1 READ(5,1001) I1,A1,R1,I2,A2,R2
      IF(I1.EQ.900) STOP
      IF(I2.EQ.900) THEN
      WRITE(*,1002) I1,A1,R1
      STOP
      END IF

      WRITE(*,1002) I1,A1,R1,I2,A2,R2
      GO TO 1

1001  FORMAT(I4,A24 / 4X,A8)
1002  FORMAT(1X,I3,1X,A24,A8,5X,I3,1X,A24,A8)

      STOP
      END
```