Calculation of Chemical and Phase Equilibria

by

C.A. Meyer

Submitted to the University of Cape Town in fulfillment of the requirements for the degree of Master of Science in Engineering.

May 1996

Synopsis

The computation of chemical and phase equilibria is an essential aspect of chemical engineering design and development. Important applications range from flash calculations to distillation and pyrometallurgy. Despite the firm theoretical foundations on which the theory of chemical equilibrium is based there are two major difficulties that prevent the equilibrium state from being accurately determined. The first of these hindrances is the inaccuracy or total absence of pertinent thermodynamic data. The second is the complexity of the required calculation. It is the latter consideration which is the sole concern of this dissertation.

Research into the computation of the equilibrium state for systems containing a large number of species led to the development of algorithms based on the minimization of thermodynamic energy functions. In chemical engineering applications isobaric, isothermal conditions are the most common. Under these conditions the minimizer of the Gibbs energy function defines the equilibrium state. Developments in Gibbs energy minimization algorithms were driven by a need to make the algorithms more robust, faster, and capable of handling systems involving multiple phases and complicated thermodynamic models.

The general problem of minimizing a continuous function subject to constraints falls into the field of nonlinear programming. Many of the advances in Gibbs energy minimization algorithms have resulted from the application of general minimization algorithms to the chemical equilibrium problem, sometimes in a modified form to take into account its peculiarities. As there is a substantial literature on general nonlinear programming methods and Gibbs energy minimization methods, a fundamental understanding of the features specific to the chemical equilibrium problem is necessary if the pitfalls are to be avoided and the opportunities arising from the structural features are to be exploited.

The broad objectives of this dissertation are to identify methods which are suitable for solving Gibbs energy minimization problems and to point out potential difficulties involved in the solution of this problem. A review of the literature places the Gibbs energy minimization methods in context with mathematical programming techniques. The features of the Gibbs energy minimization problem which distinguish it from more general types of nonlinear programming problems are described and the use of problem structure to customize nonlinear programming techniques to suit the Gibbs energy minimization problem are examined.

One of the difficulties inherent in the Gibbs energy minimization problem is that the number of equilibrium phases and species is not known *a priori*. An approach to treating this problem, suggested by Paules and Floudas [57a], involves its formulation as a mixed integer nonlinear program. In this formulation, integer variables are introduced to explicitly

represent the presence or absence of a phase or species. A mixed integer nonlinear programming algorithm based on the Generalized Benders Decomposition was implemented and the methodology was assessed.

Arguably, the most important factor influencing the solution of a Gibbs energy minimization problem is whether the problem is convex or not. Convex problems are easy to solve as convexity guarantees that a local minimum is also the global minimum. When a problem is not convex there is no such guarantee and global optimization strategies are required to find the global minimum. There are two general global optimization approaches which have been applied to the Gibbs energy minimization problem; rigorous and heuristic.

Rigorous global optimization approaches take into account the structure of a problem's nonconvexity and provide a guarantee on the global optimality of the computed solution. A rigorous global optimization approach for NRTL based Gibbs energy minimization problems, which was developed by McDonald and Floudas [44], was implemented in FORTRAN and assessed on a set of nonconvex phase equilibrium test problems. A modification of the algorithm was developed to exploit regional convexity, thereby improving the efficiency of the algorithm without compromising its rigor. The performance of the modified algorithm was evaluated against the unmodified version.

Heuristic methods, unlike the rigorous ones, cannot guarantee the global optimality of the solutions they find. Nevertheless, they are simpler to implement and run in a small fraction of the time required by the rigorous approaches. The reliabilities of three heuristic methods, Gautam and Seider's method [23], Walraven and van Rompay's method [80], and a tangent plane method [1, 48], were tested on a set of ternary phase equilibrium problems. The heuristic methods' CPU times were compared against those of the rigorous approaches.

Acknowledgments

I would like to thank Dr. Chris Swartz for supervising this project, and MINTEK for their financial support. For their help, primarily in sorting out computer problems, I wish to acknowledge the members of the Process Systems Group: Trevor Hadley, Roderick Ross, Mark Seager and Julian Young. I am grateful to Trusha for helping me cross the t's and dot the i's.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The computation of chemical and phase equilibrium is an essential aspect of chemical engineering design and development. Important applications range from flash calculations to distillation and pyrometallurgy. Despite the firm theoretical foundations on which the theory of chemical equilibrium is based there are two major difficulties that prevent the equilibrium state from being accurately determined. The first of these hindrances is the inaccuracy in, or total absence of, pertinent thermodynamic data. The second is the complexity of the required calculation. It is the latter consideration which is the sole concern of this dissertation.

Since the emergence of the digital computer in the 1940's a large number of chemical equilibrium computation algorithms have been developed. Some of the earliest work on chemical equilibrium algorithms was carried out at NASA and was motivated by the problem of calculating the combustion products of rocket fuels. The need to compute the equilibrium state of systems containing a large number of species in a simple yet rigorous manner lead to the development of algorithms based on the minimization of thermodynamic energy functions. In chemical engineering applications isobaric, isothermal conditions are the most common. Under these conditions the minimizer of the Gibbs energy function defines the equilibrium state. Developments in Gibbs energy minimization algorithms were driven by a need to make the algorithms more robust, faster, and capable of handling systems involving multiple phases and complicated thermodynamic models.

The general problem of minimizing a continuous function subject to constraints falls into the field of nonlinear programming. Many of the advances in Gibbs energy minimization algorithms have resulted from application of the general minimization algorithms to the chemical equilibrium problem, sometimes in a modified form to take into account its peculiarities. As

a vast literature on both general nonlinear programming methods and Gibbs energy minimization methods has been generated, a fundamental understanding of the features specific to the chemical equilibrium problem is necessary if the pitfalls are to be avoided and the opportunities arising from the structural features are to be capitalized.

The broad objectives of this dissertation are to identify methods which are suitable for solving Gibbs energy minimization problems and to point out potential difficulties involved in the solution of this problem. More specifically the objectives of this thesis are:

- to describe the features of the Gibbs energy minimization problem; to show how these features influence the performance of the Gibbs energy minimization algorithms proposed in the literature;
- to assess the mixed integer nonlinear programming approach to Gibbs free energy minimization;
- to assess the following techniques developed to solve nonconvex Gibbs energy minimization problems:
  - the deterministic decomposition based global optimization algorithm developed by McDonald and Floudas [44]
  - heuristic "phase splitting" techniques.

This dissertation addresses these objectives in two parts. The first part, contained in chapters 2 to 5 is based on the literature about Gibbs energy minimization algorithms and constrained nonlinear programming. The second, contained in chapters 6 to 8, is an investigation into the novel methods which have been proposed to address the more difficult issues involved in Gibbs energy minimization computations.

The number of proposed Gibbs energy minimization procedures is vast; yet many of these algorithms are merely minor variations of one another. Consequently, in this dissertation each algorithm is not examined individually, instead the discussion is based primarily on generic algorithm types with particular focus being placed on individual algorithms which are of particular interest due to their good or sometimes misguided use of problem structure. The general features of the Gibbs energy minimization problem are discussed in chapter 2. Chapters 3,4 and 5 deal with nonlinear programming algorithms and the relationship between problem structure and solution method. The categorization of methods into "reduced space" ,"Lagrangian" and "barrier" methods follows the approach adopted by Luenberger [42], the difference between the techniques being primarily the way in which the constraints are handled. Chapter 3 is about "reduced space methods" which employ linear algebra techniques

to incorporate the constraints. "Lagrangian methods" which use the Lagrangian function to incorporate the constraints are discussed in chapter 4. Chapter 5 is a short chapter on "barrier methods" which introduces polynomial-time interior point methods. Strictly speaking, some of the concepts relating to linear equality constraints discussed in chapters 3 and 4 also apply to the methods in chapter 5; however, the issues dealt with chapter 5 are primarily the treatment of inequality constraints and more importantly the use of convexity and smoothness conditions in the development of efficient algorithms.

Two difficulties associated with the chemical equilibrium problem are ill-conditioning of the projected Hessian and an *a priori* lack of knowledge about the number of phases present at equilibrium. A new approach aimed at tackling these issues is a mixed integer nonlinear programming (MINLP) approach proposed by Paules and Floudas [57]. In this approach, integer variables are used to explicitly indicate the presence or absence of a chemical species or phase. In chapter 6 this approach is examined and results based on an implementation of the method are reported.

A severe impediment to chemical equilibrium computation, identified in chapter 2, is manifest when the Gibbs energy function is nonconvex. The implication of nonconvexity is that the Gibbs energy surface may have many local minima whereas the equilibrium composition corresponds to the global minimizer. As traditional nonlinear programming methods use local information about the Gibbs energy surface, they are unable to distinguish a local minimum from a global one. Two general methodologies have been proposed in the literature to resolve this problem. The first, based on heuristic rules, cannot guarantee that the solution obtained is the correct one. The second, based on a rigorous analysis of the problem can provide certain guarantees on the optimality of the solution at the expense of complexity.

In chapter 7 the rigorous decomposition based approach of McDonald and Floudas [44] is explained and the results from a FORTRAN implementation of the algorithm are discussed. A method which can be used to speed up the convergence rate of the global optimization algorithm is then developed. Efficiencies of the modified version of the global optimization algorithm are then compared with those of the unmodified version.

In chapter 8 three heuristic global optimization approaches are examined:

- Gautam and Seider's phase splitting method [22];
- Walraven and Van Rompay's phase splitting method [80];
- a tangent plane based method [1, 47].

The reliability of these methods is assessed on their ability to solve phase equilibrium problems where the Gibbs energy function is based on the NRTL model. CPU times of the heuristic methods are compared against those of the rigorous ones.

Conclusions on the findings of this dissertation are drawn in chapter 9 and recommendations for further research are made.

Chapter 2

The Phase and Chemical Equilibrium Problem

2.1 Criteria for Thermodynamic Equilibrium

The thermodynamic equilibrium conditions, originally derived by Gibbs towards the end of the 19th century, and presented in various forms in numerous thermodynamic textbooks, are outlined here. Intuitively, an equilibrium state is one that does not change with time, and when perturbed to a non-equilibrium state, given enough time, will return to the original equilibrium state. This qualitative idea can be applied to chemical systems with the aid of the two fundamental laws of thermodynamics to establish a quantifiable criterion for equilibrium. The first law of thermodynamics, the principle of energy conservation, for our purposes can stated be in the following form:

$$dU = dQ - PdV \tag{2.1}$$

Where U, P, and V are respectively, the internal energy, pressure and volume of the system, and Q is the heat entering the system. A classical presentation of the second law is [61]:

$$\begin{aligned} dS &= \frac{dQ}{T} \text{ for reversible processes in a closed system,} \\ dS &> \frac{dQ}{T} \text{ for irreversible processes in a closed system,} \end{aligned} \tag{2.2}$$

where S is the entropy of the system and T the absolute temperature. A quantity Q', referred to by Clausius as the "uncompensated heat" [61] , is defined below as:

$$\frac{dQ'}{T} \equiv dS - \frac{dQ}{T}. \tag{2.3}$$

Combining 2.1 and 2.3 to eliminate Q, the following is obtained:

$$dU + PdV = TdS - dQ'$$

Now, the entropy of a system can vary for two reasons only:

1. due to the transport of entropy to or from the surroundings through the system's boundary surface.
2. due to the creation of entropy inside the system by irreversible processes such as chemical reactions and diffusion.

The "uncompensated heat" is associated with the entropy that is generated inside the system and is of particular importance in defining a criterion for equilibrium.

Consider a system that is closed and is maintained at a constant temperature and pressure. For an irreversible change, because dQ' is always positive, the following relation applies:

$$dU - TdS + PdV \leq 0 \tag{2.4}$$

where the equality holds for equilibrium states only [66]. As the Gibbs free energy is defined by the equation:

$$G = U + PV - TS$$

and, because the temperature and pressure of the system are invariant, relation 2.4 can be expressed in terms of the Gibbs energy:

$$dG \leq 0$$

This means that if a system is not at equilibrium then the Gibbs energy of the system tends to decrease until the equilibrium is attained, at which point there is no further change in G. In other words the thermodynamic criterion for equilibrium in a closed isothermal, isobaric system is that the Gibbs free energy of the system is minimized. Analogous to the isothermal, isobaric case, similar thermodynamic equilibrium criteria apply to closed systems under different conditions. In particular:

Entropy is maximized for equilibrium at constant U, V;

Helmholtz free energy is minimized for equilibrium at constant T, V.

In chemical engineering applications the isobaric, isothermal case is most frequently encountered, and is therefore the focus of this study. Nevertheless, many of the issues associated with the Gibbs free energy minimization problem are common to chemical equilibrium computations in general.

2.2 Mathematical Formulation of the Chemical Equilibrium Problem

The chemical equilibrium problem can be posed as a constrained nonlinear program. Before presenting this formulation, some notation similar to that used by McDonald and Floudas [44] is introduced. The set of chemical species, where each species is defined by its molecular formula alone, is represented by the index set $C = \{i\}$. Similarly, the set of elements that constitute these species are referred to by $E = \{e\}$, and the set of phases is denoted $P = \{k\}$. The molar quantity of chemical species i in phase k is then denoted n_{ik}. One of the problems associated with the formulation is that the number and nature of the phases present in the equilibrium system is not known *a priori*. A rigorous formulation of the problem therefore includes all phases which could possibly manifest themselves at equilibrium. The chemical equilibrium problem can be stated as:

$$\begin{aligned} & \min_n G(n) \\ \text{subject to } An &= b \\ n_{ik} &\geq 0 \quad \forall\ i \in C,\ k \in P \end{aligned} \tag{2.5}$$

where n is a column vector with elements n_{ik}, and G is the Gibbs free energy function for the system. To render the objective function dimensionless, G is usually replaced by $\frac{G}{RT}$ where R is the gas constant and T is the absolute temperature of the system. The equation $An = b$ is a mass balance relation which is described in more detail below. The inequality constraints ensure that mole numbers are always positive quantities, an obvious requirement of physical systems.

2.2.1 Material Balances

In section 2.1 we noted the Gibbs energy minimization criterion for equilibrium applies to *closed*, isobaric, isothermal systems. A closed system is one where no matter is transferred across its boundaries. To describe the compositional states attainable by a closed system, the principle of material conservation therefore needs to be taken into account. This is done using the mass balance relations:

$$\sum_{k \in P} \sum_{i \in C} a_{eik} n_{ik} = b_e \quad \forall \quad e \in E$$

where a_{eik} represents the number of moles of element e in component i of phase k, and b_e denotes the total number of moles of element e in the closed system. In the general case where chemical reactions take place, E represents the set of atomic species in the system.

In a system where no reaction takes place at all, the equilibrium problem becomes a pure phase equilibrium problem, in which case each element e will represent a chemical species i. The equation $An = b$ in formulation 2.5 is simply the matrix representation of these mass balance constraints.

Example 1 *Consider a chemically reacting system, involving the chemical species* $C, O_2, CO,$ *and* CO_2. *Suppose the feed to the system consists of 1 mole C, 2 moles* O_2, *3 moles CO and 4 moles of* CO_2. *The number of moles of the O element will then be* $(1\times0)+(2\times2)+(3\times1)+(4\times 2) = 15$, *while the number of moles of the C element will be* $(1\times1)+(2\times0)+(3\times1)+(4\times1) = 8$. *We can then let* $b_O = 15$, *and* $b_C = 8$. *The mass balance equation for carbon in this system is:*

$$1n_C + 0n_{O_2} + 1n_{CO} + 1n_{CO_2} = 8,$$

while the balance equation for oxygen is:

$$0n_C + 2n_{O_2} + 1n_{CO} + 2n_{CO_2} = 15.$$

These equations can be expressed in matrix form as

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} n_C \\ n_{O_2} \\ n_{CO} \\ n_{CO_2} \end{bmatrix} = \begin{bmatrix} b_C \\ b_O \end{bmatrix}.$$

2.2.2 Alternative Formulations

Mathematical formulations of the phase and chemical equilibrium problem that are, under certain circumstances equivalent to formulation 2.5, are introduced in this section. These formulations, although mathematically similar can be seen to underlie solution methods that may differ greatly in terms of their computational requirements. Much of the discussion on chemical equilibrium algorithms in the literature has revolved around which formulation is the "best". This debate is arguably of little interest, as the computational performance of a method usually has more to do with the finer details of the algorithm than the broad underlying mathematical formulation. The underlying formulations nonetheless provide the foundations on which algorithms can be constructed and an examination of these formulations and their limitations is therefore of fundamental importance.

Nonstoichiometric Optimality Condition Formulation

This formulation is based on the Karush-Kuhn-Tucker necessary conditions for constrained smooth nonlinear programs. At a local minimum on the Gibbs energy surface the following conditions can be shown to apply:

$$\begin{aligned}
\nabla G(n) + \sum_{j \in E} \lambda_j \nabla g_j(n) + \sum_{k \in P} \sum_{i \in C^k} \rho_{ik} \nabla h_{ik}(n) &= 0 \\
\rho_{ik} h_{ik}(n) &= 0 \qquad \forall i \in C^k, \forall k \in P \\
g_j(n) &= 0 \qquad \forall j \in E \\
h_{ik}(n) &\geq 0 \qquad \forall i \in C^k, \forall k \in P \\
\rho_{ik} &\leq 0 \qquad \forall i \in C^k, \forall k \in P \\
\text{where } g(n) &= An - b, \\
\text{and } h_{ik}(n) &= n_{ik}
\end{aligned}$$

The variables λ and ρ are Lagrange multipliers for the mass balance equality constraints and the non-negativity constraints respectively. In some of the earlier work on equilibrium computations [81, 55] the inequality constraints are ignored, resulting in the same formulation as above without the ρ Lagrange multipliers and the complementary slackness condition $\rho_{ik} h_{ik}(n) = 0$. When the Gibbs free energy is convex these conditions are sufficient for global optimality [58]. Local minima, maxima and saddle points can, however, fulfill these conditions when the Gibbs free energy objective function is nonconvex. This formulation is therefore mathematically equivalent to formulation 2.5 only when the Gibbs energy function is convex.

Stoichiometric Formulations

The use of stoichiometric reaction equations in the representation of chemical equilibrium problems predates the nonstoichiometric formulation. In this formulation the linear mass balance relations are taken into account through the use of linear algebra methods. The following, being a fundamental formulation rather than one based on optimality conditions, is mathematically equivalent to 2.5:

$$\begin{aligned}
&\min G(p) \\
\text{subject to} \quad & n + Zp \geq 0.
\end{aligned}$$

Here p can be seen as being a vector of reaction extents and Z as a matrix of stoichiometric reaction coefficients.

Example 2 *Consider the system described in example 1. In this system the following chemical reactions may take place:*

$$2C + O_2 \rightleftharpoons 2CO \qquad \textit{(reaction 1)}$$

$$O_2 + 2CO \rightleftharpoons 2CO_2. \qquad \textit{(reaction 2)}$$

Denoting the feed as n^{feed}, and the extents of the reactions as, p_1 and p_2, the number of moles in the system is defined via the equations:

$$n = n^{feed} + Zp$$

$$\begin{bmatrix} n_C \\ n_{O_2} \\ n_{CO} \\ n_{CO_2} \end{bmatrix} = \begin{bmatrix} n_C^{feed} \\ n_{O_2}^{feed} \\ n_{CO}^{feed} \\ n_{CO_2}^{feed} \end{bmatrix} + \begin{bmatrix} -2 & 0 \\ -1 & -1 \\ 2 & -2 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}.$$

In this equation the columns 1 and 2 of matrix Z are defined by the stoichiometric coefficients of reactions 1 and 2 respectively. Using this relation the search vector n can be replaced be the search vector p.

Methods based on this type of approach are discussed in chapter 3.

Geometric Programming Formulation

Geometric programming formulations for the equilibrium problems are applicable to problems that include ideal phases only. There are two types of geometric programming formulation, the primal and the dual. The standard form of the dual geometric program is [58]:

$$\begin{aligned} \max_{x,y} \left\{ \prod_{i \in I} \left(\frac{c_i}{x_i} \right)^{x_i} \right\} \prod_{j \in J} y_j^{y_j} & \\ \text{subject to} \quad \sum_{i \in I^0} x_i &= 1 \\ Ax &= 0 \\ x &\geq 0 \\ \text{where} \quad y_j &= \sum_{i \in I^j} x_i \quad \forall \quad j \in J \\ \cup_{j \in J} I^j \cup I^0 = I, \quad I^j \cap I^{k \neq j} &= \emptyset \end{aligned}$$

Passy and Wilde [56] showed that the chemical equilibrium problem can be manipulated to conform to the dual geometric program [63]:

$$\max_{n, n_t} f(n, n_t) = \exp\left(-\frac{G}{RT}\right) = \frac{c_0}{n_0} \prod_{i \in C} \prod_{k \in P} \left(\frac{c_{ik}}{n_{ik}} \right)^{n_{ik}} \prod_{k \in P} n_{tk}^{n_{tk}}$$

$$
\begin{aligned}
\text{subject to} \quad n_0 &= 1 \\
An - bn_0 &= 0 \\
n &\geq 0 \\
n_{tk} &= \sum_{i \in C} n_{ik} \\
\text{where } c_i &= \exp\left(-\frac{\mu_i^0}{RT} - \ln P\right) \qquad \forall \quad i \in C \\
c_0 &= 1.
\end{aligned}
$$

The variable n_0 in this formulation merely serves as a dummy species to fulfill a technical requirement of geometric programs. This formulation can be seen to yield a more familiar form of the Gibbs free energy function when the natural logarithm of the geometric programming objective function is taken:

$$
\begin{aligned}
& \ln\left(\prod_{i \in C}\prod_{k \in P}\left(\frac{c_{ik}}{n_{ik}}\right)^{n_{ik}}\prod_{k \in P} n_{tk}^{n_{tk}}\right) \\
= & \sum_{k \in P}\sum_{i \in C} n_i^k\left(\ln(c_{ik}) - \ln(n_{ik})\right) + \sum_{k \in P} n_{tk}\ln(n_{tk}) \\
= & \frac{-G}{RT}
\end{aligned}
$$

Geometric programming theory has yielded results on the optimality conditions for geometric programs, and duality concepts that are useful for problem analysis and termination criteria [60]. Numerical procedures used for the solution of geometric programs are based mainly on conventional nonlinear programming techiques which have been modified to exploit some of the structure inherent in geometric programs. According to Sarma et al. [68], convexity is the main structural feature of geometric programs that can be exploited effectively in computational methods. Recent developments in interior point methods for convex programming have resulted in efficient methods which exploit the convexity of geometric programs [53].

feed to each tray is not known *a priori,* but has to be determined from the equilibrium computations defining the products of the tray above or below the tray in question. The distillation column problem can be formalized and generalized using Garcia and Zangwill's [21] "equilibrium programming" concept .

Equilibrium programming considers the case of a composite system, composed of a number of subsystems; similar to a distillation column, composed of a number of trays. The variables x in the equilibrium programming problem are partitioned into disjoint subsets, so that each subsystem $[i]$ has its corresponding subset of variables $x_{[i]}$. In the distillation column case, the variables include the flowrates of the chemical species leaving each tray as well as the temperature of these trays. In equilibrium programming there is a minimization subproblem associated with each subsystem:

$$\begin{aligned} \min_{x_{[i]}} & f_{[i]}\left(x_{[i]}, x_{[i']}\right) \\ \text{subject to } & g_{[i]}(x_{[i]}, x_{[i']}) \geq 0 \\ & h_{[i]}(x_{[i]}, x_{[i']}) = 0 \end{aligned}$$

where $[i']$ denotes the union of subsets in the systems that are not subset $[i]$. The objective function for distillation column tray $[i]$ would be the Gibbs free energy function, and the equality constraints would include mass and energy balance constraints. Garcia and Zangwill define the equilibrium program as the process in which all subproblems are being continuously resolved, given the current information on the variables of the other subproblems. At a solution to the equilibrium problem, all subproblems are at a simultaneous minimum, and it is such a point that defines the steady state of a distillation column.

The usual way of carrying out equilibrium computations is therefore, not to minimize the Gibbs energy of individual trays, but to solve a set of nonlinear and linear equations which represent the combined Kuhn-Tucker necessary conditions for all trays. Although this scheme is usually satisfactory, there are cases where this Kuhn-Tucker system approach is inadequate without modifications. There are two reasons for this:

1. the equilibria in the individual trays are determined from criteria that are not always sufficient optimality conditions;
2. the number of phases in each tray is not known *a priori.*

As a result of 1, the equilibrium determined for any tray may represent a local minimum, maximum, or stationary point on the Gibbs energy surface for that tray. Furthermore, even if the point found corresponds to a local minimum, this minimum might not be the global

minimum required to predict the stage equilibrium. Consideration 2 results in the Kuhn-Tucker system being potentially incompatible with the true steady state behaviour of the column.

Clearly, neither the Kuhn-Tucker system approach, nor a simplistic tray-by-tray equilibrium computation strategy is entirely satisfactory for this problem. The role of the chemical equilibrium system within the global process is therefore a pivotal consideration in the development of a successful computation technique.

2.3 The Gibbs Free Energy Function

In formulation 2.5 it is assumed that an explicit Gibbs energy function for the system is known. The way in which the Gibbs energy function is modelled is of interest in a study of chemical equilibrium computations because the structure of the Gibbs energy objective function is the main cause of the difficulties associated with the equilibrium problem. In this section the Gibbs energy function is described, starting with some observations which apply in general:

1. Because the interaction between phases does not contribute significantly to the Gibbs energy of the system, it is assumed that the Gibbs energy of a multiphase system is equal to the sum of the Gibbs energies of the individual phases:

$$G(n) = \sum_{k \in P} G(n_k).$$

2. The Gibbs energy G of a phase can be written in the following form:

$$G = \sum_{i \in C} n_i \mu_i$$
$$\text{where } \mu_i = \frac{\partial G}{\partial n_i}.$$

To fully define G we require an expression for the chemical potential μ. As the chemical potential cannot be easily deduced by purely theoretical means, it is usually based on empirical data. The strategy used in thermodynamics to model the large variety of different situations that may occur, is to first model the chemical potential of an idealized system which displays features that are common to all systems. This approximation to the real system is then corrected by superimposing a model of the nonideal behaviour over the ideal model. Three ideal models are introduced in this chapter:

1. the ideal gas model
2. the ideal gas mixture
3. the ideal mixture

In the following discussion of ideal models, the effect of pressure on chemical potential is explained using the ideal gas model. Then, using the ideal gas mixture concept, the mixing of pure ideal gases is related to the changes in pressure for pure ideal gases. Finally, the ideal gas mixture notion is generalized to that of the ideal mixture.

2.3.1 Pressure Effects for the Ideal Gas Model

An ideal gas is one that behaves according to the ideal gas law:

$$Pv = RT.$$

where v denotes the molar volume of the gas. Given the chemical potential of the gas at one pressure we now look at how the potential at a different pressure can be obtained. The following relation describes changes in the Gibbs energy with respect to changes in pressure, temperature and mole number [66]:

$$dG = VdP - SdT + \sum_{i \in C} \mu_i dn_i.$$

From this equation, the following result is obtained [17]:

$$\frac{\partial \mu_i}{\partial P} = \frac{\partial V}{\partial n_i}.$$

Using this identity and the fact that the partial molar volume, $v_i = \frac{\partial V}{\partial n_i}$, and molar volume, $\frac{V}{n}$, are the same in a pure gas we can relate the potential of a pure gas at a given temperature and pressure P_1 to the potential at another pressure P_2:

$$\begin{aligned} \mu_i^{IG}(T, P_2) - \mu_i^{IG}(T, P_1) &= \int_{P_1}^{P_2} v_i dP \\ &= \int_{P_1}^{P_2} \frac{RT}{P} dP \\ &= RT \ln\left(\frac{P_2}{P_1}\right) \end{aligned}$$

2.3.2 The Ideal Gas Mixture

Denbigh [17] defines an ideal gas mixture as one that:

1. obeys the ideal gas equation for a mixture:

$$Pv_{\text{mix}} = \sum_{i \in C} x_i RT$$

   where x_i denotes the mole fraction of component i and v_{mix} is the total volume divided by the molar extent of the mixture;

2. is at equilibrium with another such mixture separated by a semipermeable membrane when the partial pressure for each component that permeates the membrane is equal in both mixtures;

3. has no heat of mixing.

Lets look at the difference in the chemical potential between a pure ideal gas "A" at pressure P_1 and the same gas in a mixture of ideal gases at the same pressure.

If the mole fraction of A in the mixture is x_A then the gas in the mixture exerts a pressure of $x_A P_1$. The change in the chemical potential between the pure gas A, and A in the mixture is:

$$\begin{aligned} \mu_A^{IG}(T, x_A P_1) - \mu_A^{IG}(T, P_1) &= \int_{P_1}^{x_A P_1} \frac{RT}{P} dP \\ &= RT \ln \frac{x_A P_1}{P_1} \\ &= RT \ln x_A \end{aligned}$$

The logarithmic function features in all Gibbs energy functions for mixed phases, and is the cause of many computational difficulties.

2.3.3 The Ideal Mixture

The ideal mixture model is useful for describing mixtures of species that are not ideal gas species but nevertheless display some of the behaviour that is characteristic of an ideal gas mixture. An ideal mixture can be defined as a mixture in which the partial molar volume and enthalpy of a substance in the mixture is the same as that of the pure compound [66]:

$$\frac{\partial H(T, P, n)^{IM}}{\partial n_i} = H_i(T, P)$$

$$\frac{\partial V(T,P,n)^{IM}}{\partial n_i} = V_i(T,P)$$

where H_i, and V_i are respectively the molar enthalpy and volume of substance i, and superscript IM refers to the thermodynamic quantity in the ideal mixture. Like the ideal gas mixture, the chemical potential of the pure species is related to that of the species in the ideal mixture by the relation:

$$\mu_i^{IM}(T,P,x) - \mu_i(T,P) = RT \ln x_i$$

2.3.4 Deviations from Ideal Models

The pressure effects on the chemical potential of real pure substances can be treated in a way analogous to the ideal gas case through the introduction of a new thermodynamic quantity called fugacity. In the ideal gas such pressure effects are predicted by the relation:

$$\mu_i^{IG}(T,P_2) - \mu_i^{IG}(T,P_1) = RT \ln\left(\frac{P_2}{P_1}\right).$$

The fugacity function satisfies the relation:

$$\mu(T,P_2) - \mu(T,P_1) = RT \ln\left(\frac{f(T,P_2)}{f(T,P_1)}\right).$$

Intuitively the fugacity appears to be the pressure adjusted for lack of ideality, but is in fact defined as [66]

$$f(T,P) = P \exp\left(\frac{G(T,P) - G^{IG}(T,P)}{RT}\right)$$

where G and G^{IG} are the molar Gibbs energies of the pure real and pure ideal substances respectively. At sufficiently low pressures, the behaviour of a real gas closely approximates that of the ideal gas model. Under these conditions the fugacity and the pressure are nearly equal. If P_0 is a pressure low enough for the fugacity to equal the pressure then the chemical potential of a species at a higher pressure can be determined by:

$$\begin{aligned}\mu(T,P) &= \mu^{IG}(T,P_0) + RT \ln \frac{f(T,P)}{P_0} \\ &= \mu^{IG}(T,P_0) + RT \ln \frac{\phi(T,P)P}{P_0}\end{aligned}$$

where ϕ is called the fugacity coefficient. The fugacity of a species i in a *mixture* is defined as follows:

$$\tilde{f}_i(T,P,x) = x_i P \exp\left(\frac{\mu_i(T,P,x) - \mu_i^{IGM}(T,P,x)}{RT}\right)$$

This definition allows the chemical potential of a species in a mixture to be treated in a way analogous to the ideal gas case:

$$\mu_i(T,P,x) - \mu_i^{IG}(T,P_0) = RT \ln \tilde{f}_i(T,P,x).$$

2.3.5 Estimation of the Fugacity in Mixtures

Two common methods of estimating the fugacity of species in mixtures are:

1. equations of state
2. activity coefficient models.

Equations of state, which relate pressure to temperature and molar volume for pure species, with the aid of mixing rules and interaction parameters can be applied to estimate fugacity in mixtures. When the nonideality in the liquid phase is not too great, the equation of state can be used to represent the behaviour of both liquid and vapour phases. Liquid mixtures that are not describable by equations of state are often modelled using activity coefficients $\gamma_i(T, P, x_i)$ which are defined by the equation

$$\tilde{f}_i(T, P, x_i) = x_i \gamma_i(T, P, x_i) f_i.$$

or, alternatively

$$RT \ln \gamma_i(T, P, x_i) = \mu_i - \mu_i^{IM}.$$

The range of application of equations of state and activity coefficient models is summarized by Sandler [66]. Modelling the Gibbs energy functions with equations of state or activity coefficients can sometimes destroy the convexity of the Gibbs energy surface.

2.4 Characteristics of the Chemical Equilibrium Problem

As noted by Wright [83] most optimization algorithms are based on a frequently unstated, particular model of the underlying minimization problem. A generic optimization algorithm applied to a particular chemical equilibrium problem may be able to solve the problem, but not necessarily in the most efficient way, depending on how closely the algorithm's underlying model matches the problem itself. Consequently, there are two reasons for examining the characteristics of the chemical equilibrium problem. Firstly, these features can be used to assess how appropriate an available algorithm is for the solution of chemical equilibrium problems, and secondly they may reveal areas that require special consideration in algorithm development.

2.4.1 Unbounded Gradients

Consider the Gibbs free energy function for a single multicomponent phase:

$$\frac{G}{RT} = \frac{1}{RT}\sum_{i\in C} n_i\mu_i.$$

In all multicomponent phases the chemical potential of species i, μ_i, contains the term :

$$\begin{aligned} RT\ln(x_i) &= RT\ln n_i - RT\ln n_t, \\ \text{where} \quad n_t &= \sum_{i\in C} n_i. \end{aligned}$$

The $n_i \ln n_i$ and -$n_i \ln n_t$ terms in the Gibbs energy objective function are responsible for many of the difficulties inherent in chemical equilibrium calculations. As a species disappears from a non-vanishing phase the following occurs:

$$\lim_{n_i\to 0} \ln n_i \to -\infty.$$

The objective function term $n_i \ln n_i$, however, is well defined at $n_i = 0$, and can be evaluated using l'Hôpital's rule

$$\lim_{n_i\to 0} n_i \ln n_i = \lim_{n_i\to 0} \frac{\frac{d\ln n_i}{dn_i}}{\frac{d\frac{1}{n_i}}{dn_i}} = 0.$$

The partial derivative:

$$\frac{\partial \sum_{i\in C}(n_i \ln n_i - n_i \ln n_t)}{\partial n_i} = \ln n_i - \ln n_t,$$

on the other hand, is unbounded as n_i tends to zero for nonvanishing n_t. Consequently it is theoretically impossible for a species to vanish from a particular phase without the disappearance of the entire phase. In practice, due to finite digit representations of real numbers, the equilibrium quantity of the minor species may not be representable or of significance. The pathological behaviour in the objective function gradients demands special consideration in the removal or introduction of species and phases during an equilibrium computation. The earlier computational methods such as the RAND algorithm [81] dealt with the problem of vanishing species in a fairly *ad hoc* manner, removing a vanishing species from the computation to prevent the ensuing numerical difficulties. More recent approaches such as that of Castillo and Grossmann [12] have augmented the linear mass balance constraints with inequality constraints that ensure that no variable drops below a small positive predefined value. How small this lower bound should be depends on the accuracy required in the calculation as well as the floating point precision used. Typically the lower bound is not less than 10^{-10} .

2.4.2 Undefined Objective in the Infeasible Region

The objective function is undefined for negative mole values, due to the presence of logarithms in the function. Hence, any method chosen to minimize the Gibbs energy function must ensure that feasibility with respect to the inequality constraints must be maintained. This precludes methods such as the gradient projection method which involves computing an unconstrained step, and a projection of the result back into the feasible region. Nevertheless, it may be possible to use the gradient projection method if either the method is suitably modified or the equilibrium problem is transformed. Storey and van Zeggeren [75], for example, transformed the problem by using the search variables $y_i = \ln n_i$, which removes the difficulty with infeasibility, but replaces the linear equality constraints by nonlinear ones. A modified gradient projection algorithm, capable of handling nonlinear constraints, was used by Storey and van Zeggeren to solve the equilibrium problem.

2.4.3 Use of Objective Function Gradients

The Gibbs energy function for a single phase system can be represented in the form:

$$G = \sum_{i \in C} n_i \mu_i$$

where $\mu_i = \frac{\partial G}{\partial n_i}$ is referred to as the chemical potential. In this representation gradients must be computed before the Gibbs energy function itself can be evaluated. Consequently, analytical gradients are readily available and there is no additional cost in computing the gradients in addition to the Gibbs energy. This is useful as the convergence rate of a minimization algorithm is usually enhanced by the use of derivative information. Consider a line search procedure, for example, where an approximate minimum to the objective is sought along this line. A number of line search procedures such as the Golden Section search make use of function values alone and converge linearly. The cubic fit method ,on the other hand, makes use of first derivatives and has an order of convergence of 2 [42]. Although the above representation of the Gibbs energy function is standard, it is not necessarily the simplest form. McDonald and Floudas [44] have shown that the objective function described using the NRTL and UNIQUAC equations can be significantly simplified, at the cost of losing free derivative information.

2.4.4 Hessian Structure

The Hessian matrix of a twice differentiable function $f(x_{1,\ldots,}x_n)$,

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix},$$

provides information about the curvature of the function at a point, and may be used in minimization algorithms to generate a local quadratic model of the function. The Hessian of the Gibbs energy function has some special features, in addition to the symmetry inherent in any Hessian matrix. From a thermodynamic consistency relation known as the Gibbs-Duhem equation:

$$\sum_{i \in C} n_i \frac{\partial^2 G}{\partial n_j \partial n_i} = 0,$$

it is clear that the vector n lies in the null-space of the Hessian matrix and that the Hessian is always singular. Quasi-Newton updating procedures which approximate the Hessian matrix using first derivative information have been tailored to suit the Gibbs energy function so that the approximation of the Hessian has the same structure as the true Hessian [64, 41].

2.4.5 Ill-Conditioning of the Projected Hessian

The condition number κ of a matrix B is $\|B\|\|B^{-1}\|$, and using the spectral norm of B the condition number is $\frac{\lambda_{\max}}{\lambda_{\min}}$ where $\lambda_{\max}$ and $\lambda_{\min}$ are respectively the largest and smallest eigenvalues of B. As discussed by Luenberger [42], many classes of minimization procedure converge at a rate closely associated with the condition number of the projected Hessian matrix; the rate being slow for large condition numbers. It is well known that the chemical equilibrium problem is notoriously ill-conditioned [63], in particular when there is a species that is present in very small quantities.

A number of approaches have been suggested to limit the ill-conditioning effects. Murtagh and Saunders [50], using their general purpose MINOS optimization code, employed a technique whereby the lower bound on the mole variables is set initially at a relatively large value, and decreased in steps during the course of the minimization as the solution is approached. In this way ill-conditioning is alleviated at the initial stages of the computation, but progressively deteriorates as the solution is refined. They found that the number of iterations required to reach the solution with this technique was significantly less than those required

with a static lower bound. Positive lower bounds on mole numbers used to deal with unbounded gradients can therefore fulfill a dual purpose in the treatment of ill-conditioning.

Trangenstein [77], applying Gibbs energy minimization techniques for two-phase equilibrium computations, observed that the entries in the Hessian are inversely proportional to the total number of moles in their phase. As a result, when one of the phases becomes small or when the two phases are given nearly identical compositions, the projected Hessian becomes nearly singular. Consequently, one can run into the following difficulties when using reduced space methods for the phase equilibrium problem:

1. Cancellation of one of the pivots in the factorization of the projected Hessian as a result of the near singularity.
2. Failure, due to rounding errors, to correctly determine whether the projected Hessian is positive definite at a proposed minimum point, and consequently an inability to determine whether the iterates are converging to a local minimum or a saddle point.
3. Inaccuracy in the solution of linear systems involving the projected Hessian due to factorization errors, resulting in a failure to make progress near a minimum.

Knowing that the eigenvector corresponding to the smallest eigenvalue is the vector of mole fractions in the smaller phase, Trangenstein dealt with these difficulties by isolating this vector through the use of the Householder transformation. In this way the projected Hessian could be accurately evaluated, and the Cholesky factorization could be applied in a numerically stable fashion. Although Trangenstein was considering non-reacting systems, the same observations apply to reacting systems, and the cause of the difficulties is likely to manifest itself in most minimization methods.

2.4.6 Convexity

A function is convex over a set X if it satisfies the following statement:

$$\alpha f(x_1) + (1-\alpha) f(x_2) \geq f(\alpha x_1 + (1-\alpha) x_2) \quad \forall \quad x_1, x_2 \in X \text{, and } 0 \leq \alpha \leq 1$$

Convexity is an important consideration for two reasons:

1. a local minimum of a convex minimization problem is a global minimum.
2. simple local minimization procedures can be used on convex problems.

Gibbs energy functions based on ideal models are always convex, as is the Wilson equation [45]. Many nonideal activity coefficient models, including the NRTL, UNIQUAC, UNIFAC and ASOG equations are, however, nonconvex [44, 46, 45] . Chemical equilibrium computations in such instances involve obtaining the global minimum of a Gibbs free energy function that may have multiple local minima. This problem is discussed extensively in chapter 7. Even finding a local minimum of a function is exacerbated by the presence of nonconvexity. Newton's method, for example, when applied to a nonconvex problem does not always generate a descent direction and may therefore experience convergence difficulties if used in an unmodified form. Great theoretical advances in convex programming algorithms, in the form of interior point algorithms, have produced highly efficient methods for solving this type of program[53]. These methods cannot, however, be blindly applied to any convex program as they involve functions which are based on the analytic properties of the problem's objective function and constraints. The application of interior point methods to convex Gibbs energy minimization problems is discussed in chapter 5.

2.4.7 Degree of Nonlinearity of the Gibbs Energy Function

Here we consider the Gibbs energy function written in the form

$$G = \sum_{i \in C} \left(n_i \mu_i^0 + n_i \mu_i(n) \right),$$

where μ_i^0 are constants. Observe that there is a linear portion of the function resulting from the terms $n_i \mu_i^0$, and a nonlinear portion resulting from the terms $n_i \mu_i(n)$. Enthalpy effects are described primarily in the μ_i^0 constants whereas entropy and interaction effects manifest themselves in the $\mu_i(n)$ functions. Because the system temperature influences the enthalpic contribution to the Gibbs energy function far more than any other contribution, the nature of the Gibbs energy function varies with temperature. At the high temperatures encountered in smelting processes for example, the linear terms dominate.

2.4.8 Problem Size

Chemical equilibrium problems typically involve less than 100 variables and by today's optimization standards are considered small. Most nonlinear programming algorithms require the solution of linear systems on each iteration. In large scale problems the amount of computational effort involved in solving these linear systems is significant enough to justify methods which take advantage of the sparsity of the linear systems [83]. Although there may be a significant amount of sparsity in linear systems arising in large chemical equilib-

rium problems, the size of these problems is too small for the savings obtained from sparse factorization methods to counterbalance the overheads needed for sparse computations.

2.4.9 Partial Separability of Multiphase Systems

A minimization problem is said to be partially separable if the search variables can be separated into different groups in such a way that the objective and constraint functions can be expressed as sums of functions of these groups [42]:

$$\begin{aligned} & \min_x \sum_{i=1}^{q} f_i(x_{[i]}) \\ \text{such that} \quad & \sum_{i=1}^{q} h_i(x_{[i]}) = 0 \\ & \sum_{i=1}^{q} g_i(x_{[i]}) \leq 0 \end{aligned}$$

In this formulation the vector x has been partitioned into q disjoint groups $x = (x_{[1]}, x_{[2]}, ..., x_{[q]})$. Chemical equilibrium problems with several phases can be formulated in this way by partitioning the variables according to their phase designation. Partial separability may be used to decompose a problem into subproblems which may be solved more efficiently than the original problem [25, 42].

Chapter 3

Reduced Space Methods

The chemical equilibrium problem falls into the class of nonlinear programs of the form:

$$\begin{aligned} \min_{x \in \Re^n} & f(x) \\ \text{subject to} \quad Ax &= b \\ Cx &\geq d \\ \text{where } A &\in \Re^{m_1} \times \Re^n \\ C &\in \Re^{m_2} \times \Re^n \\ b &\in \Re^{m_1},\ d \in \Re^{m_2}, \end{aligned} \tag{3.1}$$

This problem's equality and inequality constraints are all linear. Reduced space methods, use mole numbers as search variables, and take constraints into account using results from linear algebra. One of the key concepts underlying reduced space methods is the notion of an affine manifold:

$$\begin{aligned} \{x &: x = x_0 + Ay\} \\ \text{where } x, x_0 &\in \Re^n \\ A &\in \Re^n \times \Re^m \\ y &\in \Re^m, \end{aligned}$$

which is simply a linear space plus a vector. The feasible region defined by the equality constraints in problem 3.1 is an affine manifold. This affine manifold is not the only one of interest in the minimization problem, as inequality constraints too play a role in defining the feasible region. Let the ith row of C be denoted as $C_{i\bullet}$. An inequality constraint $C_{i\bullet}x \geq d_i$, is said to be *active* when this constraint could be replaced by an equality constraint $C_{i\bullet}x = d_i$ without altering the solution to the optimization problem. The solution to the optimization

problem therefore lies on the manifold defined by the set of equality constraints and active inequality constraints.

Reduced space methods for linearly constrained optimization problems involve two conceptual parts:

1. finding the active manifold;
2. minimizing the objective function over the active manifold.

Finding the active manifold is achieved by a process in which a subset of the inequality constraints are included in a *working set* I. These constraints together with the equality constraints define a working manifold which is a hypothetical *active* manifold:

$$x \in \left\{ \begin{array}{c} x : Ax = b; \\ C_{\bullet i}x = d_i; \\ \text{for all } i \in I \end{array} \right\}.$$

Using a strategy based on objective function slope and curvature information, inequality constraints are added or removed from the working set during the course of the optimization. Minimization of the objective function over a working manifold is achieved using a combination of unconstrained minimization techniques and algebraic transformations. Neither of these factors takes precedence over the other. On the one hand it is desirable to find the active set of constraints at the earliest possible stage in the iterative process. On the other the decision criteria for adding or removing a constraint from the working set become more reliable as the minimum over the current working set is approached.

3.1 Effect of Linear Equality Constraints on the Search Direction

In this section, the algebraic treatment of the equality constraints defining a working manifold is described. Here we consider the minimization problem constrained only by linear *equality* constraints:

$$\begin{array}{rcl} \min\limits_{x \in \Re^n} f(x) & & \\ \text{subject to } Ax & = & b \end{array}$$

where $A \in \Re^{m \times n}$ and $b \in \Re^m$. This problem contains no inequality constraints therefore the active manifold is defined *a priori*. Assuming there is a feasible vector x_0, any other x

where

$$Ax = b$$

can be found by finding a vector p which satisfies

$$Ap = 0, \tag{3.2}$$

for then

$$A(x_0 + p) = Ax_0 + Ap = b$$

To simplify the discussion, A is assumed to have rows that are all linearly independent. This being the case, the rows of A are a basis for the range-space $R(A^T)$, a linear subspace of dimension m. Orthogonal to this subspace is the $n-m$ dimensional null-space of A, $N(A)$. Any vector $y \in \Re^n$ can be expressed as a linear combination of the vectors that form a basis for the range-space together with those that form a basis for the null-space:

$$p = Yp_y + Zp_z$$

where the columns of Y form a basis for $R(A^T)$, the columns of Z form a basis for $N(A)$, $p_y \in \Re^m$ and $p_z \in \Re^{n-m}$ [29]. Finding a vector p which satisfies 3.2 can be done by premultiplying any $p_z \in \Re^{n-m}$ by Z. Linear constraints therefore have the effect of reducing the dimension of the search variables from n to $n-m$, and in this sense simplify the minimization problem at hand. Note that because the columns of Z form a *basis* for the null-space of A, any p which satisfies equation 3.2 can be expressed as $p = Zp_z$.

3.2 Physical Interpretation of the Null-Space

The physical interpretation of the null-space in the chemical equilibrium problem led to the earliest implementations of "reduced space" methods. This interpretation is illustrated by an example:

Example 3 *Consider a chemically reacting system, described by Balzhizer et al. [2], involving the chemical species* C, O_2, CO, *and* CO_2.

The matrix representation of the linear constraints for this system is

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} n_C \\ n_{O_2} \\ n_{CO} \\ n_{CO_2} \end{bmatrix} = \begin{bmatrix} b_C \\ b_O \end{bmatrix}$$

Observe that the vectors $(-2,-1,2,0)$, $(0,-1,-2,2)$, *and* $(-1,0,2,-1)$ *are in the null-space of* A. *These vectors can be seen to respectively represent the stoichiometric coefficients of the following chemical reactions:*

$$2C + O_2 \rightleftharpoons 2CO \tag{3.3}$$

$$O_2 + 2CO \rightleftharpoons 2CO_2 \tag{3.4}$$

$$C + CO_2 \rightleftharpoons 2CO \tag{3.5}$$

The dimension of the null-space is $n - m = 2$, *therefore only two of these chemical reactions are linearly independent. Choosing the first two vectors to represent* $N(A)$, *a feasible direction* p *can therefore be found by premultiplying an arbitrary* p_z *by the matrix representation of two chemical reactions:*

$$\begin{bmatrix} -2 & 0 \\ -1 & -1 \\ 2 & -2 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} p_{z,1} \\ p_{z,2} \end{bmatrix} = \begin{bmatrix} p_C \\ p_{O_2} \\ p_{CO} \\ p_{CO_2} \end{bmatrix}$$

where $p_{z,1}$ *and* $p_{z,2}$ *can be viewed as the reaction extents of chemical reactions 3.3 and 3.4.*

3.3 Null-Space Representation

With knowledge of the independent reactions in a system, it is possible, using stoichiometric coefficients, to define a basis for the null-space of A. This approach may, however, be impractical for a system involving many species and the resulting basis may be unsuitable for computational purposes. As Z can be formed from any linearly independent set of vectors in the null-space of A, there is a great deal of flexibility in how Z is defined. Two techniques are discussed in this section, the variable reduction technique and the QR factorization.

3.3.1 The Variable Reduction Technique

Early investigations into the chemical equilibrium problem by Brinkley [8, 7, 9] introduced the idea that the set of chemical species can be segregated into sets of "component" and "non-component" species where the formula vectors of the non-component species can be written as a linear combination of the component species. The component species which constitute a set of species whose formula vectors form a basis for the range space of A are not unique; however, once the set of independent component species has been defined the null-space is also completely defined.

A more detailed treatment of this technique is based on the partitioning of the constraint matrix [29].

$$A = [V \quad U]$$

where V is an $m \times m$ nonsingular matrix. By partitioning the search variables in a consistent manner as $x = [x_V \quad x_U]$, the constraints on the linearly constrained problem can be written as

$$[V \quad U] \begin{bmatrix} x_V \\ x_U \end{bmatrix} = b \ .$$

Because V is not singular, x_V can be written explicitly in terms of x_U :

$$x_V = V^{-1}(b - Ux_U).$$

The variables referred to by Brinkley as "component" variables are elements of the x_U vector and the "non-component" species are elements of x_V. A matrix Z that has columns which are orthogonal to the rows of A is readily verified to be:

$$Z = \begin{bmatrix} -V^{-1}U \\ I_{n-m} \end{bmatrix}. \tag{3.6}$$

One can check that this matrix does indeed consist of vectors which are in the null-space of A by performing the matrix multiplication:

$$\begin{aligned} AZ &= \begin{bmatrix} V & U \end{bmatrix} \begin{bmatrix} -V^{-1}U \\ I_{n-m} \end{bmatrix} \\ &= -VV^{-1}U + U \\ &= 0. \end{aligned}$$

Smith and Missen [73] describe a method of forming the matrix Z for a given A matrix. The

first step is to reduce A by Gauss reduction to the form:

$$\begin{bmatrix} I_m & B \\ 0 & 0 \end{bmatrix}.$$

Note that the B matrix obtained from the Gauss reduction is simply $V^{-1}U$ in equation 3.6. Z can then be formed by appending the $(n-m) \times (n-m)$ identity matrix to -B :

$$Z = \begin{bmatrix} -B \\ I_{n-m} \end{bmatrix}.$$

Gill et al. [29] note that when this null-space representation is used in a reduced space method, it is not always necessary to form Z explicitly because the role that Z plays in an algorithm is affected best by means involving entities other than Z itself.

3.3.2 The QR Factorization

The QR factorization, a well known method of determining an orthonormal basis for a set of vectors, applied to the linearly constrained problem can determine an orthogonal basis for the null space of A. QR factorization of the matrix A^T yields the following relation

$$QA^T = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where Q is an orthonormal matrix and R is an upper triangular matrix. Notice that because Q is an orthonormal matrix $Q^TQ = I$ and

$$A^T = Q^T \begin{bmatrix} R \\ 0 \end{bmatrix} = [Q_1 \quad Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R$$

Orthonormal bases for the range-space of A^T and null-space of A can therefore be obtained from the columns of Q_1 and Q_2 respectively [31]. Hence, using the QR decomposition Z is represented by Q_2. The main advantage of using the QR decomposition is that this choice of Z does not cause the conditioning of the linearly constrained problem to deteriorate. Ill-conditioning impacts on quasi-Newton or steepest decent methods where the asymptotic convergence rate is adversely affected by an increase in the condition of the projected Hessian matrix $H_z = Z^THZ$. It can be shown that

$$\text{cond}(H_z) \leq \text{cond}(H)(\text{cond}(Z))^2.$$

allowing the condition of the projected Hessian to be degraded if the condition of Z is large. The matrix Z determined from the QR decomposition, being an orthonormal matrix, is such that $\text{cond}(Z) = 1$. The condition of the projected Hessian does not deteriorate for this choice of Z [29].

3.4 Model Algorithm for Solving the Linearly Constrained Optimization Problem

Before discussing aspects of various algorithms that have either been used to compute chemical equilibria or are applicable to the problem, it is useful to describe a model algorithm for reduced space methods. This model algorithm which applies to linearly constrained minimization problems of the form:

$$\begin{aligned} \min_{x \in \Re^n} & f(x) \\ \text{subject to} \quad Ax &= b \\ Cx &\geq d \\ \text{where } A &\in \Re^{m_1} \times \Re^n \\ C &\in \Re^{m_2} \times \Re^n \\ b &\in \Re^{m_1},\ d \in \Re^{m_2} \end{aligned}$$

is from the book by Gill, Murray and Wright [29]:

Step 0 Initialization:

Find a feasible vector $x^{(0)}$, set $k \leftarrow 0$.

Step 1 Test for convergence:

Terminate algorithm if the conditions for convergence are satisfied at $x^{(k)}$.

Step 2 Constraint dropping strategy:

Choose to either continue the minimization in the current subspace or to remove a constraint from the working set. If a constraint is to be removed, go to step 6, otherwise continue to step 3.

Step 3 Search direction:

Compute a nonzero vector p_z. The direction of search, $p_{(k)}$, is then given by

$$p^{(k)} = Zp_z$$

Step 4 Compute a step length:

Calculate α^{MAX}, the maximum positive feasible step along $p^{(k)}$. Compute a positive scalar $\alpha^{(k)}$, the step length which ensures that $f(x^{(k)} + \alpha^{(k)}p^{(k)}) < f(x^{(k)})$, and $\alpha \leq \alpha^{\mathrm{MAX}}$. Go to step 7 if $\alpha < \alpha^{\mathrm{MAX}}$; otherwise continue to step 5.

Step 5 Add a constraint to the working set:

If $\alpha^{(k)}$ is the step to the constraint with index r add r to $I^{(k)}$, and modify the associated quantities accordingly. Go to step 7.

Step 6 Remove a constraint from the working set:

Select the constraint to be deleted. Delete the constraint from the working set and update the associated quantities accordingly.

Step 7 Update the estimate of the minimizer: Set $x^{(k+1)} \leftarrow x^{(k)} + \alpha^{(k)}p^{(k)}, k \leftarrow k+1$, and go back to step 1.

There are many variations on this model algorithm. An important part of the algorithm is the strategy that is used to decide when to add or remove constraints from the working set. This topic is discussed in section 3.6.2. At the heart of the algorithm lies the method of minimizing the objective function over a working manifold. This important feature of reduced space algorithms, characterized primarily by the method employed to determine the search direction in step 3, is described in section 3.5.

3.5 Minimization Over a Working Manifold

Minimization over a manifold is in many respects similar to the problem of unconstrained minimization, resulting in the majority of unconstrained minimization techniques being applicable to this problem. The techniques presented in this section are not representative of the full array of possible methods but are intended to either demonstrate some properties of these algorithms or to place historically significant chemical equilibrium algorithms in context.

3.5.1 Methods of Steepest Descent

The steepest descent method, originally proposed by Cauchy for the unconstrained minimization problem, yields a search direction based on the gradient of the objective function:

$$p^{(k)} = -\nabla f \tag{3.7}$$

This method can be approached from the viewpoint of minimizing a first order Taylor series approximation of the objective function:

$$f(x^{(k)} + p) \approx f^{(k)} + (\nabla f)^T p.$$

As this function has no minimum some restriction has to be placed on the length of p. The problem of finding the most appropriate search direction p to reduce the linear function is stated as:

$$\min_{p \in \Re^n} \frac{(\nabla f)^T p}{\|p\|}. \tag{3.8}$$

where $\|.\|$ is some type of norm. When the two norm is used, that is $\|p\| = \left(p^T p\right)^{\frac{1}{2}}$, the solution to the problem is given by the familiar equation 3.7. The generalized form of the steepest descent method gives rise to an extensive family of search directions, each defined according to the choice of norm used in equation 3.8. Application of steepest descent methods to minimization over a manifold is a straightforward extension of the unconstrained case. A first order Taylor series approximation of f along any direction on the manifold around a point $x^{(k)}$ is:

$$f(x_k + Zp_z) = f^{(k)} + \nabla f^{(k)T} Zp_z.$$

The problem of finding the best search direction is therefore similar to the unconstrained case

$$\min_{p_z \in \Re^{n-m}} \frac{\left(\nabla f^{(k)}\right)^T Zp_z}{\|Zp_z\|} \tag{3.9}$$

Here the norm restriction has been placed on the search direction $p^{(k)} = Zp_z$. If the two norm is used, the solution to this problem yields the steepest descent direction [29]:

$$p^{(k)} = -Z\left(Z^T Z\right)^{-1} Z^T \left(\nabla f^{(k)}\right)$$

If the two norm is applied to p_z instead of Zp_z, as in equation 3.9, the steepest descent direction is:

$$p^{(k)} = -ZZ^T \left(\nabla f^{(k)}\right).$$

Naphtali [51, 52] used this search direction in the computation of chemical equilibria, and demonstrated an algorithm based on this idea for an ideal gas system involving a hydrocarbon - oxygen mixture. In this implementation the non-negativity constraints were taken into

account by limiting the step length to the feasible region. According to Smith and Missen [73], this algorithm has been found to converge slowly particularly near the solution. This slow asymptotic convergence rate, the cause of which having been elucidated on theoretical grounds, has been shown to be a characteristic of all first order methods. A theoretical analysis of the asymptotic convergence of the steepest descent method is based on the behaviour of the algorithm when applied to a positive definite quadratic function $f(x) = c^T x + \frac{1}{2} x^T Q x$, where the step length used is obtained from an accurate line search. As seen from the Taylor expansion of any twice differentiable function,

$$f(x^{(k)} + p) = f^{(k)} + \nabla f^{(k)T} p + \frac{1}{2} p^T \nabla^2 f^{(k)} p + \ldots$$

a quadratic approximation is valid in a sufficiently small region around the point $x^{(k)}$. The main result of the analysis is that

$$\begin{aligned} f(x^{(k+1)}) - f(x^{(k)}) &\approx \frac{(\lambda_{\max} - \lambda_{\min})^2}{(\lambda_{\max} + \lambda_{\min})^2} \left(f(x^{(k)}) - f(x^*) \right) \\ &= \frac{(\kappa - 1)^2}{(\kappa + 1)^2} \left(f(x^{(k)}) - f(x^*) \right), \end{aligned}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are respectively the maximum and minimum eigenvalues of Q, x^* is the local minimum of the quadratic function, and $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$ is the spectral condition number of Q [42]. Consequently, the steepest descent method will converge slowly if the objective function is poorly conditioned near the solution. For example, if Q has a moderate condition number of 100 then the following relation applies

$$f(x^{(k+1)}) - f(x^{(k)}) \approx \frac{(99)^2}{(101)^2} \left(f(x^{(k)}) - f(x^*) \right) \approx 0.96 \left(f(x^{(k)}) - f(x^*) \right).$$

hence little progress is made on each iteration. One of the features of the chemical equilibrium problem is the possible ill-conditioning of the projected Hessian matrix. The effects of this ill-conditioning are demonstrated in the following example.

Example 4 *In an ideal gas system containing the four components N, O_2, NO and NO_2, the objective function is:*

$$f = \alpha \sum_{i=1}^{4} \left\{ n_i G_i^f + n_i \ln(n_i) - n_i \ln \left(\sum_{j=1}^{4} n_j \right) \right\},$$

where α and G_i^f are constants.

and null-space basis vectors are $(-2, -1, 2, 0)$ and $(0, -1, -2, 2)$. The Hessian matrix for

this system is

$$\nabla^2 f = \frac{\alpha}{\sum_{i=1}^{4} n_i} \begin{bmatrix} \frac{\sum_{i=1}^{4} n_i}{n_1} - 1 & -1 & -1 & -1 \\ -1 & \frac{\sum_{i=1}^{4} n_i}{n_2} - 1 & -1 & -1 \\ -1 & -1 & \frac{\sum_{i=1}^{4} n_i}{n_3} - 1 & -1 \\ -1 & -1 & -1 & \frac{\sum_{i=1}^{4} n_i}{n_4} - 1 \end{bmatrix}.$$

Letting $n_1 = 1, n_2 = 2, n_3 = 0.001$, *and* $n_4 = 1$ *we have*

$$\nabla^2 f = \frac{\alpha}{4.001} \begin{bmatrix} 3.001 & -1 & -1 & -1 \\ -1 & 1.0005 & -1 & -1 \\ -1 & -1 & 4000 & -1 \\ -1 & -1 & -1 & 3.001 \end{bmatrix}$$

,and using the null-space basis vectors as columns of Z,

$$Z = \begin{bmatrix} -2 & 0 \\ -1 & -1 \\ 2 & -2 \\ 0 & 2 \end{bmatrix}$$

the projected Hessian is

$$Z^T \nabla^2 f Z = \alpha \begin{bmatrix} 16021. & -16003. \\ -16003. & 16021. \end{bmatrix},$$

which has a spectral condition number of 1779.1.

The large condition number in chemical equilibrium problems is due to the species that are present in small amounts. In theory these quantities can be arbitrarily small, but in practice it is necessary to set a positive lower bound on vanishing molar quantities to alleviate, amongst other numerical difficulties, ill-conditioning of the projected Hessian matrix.

3.5.2 Newton Type Methods

Newton's method can be used to solve an unconstrained nonlinear optimization problem by directly satisfying the optimality conditions. These optimality conditions

$$\nabla f(x) = 0$$

are a set of nonlinear equations for nonquadratic problems. If ∇f is nonsingular, the kth Newton step involves the solution of the local affine model of the set of equations

$$\begin{aligned} H^{(k)}p^{(k)} &= -\nabla f^{(k)}, \\ \text{where } H^{(k)} &= \nabla^2 f(x^{(k)}) \\ \text{and } p^{(k)} &= x^{(k+1)} - x^{(k)}, \end{aligned}$$

$x^{(k+1)}$ being the zero of this affine model. From an alternative point of view, consider the local quadratic model of $f^{(k)}$as given by the Taylor series expansion about $x^{(k)}$

$$f(x^{(k)} + p^{(k)}) \approx f^{(k)} + \nabla f^{(k)T} p^{(k)} + \frac{1}{2} p^{(k)T} H^{(k)} p^{(k)}$$

If $H^{(k)}$is positive definite, $x^{(k+1)}$ as defined by the Newton step $p^{(k)}$ is the local minimizer of this quadratic model. To ensure convergence to a local minimum from an arbitrary initial point a reduction in some merit function is required on each iteration. For this reason a linear search in the Newton direction is usually carried out :

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$$

where α is a positive scalar called the step length. Usually the merit function is the objective function itself and the step length is therefore an approximate of the minimum along the search direction.

Noting that

$$f(x^{(k)} + Zp_z^{(k)}) \approx f^{(k)} + \nabla f^{(k)T} Zp_z^{(k)} + \frac{1}{2} p^{(k)T} Z^T H^{(k)} Zp^{(k)}$$

Newton's method can be applied to the linearly constrained minimization problem by replacing the gradient and Hessian of $f^{(k)}$ by the reduced gradient $\nabla f^{(k)} Z$ and the projected Hessian $Z^T H^{(k)} Z$. The Newton step $p^{(k)} = Zp_z^{(k)}$is then found from the solution to the linear system

$$Z^T H^{(k)} Z p_z = -\nabla f^{(k)} Z.$$

The vector p_z corresponds to the step to the local minimum of the quadratic model of the objective function restricted to the null-space. Note that when the projected Hessian matrix is not positive definite, the local quadratic model may not have a local minimum and Newton's method therefore needs to be modified under these circumstances. In chemical equilibrium problems where the ideal gas law is used to describe the Gibbs energy function, the objective function is convex and the projected Hessian is therefore positive definite. In cases where a number of condensed, pure component phases are included in the system, the projected Hessian matrix may become singular during a computation. In these cases and in systems where nonconvex thermodynamic models are used, modifications to Newton's method are necessary. Even in cases where the projected Hessian is positive definite, a large

condition number may introduce numerical difficulties into the solution of the linear system. Under such circumstances a modification of Newton's method is also required. One of the most stable options available for treating an indefinite Hessian is the modified Cholesky factorization [27].

Newton's method has been applied to the chemical equilibrium problem to solve the necessary optimality conditions in terms of reaction extents by Stone [74]. Ma and Shipman[43] suggested a method whereby a steepest descent algorithm is used in the initial stages followed by Newton's method in the final stages.

3.5.3 The Villars-Cruise-Smith Algorithm

Villars [78], Cruise [14], Smith [71], and Smith and Missen [72, 73] developed a method for ideal gas systems which is an approximation of Newton's method and exhibits the properties of a steepest descent method in the sense that no linear system needs to be solved by numerical means. The method exploits the form of the Hessian and the flexibility of the variable reduction technique to form an easily invertible projected Hessian matrix. Consider the generic representation of the projected Hessian matrix:

$$Z^T H Z = \begin{bmatrix} W^T & I_{n-m} \end{bmatrix} \begin{bmatrix} R & S \\ S^T & Q \end{bmatrix} \begin{bmatrix} W \\ I_{n-m} \end{bmatrix} = W^T R W + S^T W + W^T S + Q$$

where the species have been reordered so that the "component" species are the first m species and W is the $m \times (n-m)$ matrix $-V^{-1}U$ as in equation 3.6. For an ideal gas system the $m \times m, m \times (n-m)$, and $(n-m) \times (n-m)$ matrices R, S, and Q can be shown to have the following form.

$$R = R_1 + R_2 = \begin{bmatrix} \frac{1}{n_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{1}{n_m} \end{bmatrix} + \frac{-1}{\sum_{i=1}^{n} n_i} \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

$$S = \frac{-1}{\sum_{i=1}^{n} n_i} \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

$$Q = Q_1 + Q_2 = \begin{bmatrix} \frac{1}{n_{m+1}} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{1}{n_n} \end{bmatrix} + \frac{-1}{\sum_{i=1}^{n} n_i} \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

If the "component" species are chosen so that their molar quantities are all larger than those of the "non-component" species, the diagonal elements in R_1 will be smaller than those in

Q_1. In addition the elements in S, R_2 and Q_2 will be relatively small. On this basis Smith and Missen [73] argue that the projected Hessian is dominated by the diagonal matrix Q_1 and can therefore be approximated as a diagonal matrix

$$Z^T H Z \approx \tilde{H} \equiv \text{diag}\left(Z^T H Z\right)$$

which is easily inverted in closed form. Replacing the Hessian by this matrix the following is used to compute a search direction in the minimization procedure:

$$p_z = -\tilde{H}^{-1} Z^T \nabla f$$

This method can easily be extended to include multiple ideal phases. Pure condensed phases can be included by augmenting the matrix for the multicomponent phases with an identity matrix to take into account the pure phases. In all cases the method yields a direction of descent because $\tilde{H}$ is a positive definite matrix.

Historically the idea of using an arbitrary null-space rather than physically meaningful chemical reactions was proposed by Villars who in 1959 [78] solved the classical chemical equilibrium equations by adjusting each equation one at a time. Cruise [14] then introduced the choice of "component" species as described above, and proposed that all the equations be manipulated simultaneously. These two improvements resulted in substantial improvements in convergence rates and reliability. Smith and Missen [72] made further improvements on the reliability of the methods by viewing the process as a minimization procedure and introducing a step length parameter to ensure that a decrease in the objective is achieved on every step.

Although the method benefits from the computational savings gained by avoiding the need to solve a linear system on each iteration, this consideration needs to be balanced against the effect it is likely to have on the convergence rate of the algorithm. Loss of accuracy in the second order approximation of the objective function results in a loss of the quadratic asymptotic convergence rate associated with a pure Newton's method; as shown in section 3.5.1 a linear convergence rate in the case of a poorly conditioned problem can be arbitrarily slow.

3.6 Active Set Strategy

In the discussion on the characteristics of chemical equilibrium problems, we noted that inequality constraints may be included for the following reasons:

1. A species in a multicomponent phase can only vanish if the phase itself disappears. The disappearance of a phase can be taken into account by including in the working set of constraints all the inequality constraints corresponding to the species in that phase. Inequality constraints are therefore a means of dealing with the lack of *a priori* information on the phases present at equilibrium.

2. In chemical equilibrium problems it is desirable to set a small positive lower bound on species to alleviate the ill-conditioning problems associated with vanishing species.

The earlier methods such as the Villars-Cruise-Smith algorithm did not formally employ active set constraints, but removed species from the calculation when a species tended to violate physical constraints. This sometimes resulted in convergence to what Gautam and Seider [22] termed a constrained minimum. Gautam and Seider were not the first to recognize the phenomenon; Oliver et al. [55] for example used a test based on Lagrange multiplier estimates to reintroduce species back into the computation. The treatment of inequality constraints was however more of a "fix" tagged onto the algorithm than an integral part of the procedure. Castillo and Grossmann [12] used Sargent and Murtagh's[67] reduced space method which treated inequality constraints in as an inherent part of the minimization problem.

Precise rules on adding and deleting constraints from the working set do not exist and are likely to vary from problem to problem. Addition of a constraint to the working set certainly occurs once the constraint in question becomes a hindrance to the minimization procedure's progress, constraint deletion, on the other hand, involves a more subtle mechanism. General methods used to decide whether or not an inequality constraint should be deleted from the working set are discussed in section 3.6.1. Applications of the active set strategy to chemical equilibrium problems are discussed in section 3.6.2.

3.6.1 Deleting a Constraint from the Active Set

This section is based mainly on Gill, Murray and Wright's [29] exposition of the subject. An inequality constraint plays a role in defining the minimizer only when it is active at the solution. If the set of active constraints were known, the inequality constrained problem could, in theory, be replaced by an equality constrained problem:

$$\begin{aligned} &\min_x f(x) \\ &\text{subject to } \hat{A}x = b. \end{aligned}$$

The question we consider here is how to verify a solution where some of the equality con-

straints are in fact working inequality constraints. If $x^{(k)}$ is optimal for the equality constrained problem the following Karush-Kuhn-Tucker optimality relation is satisfied:

$$\nabla f^{(k)} = \hat{A}^{(k)T}\hat{\lambda} \tag{3.10}$$

where $\hat{\lambda}$ is the Lagrange multiplier vector corresponding to the set of working constraints. If any component of $\hat{\lambda}$ corresponding to a working inequality constraint is negative, f can be reduced by moving in a direction which is feasible, in which case $x^{(k)}$ is not optimal for the linear inequality constrained optimization problem. In this way the Lagrange multipliers can be used to determine which constraints should be deleted from the working set.

The approach of dropping constraints from the working set only once a solution has been obtained for the equality constrained problem can be very inefficient because the problem being solved may have the wrong set of equality constraints. A strategy that enables working inequality constraints to be removed before accurate convergence has been achieved is therefore considered. When convergence has not be achieved, the linear system 3.10 may be inconsistent, and $\hat{\lambda}$ can only be estimated using 1st or 2nd order derivative information about the point $x^{(k)}$. Unfortunately, no known estimation technique can guarantee that even the signs of the estimated components of $\hat{\lambda}$ are correct unless $x^{(k)}$ is close enough to the working set minimizer. The use of Lagrange multiplier estimates may therefore lead to an inappropriate removal of a constraint. The phenomenon of zig-zagging which occurs when a constraint is repeatedly removed on one iteration and added on the next can severely hamper the progress or in the worst case cause convergence to a nonoptimal point.

3.6.2 Active Set Methods in the Chemical and Phase Equilibrium Problem

The active set method of Sargent and Murtagh [67] was used by Castillo and Grossmann [12] to solve the chemical equilibrium problem. In this method, the working set strategy tries to maintain a minimal number of active constraints in the working set, and more than one constraint may be added or dropped from this working set on any iteration. Lenard [40] studied the relative benefits of different working set strategies by implementing variations of the Sargent and Murtagh algorithm. Except in the case when the optimal solution is at a vertex, the best active set strategy was found to be that which keeps the set of active constraints as small as possible. This study was applied to linearly constrained nonlinear programming methods in general and the conclusions are by no means applicable to all methods and all problems, and may be inappropriate in the chemical equilibrium context. Nevertheless, the observation that a particular working set strategy may be significantly more successful than others is a useful one.

An example of a working set method tailored to the requirements of multiphase chemical equilibrium problems appears in the method proposed by Harvie, Greenberg and Weare [32]. This method is actually an adaptation of Gill and Murray's [26] method for general linearly constrained nonlinear programs. Recognizing that the exclusion of a phase results in the constraints for all species becoming active, they devised a specialized strategy whereby species may be added or deleted one at a time or simultaneously in the event of phase formation or annihilation. Tests for introducing single or multispecies phases can be derived by applying equation 3.10 to the chemical equilibrium problem:

$$\nabla G \approx A^T \lambda^A + \hat{C}^T \lambda^{\hat{C}}.$$

In this equation, A and $\hat{C}$ represent the mass balance and working inequality constraints respectively. The multipliers corresponding to these equations are λ^A and $\lambda^{\hat{C}}$. Because the non-negativity constraints are simple bounds, this relation can be expressed as:

$$\nabla G \approx A^T \lambda^A + \lambda^{\hat{C}}.$$

Noting that all directions p which are feasible with respect to the mass balance constraints can be written as Zp_z, an estimate for feasible Gibbs energy changes can be derived:

$$\begin{aligned} p_z^T Z^T \nabla G &\approx p_z^T Z^T A^T \lambda^A + p_z^T Z^T \lambda^{\hat{C}}, \\ &\approx p_z^T Z^T \lambda^{\hat{C}}, \\ \frac{dG}{dn_{ik}} &\approx \lambda_{ik}^{\hat{C}}. \end{aligned}$$

The test for addition of a pure phase k is based on the relation:

$$\frac{dG}{dn_{ik}} \approx \lambda_{ik}^{\hat{C}},$$

where n_{ik} is the single species belonging to phase k. When $\lambda_{ik}^{\hat{C}}$ is negative the Gibbs energy can be reduced by introducing phase k. The Gibbs energy change on introducing a multispecies phase is:

$$\begin{aligned} dG &\approx \sum_{j \in C^k} \lambda_{jk}^{\hat{C}} dn_{jk}, \\ &\approx d\left\{ (n_k) \sum_{j \in C^k} \lambda_{jk}^{\hat{C}} x_{jk} \right\}, \end{aligned}$$

where x_{jk} is the mole fraction of species j in solution phase k. The Gibbs energy change dG is therefore dependent on the composition of the phase to be introduced, and any composition for which dG is negative is therefore a candidate for phase addition. Testing the viability of adding a solution phase therefore requires some technique to determine the optimal composition of this phase. This can be done by minimizing $\lambda_k^{\hat{C}} = \sum_{j \in C^k} \lambda_{jk}^{\hat{C}}$ with respect to the

composition of the incipient phase. Harvie et al. [32] posed the following minimization subproblem for this purpose:

$$\begin{aligned} \min_{x_{jk}} \lambda_k^{\hat{C}} &= \sum_{j \in C^k} x_{jk} \left(\nabla G_{jk}(x) - \sum_{j \in C^k} a_{ejk} \lambda_{jk}^{A} \right) \\ \text{subject to } \sum_{j \in C^k} x_{jk} &= 1 \end{aligned}$$

Harvie et al. [32] refer to $\lambda_k^{\hat{C}}$ as the "phase multiplier" for phase k as it plays the same role for phase addition and deletion as the species multiplier do for species addition or deletion. It can be shown that $\lambda_k^{\hat{C}} = \lambda_{jk}^{\hat{C}}$ for all j when $\lambda_k^{\hat{C}}$ is the minimum for the above problem. To prevent zigzagging due to premature phase inclusion, a phase is introduced only when

$$\left| \lambda_k^{\hat{C}} \right| > \max_i \left\{ \left(Z^T \nabla G \right)_i \right\},$$

the motivation being that if a multiplier is small relative to the norm of the gradient projected on the working subspace, better progress is likely to result if the current set is retained. This is in accord with the general principle that a constraint be deleted from the working set only if a lower value of the objective function is likely to result following the deletion than if the constraint were not deleted [29].

Chapter 4

Lagrangian Methods

The methods described in this section, as in the previous one, make use of a working set strategy to incorporate inequality constraints. Let the following problem represent a minimization over a working set:

$$\min_{x \in \Re^n} f(x) \quad \text{subject to} \quad Ax = b.$$

Lagrangian methods solve this problem by making use of the Karush-Kuhn-Tucker first order necessary optimality conditions:

$$\begin{aligned} \nabla f(x^*) &= A^T \lambda^* \\ Ax^* &= b \end{aligned} \tag{4.1}$$

where x^* is a constrained local minimum and λ^* is the vector of Lagrange multipliers. The Lagrangian function corresponding to the minimization problem is

$$L(x, \lambda) = f(x) - \lambda^T (Ax - b).$$

The optimality conditions 4.1 are satisfied when the gradients of the Lagrangian are zero. The Lagrangian methods discussed in this chapter solve a minimization problem by searching the $m + n$ dimensional space for values of x and λ which satisfy the optimality conditions.

4.1 A Model Lagrangian Algorithm for Solving the Linearly Constrained Optimization Problem

As in the case of reduced space methods, there are a number of ways of using Lagrangian type methods to solve the chemical equilibrium problem. A Lagrangian based model algorithm

for minimization problems of the form:

$$\begin{aligned} &\min_{x\in\Re^n} f(x) \\ \text{subject to} \quad Ax &= b \\ Cx &\geq d \\ \text{where} \quad A &\in \Re^{m_1}\times\Re^n \\ C &\in \Re^{m_2}\times\Re^n \\ b &\in \Re^{m_1},\ d\in\Re^{m_2} \end{aligned}$$

is described below.

Step 1 Test for convergence:

Terminate algorithm if the conditions for convergence are satisfied at $x^{(k)}$.

Step 2 Carry out the constraint dropping strategy:

Choose to either continue the minimization in the current subspace or to remove a constraint from the working set. If a constraint is to be removed, go to step 6; otherwise, continue to step 3.

Step 3 Search direction:

Solve the linear system 4.1, then determine a direction of search, $p^{(k)}$ from this solution.

Step 4 Compute a step length:

Calculate α^{MAX} the maximum positive feasible step along $p^{(k)}$. Compute a positive scalar $\alpha^{(k)} \leq \alpha^{\text{MAX}}$ which ensures that $f(x^{(k)}+\alpha^{(k)}p^{(k)}) < f(x^{(k)})$. Go to step 7 if $\alpha < \alpha^{\text{MAX}}$; otherwise continue to step 5.

Step 5 Add a constraint to the working set I:

If $\alpha^{(k)}$ is the step to the constraint with index r, add r to $I^{(k)}$, and modify the associated quantities accordingly. Go to step 7.

Step 6 Remove a constraint from the working set:

Select the constraint to be deleted. Delete the constraint from the working set and update the associated quantities accordingly.

Step 7 Update the estimate of the minimizer:

Set $x^{(k+1)} \leftarrow x^{(k)}+\alpha^{(k)}p^{(k)}$, $k \leftarrow k+1$, and go back to step 1.

This model algorithm is very similar to that for the reduced space method as both methods use the active set strategy and require minimization algorithm paraphernalia such as step length procedures and convergence checks. The main difference between the methods is the means by which a feasible search direction is found. The solution of the linear system of equations in step 3 is the focus of the discussion below.

4.2 Solving the Karush-Kuhn-Tucker System by Newton's Method

The Karush-Kuhn-Tucker conditions form a set of nonlinear and linear equations. Newton's method is the most popular method of solving systems of nonlinear equations. Suppose we seek a point x^* which satisfies the general system of equations

$$\Phi(x) = 0$$

where Φ is a vector of smooth scalar functions $\psi_i(x), \quad i = 1, .., n$. If the Jacobian matrix of $\Phi(x)$, denoted $J(x)$, is nonsingular, the k^{th} Newton step $p^{(k)}$ is the step from $x^{(k)}$ to the zero of the linearized model of Φ

$$J^{(k)}p^{(k)} = -\Phi^{(k)}$$

where $J^{(k)}$ and $\Phi^{(k)}$represent $J(x^{(k)})$ and $\Phi(x^{(k)})$ respectively. Newton's method when applied to the system of linear and nonlinear equations associated with the optimality conditions 4.1 results in the linear system

$$\begin{bmatrix} H^{(k)} & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} p^{(k)} \\ \delta^{(k)} \end{bmatrix} = \begin{bmatrix} H^{(k)} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} p^{(k)} \\ -\delta^{(k)} \end{bmatrix} = \begin{bmatrix} -\nabla f^{(k)} + A^T\lambda^{(k)} \\ b - Ax^{(k)} \end{bmatrix}$$

where $p^{(k)}$ and $\delta^{(k)}$ denote the Newton step for the x and λ variables respectively, and $H^{(k)}$ represents the Hessian of the objective function evaluated at $x^{(k)}$. An alternative form is derived by substituting $\lambda^{(k+1)} = \lambda^{(k)} + \delta^{(k)}$ for $\delta^{(k)}$:

$$\begin{bmatrix} H^{(k)} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} p^{(k)} \\ -\lambda^{(k+1)} \end{bmatrix} = \begin{bmatrix} -\nabla f^{(k)} \\ b - Ax^{(k)} \end{bmatrix}. \tag{4.2}$$

Viewed from a minimization perspective each Newton step finds the solution to a quadratic programming problem which is a local approximation of the original nonlinear problem. Minor variations of the above approach have been applied to the chemical equilibrium problem by Boynton [6], Zeleznik and Gordon [86], Gautam and Seider [22] and Heuze et al.[33]. When applied to ideal systems, it is possible to reduce the number of variables involved in the numerical solution of the linear system from $n + m$ to $m + 1$. Such a scheme, referred to in the literature as the RAND algorithm, was derived in 1958 by White, Johnson and Dantzig [81] of the RAND corporation.

4.2.1 The RAND Algorithm for Ideal Systems

Consider the chemical equilibrium problem involving a single ideal gas phase where the dimensionless Gibbs energy function is defined as:

$$\begin{aligned} \frac{G}{RT} &= \sum_{i \in C} n_i \left(\frac{G_i^f}{RT} + \ln n_i - \ln n_t \right) \\ \text{where } n_t &= \sum_{i \in C} n_i, \\ & \text{and } G_i^f \text{ is a constant.} \end{aligned}$$

The Hessian of the objective function has a form such that the element in the i^{th} row and j^{th} column of $\nabla^2 f$ is $\frac{\delta_{ij}}{n_i} - \frac{1}{n_t}$, where δ_{ij} is the Kronecker delta. Substitution into the first row of 4.2 yields the linear system:

$$\begin{bmatrix} \text{diag}\left(\frac{1}{n^{(k)}}\right) & -\frac{1}{n_t^{(k)}} e & A^T \end{bmatrix} \begin{bmatrix} p^{(k)} \\ e^T p^{(k)} \\ -\lambda^{(k+1)} \end{bmatrix} = -\nabla f^{(k)},$$

where $\text{diag}\left(\frac{1}{n}\right)$ is a diagonal $n \times n$ matrix with $\frac{1}{n_i}$ being the i^{th} diagonal element, and e is an n- vector of ones. If a new variable $u^{(k)} = \frac{e^T p^{(k)}}{n_t^{(k)}}$ is introduced then $p^{(k)}$ can be found explicitly in terms of $u^{(k)}$ and $\lambda^{(k+1)}$

$$p^{(k)} = \text{diag}(n^{(k)}) \left(A^T \lambda^{(k+1)} + e u^{(k)} - \nabla f^{(k)} \right) \tag{4.3}$$

where $\text{diag}(n_i)$ is the inverse of $\text{diag}\left(\frac{1}{n_i}\right)$. Substituting the above expression into the second row of 4.2 yields the set of linear equations

$$\begin{bmatrix} A\,\text{diag}(n^{(k)}) A^T & A\,\text{diag}(n^{(k)}) e \end{bmatrix} \begin{bmatrix} \lambda^{(k+1)} \\ u^{(k)} \end{bmatrix} = \left[A\,\text{diag}(n^{(k)}) \nabla f^{(k)} + b - A n^{(k)} \right] \tag{4.4}$$

Another linear relation is obtained from the dot product $e \cdot p^{(k)}$ in the definition of $u^{(k)}$:

$$e^T p^{(k)} - u^{(k)} n_t^{(k)} = 0.$$

By substituting expression 4.3 for $p^{(k)}$ and rearranging, the following is obtained:

$$e^T \text{diag}(n^{(k)}) A^T \lambda^{(k+1)} + e^T \text{diag}(n^{(k)}) e u^{(k)} - u^{(k)} n_t^{(k)} = e^T \text{diag}(n^{(k)}) \nabla f^{(k)}.$$

Noting that $e^T \text{diag}(n^{(k)}) e = n_t^{(k)}$, this expression can be simplified to:

$$\begin{bmatrix} e^T \text{diag}(n^{(k)}) A^T & 0 \end{bmatrix} \begin{bmatrix} \lambda^{(k+1)} \\ u^{(k)} \end{bmatrix} = \left[e^T \text{diag}(n^{(k)}) \nabla f^{(k)} \right] \tag{4.5}$$

Each iteration of the RAND algorithm involves solving the set of $m+1$ linear equations 4.4 and 4.5, then using 4.3 to determine $p^{(k)}$. A line search is then carried out in the direction $p^{(k)}$ so that $n^{(k+1)} = n^{(k)} + \alpha p^{(k)}$ results in a decrease in the objective function.

The RAND algorithm has been extended by Oliver et al.[55] to include systems that have a number of single species phases. Boynton [6] and Raju and Krishnaswami [62] adapted the RAND algorithm to include more than one multispecies phase. In the case where there are P^M multispecies phases and P^S single species phases, the dimensionless Gibbs energy function becomes:

$$\frac{G}{RT} = \sum_{k=1}^{P^M} \sum_{i \in C^k} n_{ik} \left(\frac{G_i^f}{RT} + \ln n_{ik} - \ln n_{tk} \right) + \sum_{k=P^M+1}^{P^M+P^S} n_k \frac{G_k^f}{RT},$$

where phases $1 \ldots P^M$ are multispecies phases

and phases (P^M+1) $\ldots$ (P^M+P^S) are single species phases.

Two modifications are necessary to incorporate the single species phases into the RAND scheme:

1. As there are P^S additional variables associated with these phases, P^S additional relations need to be added to the linear system of equations. The Karush-Kuhn-Tucker optimality conditions supply the following relations involving the single species phases:

$$A_s^T \lambda^{(k+1)} = \nabla f_s^{(k)}.$$

2. The single species phases must be included in the mass balance relations. Let $p_s^{(k)}$ denote the step $n_s^{(k+1)} - n_s^{(k)}$ where n_s is the vector of moles corresponding to the single species phases. The second row from equation 4.2:

$$\begin{bmatrix} A & 0 \end{bmatrix} \begin{bmatrix} p^{(k)} \\ -\lambda^{(k)} \end{bmatrix} = b - An^{(k)}$$

can be expanded to:

$$\begin{bmatrix} A_m & A_s & 0 \end{bmatrix} \begin{bmatrix} p_m^{(k)} \\ p_s^{(k)} \\ -\lambda^{(k)} \end{bmatrix} = b - An^{(k)}$$

$$A_s p_s^{(k)} + \begin{bmatrix} A_m & 0 \end{bmatrix} \begin{bmatrix} p_m^{(k)} \\ -\lambda^{(k)} \end{bmatrix} = b - An^{(k)}.$$

The RAND equations can now be derived in a similar way to the single multispecies phase case by substituting the multiphase analog of equation 4.3 for $p_m^{(k)}$.

By incorporating these two modifications into the RAND equations, a linear system analogous to the system of equations 4.4 and 4.5 is derived:

$$\begin{bmatrix} A_m \Lambda^{(k)} A_m^T & A_m \Lambda^{(k)} E & A_s \\ E^T \Lambda^{(k)} A_m^T & 0 & 0 \\ A_s^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda^{(k+1)} \\ u_m^{(k)} \\ p_s^{(k)} \end{bmatrix} = \begin{bmatrix} A_m \Lambda^{(k)} \nabla f_m^{(k)} + b - A n^{(k)} \\ E^T \Lambda^{(k)} \nabla f_m^{(k)} \\ \nabla f_s^{(k)} \end{bmatrix} \tag{4.6}$$

In this relation Λ is $\mathrm{diag}(n_{ik})$ with the species from the multispecies phases being on the diagonal, A_m and A_s denote those blocks of the linear constraint matrix A which correspond to the multispecies and single species phases respectively, u_m denotes a P^M-vector, and E denotes a matrix consisting of columns of zeros and ones, the ones in the k^{th} column corresponding to the variables associated with multispecies phase k [73].

Some of the problems associated with the RAND algorithm when applied to a system involving an ideal gas and a number of pure solid phases are described by Smith [70]. All the problems described are attributed to singularities in the matrix 4.6. If A_m has full row rank then using exact arithmetic the matrix can be singular only if the columns of the matrix

$$\begin{bmatrix} E^T \Lambda A_m^T \\ A_s^T \end{bmatrix}$$

are linearly independent [18]. The implications of this observation vary according to the type of system being considered.

Systems involving a single ideal gas phase only cannot have a singular matrix; however, the representation of numbers using finite word lengths can introduce numerical problems in the solution of the linear system 4.6. Consider the example, described by Gordon and McBride [30], consisting of an ideal gas of O_2, H_2, and H_2O, where the total molar quantities of atomic species H and O are 2 and 1 respectively. Under conditions where H_2O is the only species present in a significant amount, the RAND system matrix,

$$\begin{bmatrix} A\,\mathrm{diag}(n^{(k)})A^T & A\,\mathrm{diag}(n^{(k)})e \\ e^T\,\mathrm{diag}(n^{(k)})A^T & 0 \end{bmatrix},$$

has the following form:

$$\begin{bmatrix} 4+\varepsilon_1 & 2+\varepsilon_2 & 2 \\ 2+\varepsilon_2 & 1+\varepsilon_3 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

where ε_i are very small positive numbers. If the ε_i are small enough, the first two rows will be linearly dependent to machine precision. Although this example is a very specialized one, it illustrates an instance of the general problem of ill-conditioning in the linear systems involved in Gibbs energy minimization; a difficulty which is a common occurrence in the chemical equilibrium problem.

When pure solid phases are included in the system, along with an ideal gas phase, the matrix can only be singular if the columns of A_s and b are linearly dependent. If it is possible to satisfy the mass balance constraints by including only gas phase species, the RAND algorithm can be applied by first solving the system without any solid phases then using criteria described by Oliver et al. [55] to introduce the solid phases that will lead to a decrease in the Gibbs energy of the system. This technique is essentially the same as the active set method. A more difficult case is one in which the mass balance constraints cannot be satisfied with gas phase species alone. When this occurs, some pure phases are forced to be included at the start of the computation, despite a lack of *a priori* information about which phases are present at equilibrium. The system can then be minimized using an *ad hoc* phase addition and removal strategy that can become very inefficient for systems involving a large number of pure phases. Note that the difficulties associated with pure phases in the RAND method arise partly because of the absence of a rigorous treatment of the inequality constraints on pure phase species. Suitable modifications of the RAND method, such as including an active set strategy for inequality constraints, should ameliorate these difficulties. Smith and Missen [73], however, suggest that the Villars-Cruise-Smith type of algorithm is more appropriate for systems involving many pure phases.

The RAND algorithm is not the only method for ideal systems which makes explicit use of the necessary optimality conditions. One alternative in the early literature is due to Brinkley [9]. Brinkley's method uses the variables $y_i = \ln n_i$ instead of n_i as the search variables in a Newton's method procedure used to solve the optimality equations. The y_i search variables cause the mass balance constraints to be nonlinear and the method to appear very different from the RAND method. Smith and Missen [73] have shown that despite the difference in appearance between the original presentations of the RAND and Brinkley methods, their working equations are in fact quite similar, the same matrix 4.6 being part of the linear system solved in both method's Newton iterations. A second alternative, the "NASA" algorithm [34], originally formulated to compute chemical equilibria for systems at constant enthalpy and pressure, was derived by Smith and Missen [73] for the isothermal, isobaric case. This method has been shown to be very closely related to the Brinkley algorithm. The main difference between the NASA, Brinkley and RAND algorithms is the use of $\ln n_i$ in the Brinkley and NASA algorithms and n_i in the RAND algorithm. Computational experiments by Zeleznik and Gordon [86] have confirmed that there is little difference in the performance of the three variations.

Chapter 5

Barrier Function Methods

The methods that have been considered up to this point use working set strategies to explicitly take the non-negativity constraints into account. One of the main features of the methods discussed in this section is their alternative treatment of inequality constraints. Barrier function methods take the inequality constraints into account through the use of a special objective function which incorporates the original objective as well as the constraints.

5.1 A Brief Historical Overview

According to Wright [82] the earliest record of the barrier function notion for mathematical programming is attributed to Frisch in 1955. This idea did not flourish until the late 1960's when Fiacco and McCormick [19] provided a rigorous theoretical basis for, what they termed, "sequential unconstrained minimization techniques" [19]. By the late 1970's, however, interest in the barrier function methods had waned due to certain theoretical difficulties associated with the minimization subproblems. In 1984, interest in barrier methods was rekindled when Karmarkar discovered an efficient algorithm of polynomial complexity for linear programming [37]. Prior to Karmarkar's discovery, the accepted way to solve the linear programming problem was by the simplex method, developed by Dantzig in 1947. Although the simplex method performs efficiently on most problems, it does not have the property of polynomial complexity, a feature which computer scientists regard as a prerequisite for a "fast" algorithm [82]. The polynomial complexity of an algorithm, discussed in more detail in the next section, refers to the fact that the number of iterations required for the algorithm to solve a general problem type is bounded from above by a polynomial function of the problem size. However, Karmarkar's algorithm was not the first polynomial complexity

algorithm to be developed for linear programming; Khachian's ellipsoid method, developed in 1979, was. Khachian's method, however, behaved poorly in practice and failed to be competitive with the simplex method. The excitement surrounding Karmarkar's discovery was due to his algorithm being both polynomial and competitive with the simplex method. The link between Karmarkar's method and the primal logarithmic barrier function was shown by Gill et al. in 1986 [28]. Since Karmarkar's discovery research into interior point methods has been intensive, producing improvements in complexity bounds and practical performance as well as extensions into fields other than linear programming — including convex programming.

5.2 Motivation of Barrier Methods

Consider the following inequality constrained minimization problem:

$$\begin{aligned} &\min_x f(x) \\ &\text{subject to } g_i(x) \geq 0 \qquad i = 1, .., m. \end{aligned}$$

The simplest barrier based on the logarithmic singularity, was first defined by Frisch in 1955 and has been frequently used both in the 1960's and in the more recent interior point methods. The logarithmic barrier function associated with the above minimization problem is:

$$\begin{aligned} B(x, \mu) &= f(x) - \mu \sum_{i=1}^{m} \ln(g_i(x)) \\ \text{where} \quad \mu &> 0 \end{aligned}$$

This function is defined only for $g_i(x) > 0$ and becomes unbounded as the boundary of the feasible region is approached. A typical barrier function method involves determining the minimum of a sequence of unconstrained minimization problems for values of μ decreasing to zero. A generic "classical" barrier function algorithm involves the following steps:

Step 0 Set $x^{(0)}$ to a strictly feasible point; in other words $g_i(x^{(0)}) > 0$ for $i = 1, .., m$. Set $\mu^{(0)}$ to a positive value and set $k \leftarrow 0$.

Step 1 Check if $x^{(k)}$ satisfies convergence criteria for an optimal solution. If so, stop with $x^{(k)}$ as the solution.

Step 2 Solve the unconstrained minimization problem

$$\min B(x, \mu^{(k)})$$

Step 3 Set $x^{(k+1)} \leftarrow x(\mu^{(k)})$, $\mu^{(k+1)} < \mu^{(k)}$ and $k \leftarrow k+1$ then return to Step 1.

A barrier function method based on the work of Fiacco and McCormick was applied to the chemical equilibrium problem by Lantagne, Marcos and Cayrol [39]. In their implementation, the non-negativity constraints on the molar quantities where taken into account using the logarithmic barrier function. An additional penalty function was applied to force satisfaction of the mass balance constraints. The penalty function like the barrier function is incorporated into the objective function to force satisfaction of the constraints. Consider for example the penalty function applied by Lantagne et al. [39] to the equality constraints $g(x) = 0$:

$$P(x) = \frac{1}{\mu} \sum_{i=1}^{m} g_i(x)^2.$$

As μ is decremented, the penalty associated with a violation of the equality constraints increases thereby forcing satisfaction of these constraints as μ tends to zero. The strength of the penalty function concept lies in the simplicity of treatment of nonlinear constraints in nonlinearly constrained minimization problems. Application of a penalty function method to linear equality constraints is, however, an unnecessary complication, as satisfaction of such constraints can be achieved by purely analytical means such as projection.

5.3 Complexity

In mathematical programming the performance of an algorithm is often assessed on its efficiency in solving test problems. This type of assessment is not satisfactory as the performance of an algorithm on one type of problem may be very different from its performance on another. The theory of computational complexity is an attempt to provide an objective measure of the efficiency of algorithms.

Ye [84] defines an *algorithm* A as a list of instructions to solve a problem P. For every instance of P, defined by a set of data Z, an algorithm either determines the instance to be infeasible or returns an output x which is close, in a well defined measure, to the solution to the problem. The *operational complexity* $c_o(A, Z)$ of an algorithm for solving a problem *instance* $Z \in Z_P$ is defined as the total number of arithmetic operations ($\times, \div, +, -$, and comparison of real numbers) required to perform this task. A more convenient measure of complexity is the *iteration complexity*, $c_i(A, Z)$, where it is assumed that each iteration can be carried out in a number of arithmetic operations which is polynomial in the number of variables n and constraints m.

Where the solution data set consists of real numbers, an error tolerance ϵ is used as a parameter in the complexity bound. A typical measure of the error for minimization problems is the absolute difference between the best known minimum and a lower bound on the global minimum. Let $c(A, Z, \epsilon)$ be the total number of operations of algorithm A for generating an ϵ-approximate solution to an instance of problem P. The complexity of this algorithm for problem P is then defined as:

$$c(A, \epsilon) \equiv \sup_{Z \in Z_P} c(A, Z, \epsilon) \qquad \text{for any } \epsilon > 0.$$

When $c(A, \epsilon) \leq f(m, n, \epsilon)$, where f is a polynomial function of m, n and $|\ln(\epsilon)|$, then the algorithm is polynomial.

5.4 Interior Point Algorithms for Convex Minimization

Some of the important features of interior point methods that differ from the 1960's barrier methods are :

- analysis of Newton's method for minimizing the barrier function;
- polynomial complexity bounds.

Whereas the classical barrier methods of the 1960's made little use of the underlying structure of the problem, interior point methods use structural properties to optimize the use of Newton's method in the solution. An important concept, which is the foundation of many interior point methods, is the central path. This path is defined by a parametrized family of approximate solutions which converge asymptotically to the exact solution. The analysis of Newton's method when the search variable is in close proximity to the central path enables complexity results to be shown.

The application of interior point methods to structured convex programming problems was initiated by Ye [85] and Monteiro and Adler [49] in their studies of convex quadratic programming. To facilitate the analysis of more general convex programming problems certain smoothness conditions had to be imposed on the functions involved. The relative Lipschitz condition proposed by Jarre [36] was used by den Hertog et al.[16] to construct a polynomial interior point method based on the classic logarithmic barrier function. A more general theory of polynomial interior point methods for convex programming was developed by Nesterov and Nemirovskii [53]. This theory is based on the self-concordance condition which has been

shown by den Hertog et al.[15] to be closely related to other smoothness conditions. The self-concordance property is defined by Nesterov and Nemirovskii as follows:

Definition 5 *Let D be a closed convex domain in a finite dimensional real vector space E, with nonempty interior $Q \equiv int(D)$ and let $a \geq 0$. Let f be a three times continuously differentiable function on E.*

1. *The function f is a-self-concordant if for all $x \in Q$ and $h \in E$:*

$$\left|\nabla^3 f(x)[h,h,h]\right| \leq 2a^{-\frac{1}{2}} \left(h^T \left(\nabla^2 f(x)\right) h\right)^{\frac{3}{2}}.$$

2. *The function f is called strongly self-concordant if it is self-concordant and goes to infinity for a sequence of points converging to the boundary of Q.*

3. *The function f is called a ϑ-self-concordant barrier if it is strongly 1-self-concordant and for all $h \in E$:*

$$\left|\nabla f(x)^T h\right| \leq \sqrt{\vartheta} \left(h^T \left(\nabla^2 f(x)\right) h\right)^{\frac{1}{2}}.$$

The a-self-concordancy condition is used to show that Newton's method behaves well when the search variable is within a well defined proximity of the central path. Strong self-concordancy and the ϑ-self-concordant barrier condition is used to prove the polynomial complexity of various path following methods. Den Hertog et al. [15] have proven the logarithmic barrier function associated with the Gibbs energy minimization problem for ideal mixtures to be a 2-self-concordant barrier enabling low complexity methods to be applied to this equilibrium problem. The lowest iteration complexity interior point algorithms established for either linear or convex programming, including the ideal Gibbs energy minimization problem, have complexity bounds of $O\left(\sqrt{n}\,|\ln(\varepsilon)|\right)$ [35]. This implies that the CPU time required to solve these problems increases at a rate proportional to the square root of the number of variables. Numerical experiments with interior point algorithms for convex programming have shown these methods to be efficient due to their ability to exploit problem structure[38].

Chapter 6

A Mixed Integer Nonlinear Programming Approach

This section examines the mixed integer nonlinear programming MINLP formulation of the chemical and phase equilibrium problem introduced by Paules and Floudas [57]. In this formulation the existence or nonexistence of a species or phase is taken into account through the use of discrete variables. Paules and Floudas promote the method on the following basis:

1. the phases that are present at equilibrium are not known *ab initio* and in a nonlinear programming formulation the inclusion of phases that are absent at equilibrium may lead to numerical difficulties in the computation due to objective function singularities and ill-conditioning of the reduced Hessian.

2. the inclusion of integer variables allows the problem to be partially generated in such a way that the terms in the objective function that cause these numerical difficulties are eliminated.

Later, it will be shown that in practice the solution methods for the MINLP formulation suffer from the same difficulties as those which solve the NLP formulation. Furthermore, the presence of additional integer variables makes the MINLP more difficult to solve.

6.1 Mixed Integer Nonlinear Programming Formulation

The key difference between the NLP formulation of the equilibrium problem and the MINLP formulation introduced here is the presence of integer variables. These new discrete variables y_{ik}, $yp_k \in \{0,1\}$ are defined as:

$$\begin{aligned} yp_k &= 1 \quad \text{if phase } k \text{ exists;} \\ yp_k &= 0 \quad \text{if phase } k \text{ does not exist;} \\ y_{ik} &= 1 \quad \text{if component } i \text{ in phase } k \text{ exists;} \\ y_{ik} &= 0 \quad \text{if component } i \text{ in phase } k \text{ does not exist.} \end{aligned}$$

The two sets of binary variables are related through the integer constraints that represent the logical requirement that a phase be active if any of the species belonging to that phase is active:

$$y_{ik} \leq yp_k \qquad \forall k \in P, i \in C^k,$$

where $C^k \subset C$ is a set that associates a set of components with each phase k. The integer variables control the continuous variables through constraints that contain both discrete and continous variables:

$$n_{ik} - \Omega_i y_{ik} \leq 0 \qquad \forall k \in P, i \in C^k.$$

In this equation, Ω_i represents an arbitrary upper bound on the number of moles of compound i. When $y_{ik} = 1$ species (i,k) is said to be active and the above constraint imposes no limitations on the continuous variable n_{ik}; when $y_{ik} = 0$ species (i,k) is inactive and the continous variable n_{ik} equals zero. The full MINLP formulation is:

$$\begin{aligned} & \min_{n,y,yp} G & & & (6.1) \\ \text{subject to } An - b &= 0 & & \\ n_{ik} &\geq 0 & & \forall k \in P, i \in C^k \\ y_{ik} &\leq yp_k & & \forall k \in P, i \in C^k \\ n_{ik} - \Omega_i y_{ik} &\leq 0 & & \forall k \in P, i \in C^k \\ y_{ik}, yp_k &\in \{0,1\}. & & \forall k \in P, i \in C^k \end{aligned}$$

Notice that all the constraints in this problem are affine in the continuous and integer variables, and integer variables are absent from the objective function.

6.2 Algorithms for the Solution of the MINLP

The mixed integer formulation has a structure that permits solution via at least two approaches:

1. Geoffrion's [25] approach which is known as the "Generalized Benders Decomposition";
2. The "Outer Approximation/ Equality Relaxation" approach of Visweswaran and Grossmann [79].

Both of these algorithms involve the type of scheme represented in figure 6.1 where the problem is approximated in the master problem as a mixed integer linear program (MILP) and the solution to the MINLP is reached via an alternating series of NLP and MILP problems.

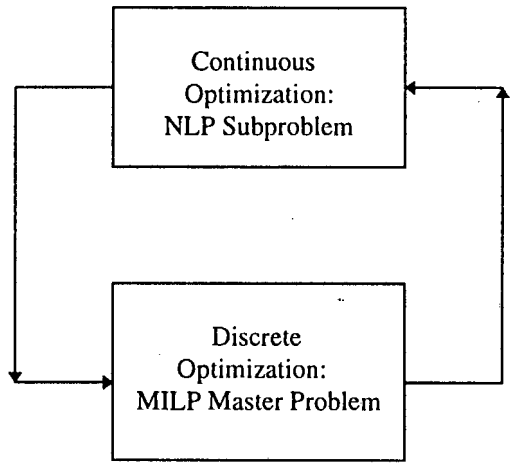

Figure 6.1: Mixed integer nonlinear progamming algorithm structure

The difference between the two approaches lies primarily in the way in which the MINLP is approximated as a MILP in the master problem: the Generalized Benders Decomposition employs duality; the Outer Approximation / Equality Relaxation uses linearization of the primal. Although the Generalized Benders Decomposition is used in this study the main conclusions apply to both methods.

6.3 Solving the MINLP Formulation Using Generalized Benders Decomposition

Finding the optimum for a problem containing both continuous and discrete variables requires elements of both nonlinear and discrete programming. Algorithms for solving continuous nonlinear programs (NLP's) and mixed integer linear programmes (MILP's) are well established. Generalized Benders Decomposition is the means by which the MINLP may be solved via a combination of established NLP and MILP algorithms. The algorithm involves solving a series of MILP approximations of the MINLP. In these approximations the MILP objective function always underestimates the MINLP objective function and the feasible region of the MILP approximations always overestimates the extent of the MINLP feasible region.

An important consideration in any optimization procedure is the method by which the optimality of a solution is to be assessed. In the Generalized Benders Decomposition two problems are involved: a primal problem and a relaxed dual or "master problem". When the Generalized Benders Decomposition is used for MINLP problems the primal problem involves optimization over the continuous variables whereas the relaxed dual is an optimization problem with integer search variables. The relaxed dual problem is constructed in such a way that its minimum is a lower bound on the global optimum for the MINLP problem. On the other hand the primal problem, being simply a restricted version of the MINLP problem, provides an upper bound on the global minimum. Denoting the upper and lower bounds on the MINLP minimum as P^U and D^L respectively, a measure of the optimality of a solution is given by the difference $P^U - D^L$. The termination criterion for the MINLP algorithm is $P^U - D^L \leq \epsilon$, where ϵ is the optimality tolerance.

To facilitate the discussion, the MINLP is written in the general form:

$$\begin{aligned} &\min_{\substack{x\in X\\ y\in Y}} f(x,y), \\ &\text{subject to } g(x,y) \leq 0, \end{aligned} \tag{6.2}$$

where x and y are column vectors of decision variables, $X = \Re^{n_1}$ contains the set of feasible continuous variables, $Y = B^{n_2}$ where B^{n_2} is the set of n_2-dimensional binary variables. The Generalized Benders Decomposition is based on three key ideas:

1. projection of the problem from the xy-space onto the y-space;

2. representation of the projected problem as a dual problem;

3. relaxation of the dual problem to a MILP problem.

These concepts are discussed in turn.

6.3.1 Projection

Problem 6.2 may be *projected* onto y, resulting in the equivalent formulation:

$$\begin{aligned} & \min_y v(y) \\ \text{subject to } y &\in Y \cap V, \\ V &\equiv \{y : g(x,y) \leq 0, \quad \text{for some } x \in X\} \\ \text{where } v(y) &= \min_{x \in X} f(x,y) \\ \text{subject to } g(x,y) &\leq 0 \end{aligned} \tag{6.3}$$

This projection results in a problem involving minimization over the integer variables only. Unfortunately this problem is intractable as the objective function is implicitly defined as a minimization problem and the set V is not explicit. The strategy adopted in the Generalized Benders Decomposition is to construct an approximation of problem 6.3 in such a way that the objective function and constraints are explicit. The dual representation discussed below is the framework within which the approximation is made.

6.3.2 Dual Representation

Suppose the primal problem defined as:

$$\begin{aligned} & \min_{x \in X} f(x, y^k), \\ \text{subject to} \quad & g(x, y^k) \leq 0, \end{aligned}$$

is *convex* in x for all fixed y^k. Then, provided a strictly feasible solution to this problem exists, $\{x \in X : g(x,y^k) < 0\} \neq \emptyset$, $v(y)$ may be represented in the *dual* form:

$$v(y) = \max_{\{\lambda \in \Re^m | \lambda \geq 0\}} \left\{ \inf_{x \in X} \left\{ f(x,y) + \lambda^T g(x,y) \right\} \right\}.$$

The weak duality theorem yields the following result when x is feasible in the primal and λ is feasible in the dual [24]:

$$\inf_{x \in X} \left\{ f(x,y^k) + \lambda^T g(x,y^k) \right\} \qquad \leq \qquad f(x,y^k).$$

This result is immediately apparent from the following inequalities:

$$\inf_{x \in X} \left\{ f(x, y^k) + \lambda^T g(x, y^k) \right\} \leq f(x, y^k) + \lambda^T g(x, y^k) \leq f(x, y^k).$$

The relation between the primal and dual underlies the development of a MILP problem that approximates and underestimates the MINLP problem. A second relation between the primal and dual problems, which holds when the primal fulfills certain conditions, is the strong duality theorem:

$$\begin{aligned} \max_{\{\lambda \in \Re^m | \lambda \geq 0\}} \left\{ \inf_{x \in X} \left\{ f(x, y^k) + \lambda^T g(x, y^k) \right\} \right\} \quad &= \quad \min_x f(x, y^k) \\ &\text{subject to } g(x, y^k) \leq 0. \end{aligned}$$

This result underlies the assumption that a series of lower bounds on the minimum, D^L, defined by a series of MILP problems, can be made to approach the upper bound, P^U, to within an arbitrarily small tolerance ϵ.

The set V, which is the projection of the xy-space feasible region onto the y-space, can be represented in dual form in a similar manner so that a point y^k will be in the feasible set V if the following condition is satisfied:

$$\max_{\{\mu \in \Re^m | \mu \geq 0\}} \left\{ \inf_{x \in X} \left\{ \mu^T g(x, y^k) \right\} \right\} \leq 0$$

Using these dual forms the MINLP problem can be written as:

$$\begin{aligned} & \min_{\xi, y \in Y} \xi & & (6.4) \\ \text{subject to } \xi \quad &\geq \quad \inf_{x \in X} \left\{ f(x, y) + \lambda^T g(x, y) \right\}, & & \text{for all } \lambda \geq 0 \\ 0 \quad &\geq \quad \inf_{x \in X} \left\{ \mu^T g(x, y) \right\} & & \text{for all } \mu \geq 0. \end{aligned}$$

Although this minimization problem is still implicitly defined by constraints, the framework allows relaxations to be made which yield explicit MILP approximations.

6.3.3 Relaxation

Problem 6.4 is intractible for the following reasons:

- the constraints are implicitly defined as infima,
- there are an infinite number of these constraints, one for each $\lambda \geq 0$ and $\mu \geq 0$.

This section explores the relaxation of the infinite set of constraints to a finite one, and the use of problem structure to render the implicit constraints explicit. The following problem is a relaxed approximation of formulation 6.4 :

$$\begin{aligned}
&\min_{\xi, y \in Y} \xi && && (6.5)\\
&\text{subject to } \xi \;\geq\; L^{\text{feas}}(y; x^j, \lambda^j), && \lambda^j \geq 0, \quad j \in J^{\text{feas}} \\
&0 \;\geq\; L^{\text{infeas}}(y; x^j, \lambda^j), && \mu^j \geq 0, \quad j \in J^{\text{infeas}}, \\
&\text{where } L^{\text{feas}}(y; x^j, \lambda^j) \;=\; \inf_{x \in X} \left\{ f(x,y) + (\lambda^j)^T g(x,y) \right\}, && (6.6)\\
&L^{\text{infeas}}(y; x^j, \lambda^j) \;=\; \inf_{x \in X} \left\{ (\mu^j)^T g(x,y) \right\}. && (6.7)
\end{aligned}$$

In this problem the constraints remain implicit. A tractable formulation of these constraints can be made by appealing to the structure of the problem. The chemical equilibrium problem's MINLP representation allows $g(x,y)$ to be written as:

$$g(x,y) = Cx + Dy, \quad C \in \Re^m \times \Re^{n_1}, \; D \in \Re^m \times \Re^{n_2},$$

and $f(x,y)$ to be written as $f(x)$, independent of the y variable. Lagrangians L^{feas} and L^{infeas} then become:

$$\begin{aligned}
L^{\text{feas}}(y; x^j, \lambda^j) &= \inf_{x \in X} \left\{ f(x) + (\lambda^j)^T Cx + (\lambda^j)^T Dy \right\} \\
&= (\lambda^j)^T Dy + \inf_{x \in X} \left\{ f(x) + (\lambda^j)^T Cx \right\} \\
&= (\lambda^j)^T Dy + f(x^j) + (\lambda^j)^T Cx^j \\
\text{where } x^j &= \arg\min_{x \in X} \left\{ f(x) + (\lambda^j)^T Cx \right\}, \text{ for } \; j \in J^{\text{feas}}, \\
L^{\text{infeas}}(y; x^j, \lambda^j) &= \inf_{x \in X} \left\{ (\mu^j)^T Cx + (\mu^j)^T Dy \right\} \\
&= (\mu^j)^T Dy + \inf_{x \in X} \left\{ (\mu^j)^T Cx \right\} \\
&= (\mu^j)^T Dy + (\mu^j)^T Cx^j \\
\text{where } x^j &= \arg\min_{x \in X} \left\{ (\mu^j)^T Cx \right\}, \qquad \text{for } \; j \in J^{\text{infeas}}.
\end{aligned}$$

Consequently, problem 6.5 can be expressed as:

$$\begin{aligned}
&\min_{\xi, y \in Y} \xi \\
&\text{subject to } \xi \;\geq\; f(x^j) + \left(\lambda^j\right)^T Cx^j + \left(\lambda^j\right)^T Dy, && \lambda^j \geq 0, \quad j \in J^{\text{feas}} \\
&0 \;\geq\; (\mu^j)^T Cx^j + (\mu^j)^T Dy && \mu^j \geq 0, \quad j \in J^{\text{infeas}}. \\
&\text{where } x^j \;=\; \arg\min_{x \in X} \left\{ f(x) + (\lambda^j)^T Cx \right\}, \text{ for } \; j \in J^{\text{feas}}, \\
&x^j \;=\; \arg\min_{x \in X} \left\{ (\mu^j)^T Cx \right\}, \qquad \text{for } \; j \in J^{\text{infeas}}.
\end{aligned}$$

This observation fulfills Geoffrion's "property P" which requires that the infimum of $f(x,y)+(\lambda^j)^T g(x,y)$ and $(\mu^j)^T g(x,y)$ over X be taken independently of y [25]. Without this property the infima defining L^{feas} and L^{infeas} cannot be expressed as explicit functions of y.

A MINLP algorithm based on the Generalized Benders Decomposition uses a NLP procedure to perform the minimization over X to generate the constraints in problem 6.5 , and a MILP procedure to solve the linear approximation of the MINLP problem. Such an algorithm is described below.

6.4 MINLP Algorithm for the Chemical and Phase Equilibrium Problem

To clarify the relation between the theory discussed in the previous sections of this chapter and the MINLP algorithm for the chemical and phase equilibrium problems, the structure of these problems is repeated below for comparison:

General Problem	Equilibrium Problem
$\min_{\substack{x\in X \\ y\in Y}} f(x,y),$	$\min_{\substack{n\in X \\ y,yp\in Y}} G(n,y,yp),$
subject to $Cx + Dy \leq 0$	subject to $y_{ik} - yp_k \leq 0 \quad \forall k \in P, i \in C^k$
	$n_{ik} - \Omega_i y_{ik} \leq 0 \quad \forall k \in P, i \in C^k$
$C \in \Re^m \times \Re^{n_1}$,	
$D \in \Re^m \times \Re^{n_2}$	
$X = \Re^{n_1}$	$X = \{n_{ik} \mid An - b = 0,$
	$n_{ik} \geq 0, n_{ik} \in \Re \; \forall k \in P, i \in C^k\}$
$Y = B^{n_2}$	$Y = \{y_{ik}, yp_k \mid y_{ik}, yp_k \in \{0,1\} \quad \forall k \in P, i \in C^k\}$

A diagramatic representation of the MINLP algorithm is presented in figure 6.2:

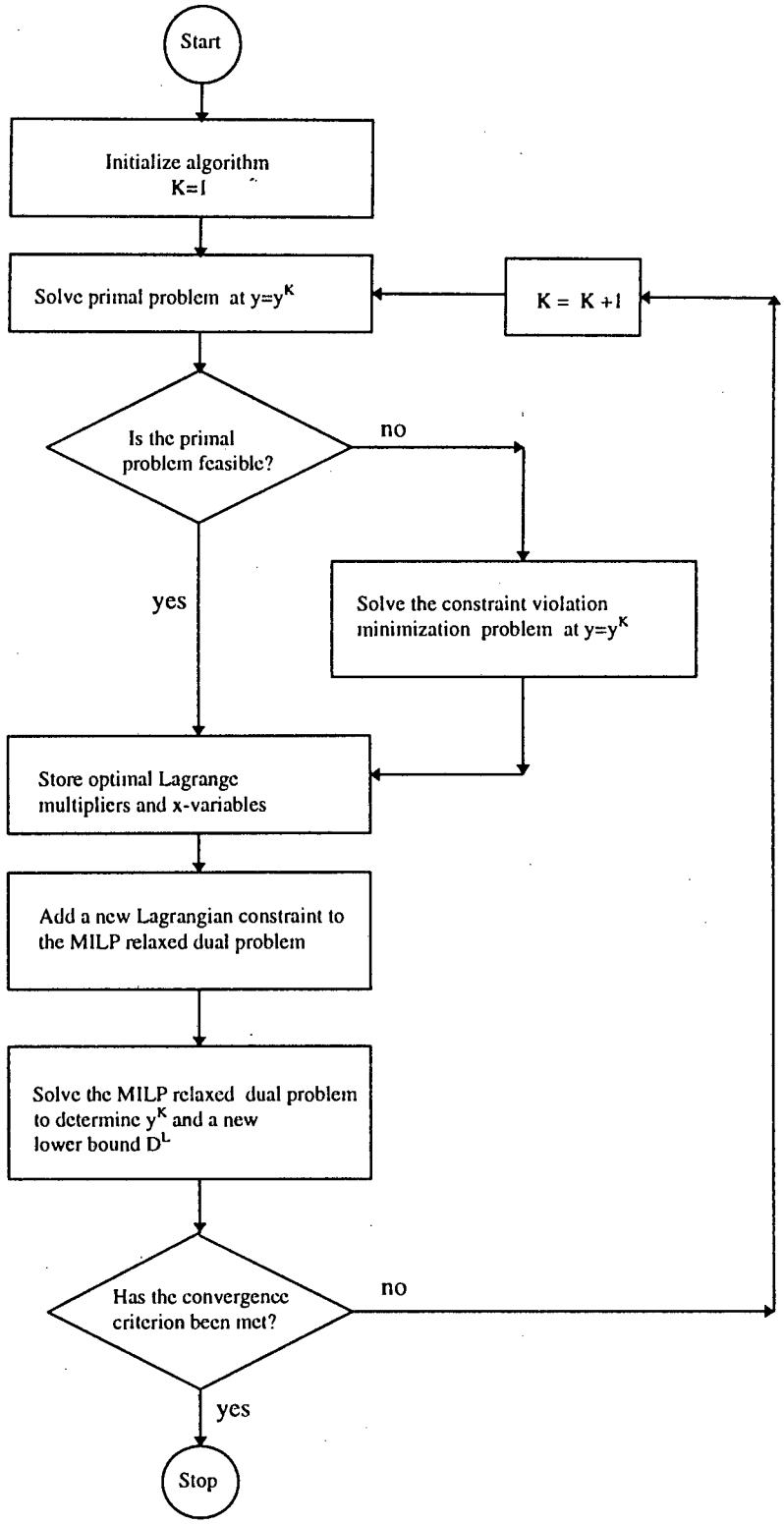

Figure 6.2: Mixed integer nonlinear programming algorithm

Step 1: Initialization

Set the lower bound on the minimum: $D^L = -\infty$.

Set the upper bound on the minimum: $P^U = +\infty$.

Set the iteration counter: $j = 1$.

Set a feasible choice of integer variables: y^j, yp^j.

Step 2a: The Primal Problem

Solve the following nonlinear program which is the primal problem:

$$\begin{aligned} \min_{n} & G \\ \text{subject to } An - b &= 0 \\ n_{ik} &\geq 0 \qquad \forall k \in P, i \in C^k \\ n_{ik} - \Omega_i y^j_{ik} &\leq 0 \qquad \forall k \in P, i \in C^k. \end{aligned}$$

if this problem is feasible:

let $J^{\text{feas}} = j \cup J^{\text{feas}}$;

store the optimal x-variables, $n^j = n$;

store the optimal Lagrange multipliers, $\lambda^j = \lambda$;

update the upper bound, $P^U = \min\{G(n^j), P^U\}$;

go to **step 3**;

else if this problem is not feasible:

go to **step 2b**.

end if

Step 2b: Infeasible Primal Problem

Solve the NLP that involves minimizing the violation of constraints:

$$\begin{aligned} \min_{n} & \alpha \\ \text{subject to } An - b &= 0 \\ n_{ik} &\geq 0 \qquad \forall k \in P, i \in C^k \\ n_{ik} - \Omega_i y^j_{ik} - \alpha &\leq 0 \qquad \forall k \in P, i \in C^k. \end{aligned}$$

Let $J^{\text{infeas}} = j \cup J^{\text{infeas}}$;

Store the optimal x-variables, $n^j = n$;

Store the optimal Lagrange multipliers, $\mu^j = \mu$.

Step 3: Relaxed Dual Problem

Solve the mixed integer linear programming master problem:

$$\begin{aligned} \min_{\xi,y,yp} \ & \xi \\ \text{subject to } \xi &\geq L^{\text{feas}}(y, yp, n^j, \lambda^j), \qquad j \in J^{\text{feas}} \\ 0 &\geq L^{\text{infeas}}(y, yp, n^j, \mu^j), \qquad j \in J^{\text{infeas}} \\ yp_k &\geq y_{ik}, \qquad \forall k \in P, i \in C^k \end{aligned}$$

Update the lower bound: $D^L = \xi$.

Step 4: Convergence Check

if the lower bound equals the upper bound on the optimum, $D^L = P^U$ then

stop,

else increment j;

set y^j to y, the minimizer of the relaxed dual problem;

return to **step 2.**

end if

6.5 Implementation

Theoretical and practical considerations in the implementation of the MINLP are discussed in this section.

6.5.1 Singularities in the Gibbs Energy Function

The MINLP reformulation of problem 6.1 requires some modification for implementation. Despite the ability of the species and phases to be explicitly activated or deactivated in the MINLP framework, the MINLP algorithm can still suffer from the numerical difficulties that can occur when a species tends to vanish in the computation of the NLP primal problem. To avoid this, the non-negativity constraints on the species are replaced by the constraints:

$$n_{ik} \geq \varepsilon y_{ik} \qquad k \in P, i \in C^k,$$

where ε is a small positive number that sets the lower bound on the quantity of an activated species. As the lower bound for the deactivated species is set to zero by this equation a

further adaptation is required to prevent singularities in the objective function. Following the approach taken by Paules and Floudas [57] this problem is resolved by partially generating the objective function to include only the terms that involve active species. The objective function now becomes:

$$\begin{aligned} G &= \sum_{k\in P}\sum_{i\in C_*^k} n_{ik}\mu_{ik}, \\ \text{where } C_*^k &\equiv C^k \cap \{i : y_{ik} = 1\}. \end{aligned}$$

In this definition the set C_*^k contains the activated components of the set C^k.

6.5.2 Integer Cuts

To prevent a primal problem defined by a particular set of binary parameters from being solved more than once, additional integer constraints are introduced into the relaxed dual problems:

$$\begin{aligned} &\sum_{k\in P}\sum_{i\in C^k}\left[sign\left(y_{ik}^j - \frac{1}{2}\right)\right] y_{ik} + \sum_{k\in P}\left[sign\left(yp_k^j - \frac{1}{2}\right)\right] yp_k \\ \leq\ &\sum_{k\in P}\sum_{i\in C^k} y_{ik}^j + \sum_{k\in P} yp_k^j - 1 \qquad\qquad \forall\ j \in J^{\text{feas}} \cup J^{\text{infeas}}. \end{aligned}$$

Note that for each j this inequality can be satisfied if and only if at least one y_{ik} (yp_k) is different from y_{ik}^j (yp_k^j). An example will illustrate this point:

Example 6 (Illustration of integer cuts) *Suppose that on iteration 1 the primal problem is defined with binary parameters:* $y_{11}^1 = 1$, $y_{21}^1 = 0$, *and* $yp_1^1 = 1$. *Then an integer cut constraint is:*

$$\left[sign\left(y_{11}^1 - \frac{1}{2}\right)\right] y_{11} + \left[sign\left(y_{21}^1 - \frac{1}{2}\right)\right] y_{21} + \left[sign\left(yp_1^1 - \frac{1}{2}\right)\right] yp_1 \leq y_{11}^1 + y_{21}^1 + yp_1^1 - 1$$

$$\left[sign\left(1 - \frac{1}{2}\right)\right] y_{11} + \left[sign\left(0 - \frac{1}{2}\right)\right] y_{21} + \left[sign\left(1 - \frac{1}{2}\right)\right] yp_1 \leq 1 + 0 + 1 - 1$$

$$[+]\, y_{11} + [-]\, y_{21} + [+]\, yp_1 \leq 1$$

Notice that only when $y_{11} = 1$, $y_{21} = 0$ *and* $yp_1 = 1$, *is the above constraint not satisfied. Using this constraint the combination of integer variables* $\{y^1, yp^1\}$ *can be excluded from further consideration.*

6.5.3 Programming Language

The algorithm was implemented using a version of the modelling and optimization package GAMS. GAMS is an acronym for " General Algebraic Modelling System " and was developed to facilitate the representation and optimization of models in a way that is simpler and more efficient than methods that make use of the programming languages such as C or FORTRAN [10].

The implementation of the MINLP algorithm in GAMS was done using the APROS (Algorithms for PROcess Synthesis) methodology proposed by Paules and Floudas [57]. This methodology facilitates the implementation of nonstandard algorithms involving communication of data between subproblems having parameters and structures that may vary during the course of the algorithm. Although the APROS methodology can be used with GAMS to implement nonstandard algorithms, programming tasks which are straightforward in a language like FORTRAN become extremely cumbersome in GAMS. The GAMS code for the MINLP algorithm can be found in appendix A.

6.5.4 Optimization Packages

The optimization algorithm GAMS/MINOS was used for the solution of the nonlinear programming subproblems. For the mixed integer linear programming master problem the GAMS/ZOOM algorithm was used. GAMS/ZOOM solves the master problem by first solving it as a linear program, after which the Pivot and Complement heuristic is used to find the initial integer feasible solution. A Branch and Bound procedure is then employed to search for improved solutions and to verify optimality[10].

6.6 Numerical Results

The example used to demonstrate the MINLP approach to multiphase reaction equilibrium problems involves the reduction of iron oxide in a system comprised of three solid phases and a gas phase. This example from Balzhiser et al. [2] has also been solved by Castillo and Grossmann [12], and Paules and Floudas [57]. The feed conditions and thermodynamic data used are shown in table 6.1 and have been taken directly from Balzhiser et al. As in the aforementioned papers, it is assumed here that the solid phases are immiscible and activity models are not necessary to describe their behaviour under the temperature and pressure conditions specified. In addition, the gas has been assumed to behave ideally. This problem

Component	Feed [mols]	$\frac{G^o}{RT}$@1366K [dimensionless]
Fe(s)	0.0000	0.00
FeO(s)	1.0000	-8.53
C(s)	2.0000	0.00
CO	0.7500	-11.30
CO_2	0.0000	-19.40
H_2	0.7500	0.00
O_2	0.5000	0.00
H_2O	0.0000	-8.39

Table 6.1: Data for Iron Oxide Reduction at 1atm

Species	MINLP Case 1	MINLP Case 2	MINLP [57]	NLP	NLP [2]	NLP [12]
Fe(s)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
FeO(s)	eliminated	eliminated	eliminated	1E-10	eliminated	1E-10
C(s)	0.1086	1.4414	0.1087	0.1086	0.1086	0.0091
CO	2.5625	eliminated	2.5720	2.5625	2.5625	2.7349
CO_2	0.0789	1.3086	0.0787	0.0789	0.0789	0.0061
H_2	0.7203	0.0617	0.7201	0.7203	0.7203	0.7470
O_2	eliminated	eliminated	1.7833E-9	1E-10	0.2962E-9	1E-10
H_2O	0.0297	0.1328	0.0296	0.0297	0.0297	0.0030

Table 6.2: Comparison of Solutions to the Iron Oxide Reduction Example

was solved by the MINLP algorithm using two different initial guesses:

Case 1 y_{ik}^1 and yp_k^1 initialized to 1 for all species and all phases;

Case 2 the y-variable for the species CO set to 0, all other y_{ik}^1 and yp_k^1 initialized to 1.

The solutions obtained are shown in table 6.2

The results obtained from the MINLP algorithm with initial guess 1 are very similar to those reported by Paules and Floudas [57]. The elimination of O_2 in this implementation is due to the lower bound on the species, ε, being set to 10^{-4}. O_2 was not eliminated by Paules and Floudas as their value of ε was 10^{-9}. The MINLP algorithm with the second initial guess returned an erroneous solution, CO being eliminated. The partial generation of the

objective function results in the NLP solver in the primal problem not being able to compute the Lagrange multiplier for the constraint:

$$n_{ik} - \Omega_i y_{ik}^j \leq 0,$$

where i represents CO. As the dual information, used to construct the constraints in the relaxed dual problem, is incorrect, so is the relaxed representation of the dual problem. A solution to this problem is to set a lower bound ϵ, on all species, including the deactivated ones, and completely generate the Gibbs energy function in the primal problem. Formulated in this way, however, the MINLP offers no advantage over a NLP formulation. The fourth column of table 6.2 shows the results obtained from the solution to the NLP formulation using GAMS/MINOS. These results are in agreement with the results reported by Balzhiser et al [2] and Paules and Floudas [57], but not with those reported by Castillo and Grossmann [12]. This discrepancy is possibly due to Castillo and Grossmann's calculations being for the system at 1363K instead of 1366K.

6.7 Conclusions

The MINLP formulation is inappropriate for the phase and chemical equilibrium problem. Integer variables are redundant in the MINLP formulation as the disappearance of a species can occur through the continuous elimination of moles. Furthermore, the MINLP formulation is able to eliminate neither the objective function singularities nor the ill-conditioning of the reduced Hessian. A more suitable approach for dealing with a lack of *a priori* information about the number of phases present at equilibrium would be a nonlinear programming approach, customized to suit the chemical equilibrium problem.

Chapter 7

Decomposition Based Global Optimization

In some instances more than one local optimum may occur on the Gibbs energy surface. Of these local minima, the one which globally minimizes the Gibbs energy surface is the true equilibrium point; the other minima may physically be interpreted as metastable states which are of little interest in engineering applications. The methods discussed in the previous chapters are all based on iterative approaches where at each iteration local Gibbs energy surface phenomena such as the gradient and curvature are used to determine an improved estimate of the minimum. Applied to nonconvex surfaces these methods can at best guarantee convergence to a local minimum but cannot guarantee in any way the global optimality of the solution obtained. This section examines a method capable of guaranteeing the global optimality of the solution. The method applies specifically to cases where the nonconvexities in the Gibbs energy function are induced by the NRTL activity coefficient model. An efficiency enhancing modification of the algorithm is introduced, and the modified algorithm is compared with the original.

7.1 Decomposition Based Global Optimization Algorithm

The algorithm described in this section was developed by Visweswaran and Floudas [20] and applied to the chemical and phase equilibrium problem by McDonald and Floudas [44, 45]. This deterministic method, is inspired by the decomposition techniques employed in the Generalized Benders Decomposition [25] which was discussed in the previous chapter. The crucial difference, with regard to the Generalized Benders Decomposition, between the type

of nonconvex problem discussed in this chapter and the MINLP problems discussed in the previous chapter is the presence of Geoffrion's "property P" [25] in the latter and its absence in the former. In the previous chapter the presence of "property P" allowed the relaxed dual problem to be formulated in an explicit manner. The global optimization algorithm discussed in this chapter deviates from the Generalized Benders Decomposition primarily in the construction of a computable relaxed dual problem in the absence of "property P". The global optimization algorithm is suitable for minimization problems that conform to the following general structure:

$$\begin{aligned} \min_{x,y} f(x,y) & & (7.1) \\ \text{subject to } g(x,y) &\leq 0 \\ h(x,y) &= 0 \\ x &\in X \\ y &\in Y \end{aligned}$$

where X and Y are convex sets, $f(x,y)$, $g(x,y)$ and $h(x,y)$ are continuous piecewise differential functions on $X \times Y$ which satisfy the following criteria:

1. $f(x,y)$ and $g(x,y)$ are convex in x for all fixed y, and convex in y for all fixed x;
2. $h(x,y)$ is affine in x for all fixed y, and affine in y for fixed x;
3. X and $Y \subseteq V$ are nonempty, compact convex sets and the Slater constraint qualification is satisfied:
$$V \equiv \{y : h(x,y) = 0, g(x,y) \leq 0, \text{ for some } x\};$$
4. $\frac{\partial f}{\partial x_i}$ and $\frac{\partial g}{\partial x_i}$ are affine in y for fixed x.

Nonconvexities in problems of this type arise through the interaction between the x and the y variables. A simple example of a nonconvex optimization problem, fulfilling the above requirements is:

$$\begin{aligned} \min_{x\in\Re, y\in\Re} & \quad xy \\ \text{subject to } -2 &\leq x \leq 1 \\ -1 &\leq y \leq 2. \end{aligned}$$

The objective function for this problem is depicted in figure 7.1, and is clearly not convex.

While some problems may naturally conform to the above structure, others may be made to conform through the augmentation, transformation and partitioning of the original problem's variables. These ideas will be demonstrated in the application of the global optimization algorithm to the chemical equilibrium problem.

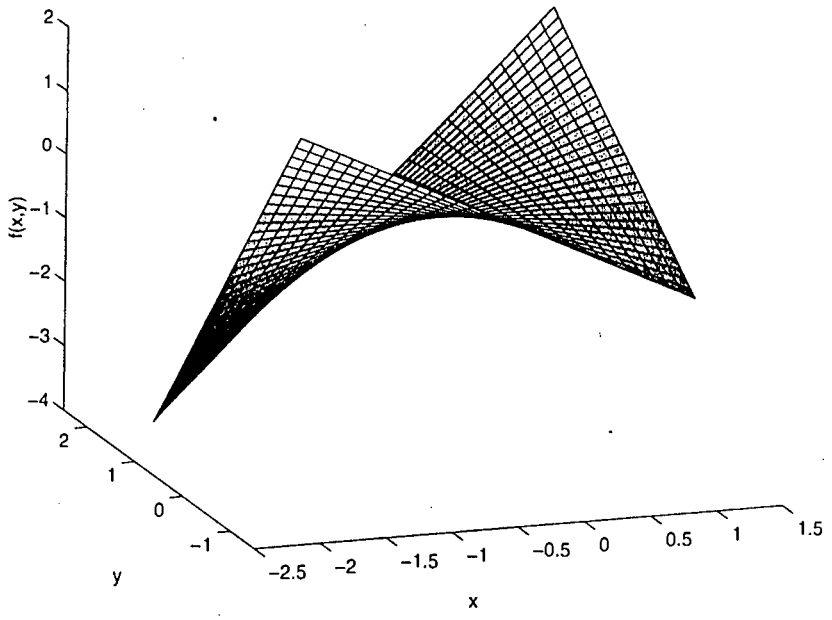

Figure 7.1: Nonconvexity of a bilinear function

7.2 Decomposition and Duality

An important issue in global optimization is that of estimating the quality of a candidate global solution. A common measure of global optimality, used by McDonald and Floudas [44] as a convergence criterion, is the difference between the candidate minimum and a tight lower bound on the global optimum. The global optimization algorithm uses the two key concepts of the Generalized Benders Decomposition to generate this lower bound:

1. *decomposition*, part of the strategy used to unravel the complicating effects of the interaction between x and y variables;
2. *duality*, used to take into account the effects of constraints in the construction of a lower bound.

Problem 7.1 may be decomposed into the following problem involving inner and outer subproblems:

$$\min_y v(y) \tag{7.2}$$

$$\begin{aligned} \text{subject to } y &\in Y \cap V, \\ \text{where} \quad V &\equiv \{y : h(x,y) = 0, g(x,y) \leq 0, \text{for some } x\} \qquad (7.3) \\ & \qquad (7.4) \\ \text{and} \quad v(y) &\equiv \min_x f(x,y) \\ & \qquad \text{subject to } h(x,y) = 0, \\ & \qquad\qquad g(x,y) \leq 0, \end{aligned}$$

In this formulation $v(y)$ and V are not explicitly defined. Using the Generalized Benders Decomposition approach, Visweswaran and Floudas [20] use duality to explicitly define these quantities. Define the *primal* problem parametrized by a fixed value $y^k \in Y$:

$$\begin{aligned} \min_x f(x,y^k) \quad &, \qquad (7.5) \\ \text{subject to } h(x,y^k) &= 0, \\ g(x,y^k) &\leq 0. \end{aligned}$$

Conditions 1-4 of section 7.1 ensure that this problem is *convex* and satisfies the Slater constraint qualification. Therefore the strong duality theorem applies; the solutions to the primal and dual problems are the same [24], with the dual problem given by:

$$\max_{\mu \geq 0, \lambda} \left\{ \inf_{x \in X} \left\{ f(x,y^k) + \lambda^T h(x,y^k) + \mu^T g(x,y^k) \right\} \right\}, \quad y^k \in Y \cap V.$$

Problem 7.2 can therefore be represented by replacing the inner problem with the dual formulation:

$$\begin{aligned} & \min_y v(y), \qquad (7.6) \\ \text{subject to } v(y) &\geq \min_x \left\{ f(x,y) + \lambda^T h(x,y) + \mu^T g(x,y) \right\}, \quad \forall \mu \geq 0, \forall \lambda, \\ y &\in Y \cap V, \\ V &\equiv \{y : h(x,y) = 0, g(x,y) \leq 0, \text{ for some } x\}. \end{aligned}$$

This problem has an infinite number of constraints and the function $v(y)$ is implicitly defined. By dropping the constraints on y and replacing the infinite set of multipliers with a finite one, a relaxed dual formulation with a finite number of constraints is obtained:

$$\begin{aligned} & \min_y v(y), \qquad (7.7) \\ \text{subject to } v(y) &\geq \min_x \left\{ f(x,y) + \left(\lambda^k\right)^T h(x,y) + \left(\mu^k\right)^T g(x,y) \right\} \quad k = 1, \ldots, K. \end{aligned}$$

As this relaxed problem has fewer constraints than 7.6 it represents a lower bound on the global solution. This result is not useful in itself as the inner problem is parametrized

by y and would appear to be very difficult to solve. A computable approximation of the intractable relaxed dual problem which can rigorously underestimate the global minimum is therefore required for the development of an implementable algorithm. In the Generalized Benders Decomposition "property P" is called upon at this stage to enable the constraint functions to be expressed explicitly. In general, the nonconvex optimization problem is such that the interaction of the x- and y- type variables invalidate property P. Visweswaran and Floudas's [20] method of rendering a computable approximation for the relaxed dual problem is discussed in the next section.

7.3 Simplification of the Relaxed Dual

Discussed in section 7.3.1 is the way in which problem 7.7, when constrained by a *single* Lagrangian function, can be simplified to an explicit form. Section 7.3.2 generalizes the concepts introduced in section 7.3.1 to include the case where there is *more than one* Lagrangian constraint in problem 7.7.

7.3.1 Simplification of the Relaxed Dual Constrained by a Single Lagrangian Function

Let the inner relaxed dual problem be defined as:

$$\begin{aligned} & \min_x L(x,\tilde{y},\lambda^k,\mu^k), \\ L(x,\tilde{y},\lambda^k,\mu^k) &= f(x,\tilde{y}) + (\lambda^k)^T h(x,\tilde{y}) + (\mu^k)^T g(x,\tilde{y}), \end{aligned}$$

where $L(x,\tilde{y},\lambda^k,\mu^k)$ is the Lagrangian function parametrized by $\tilde{y}$ and Lagrange multipliers λ^k and μ^k. In this section we consider the transformation of the inner relaxed dual minimization problem into a set of constraints which define a computable lower bound. Consider the Lagrangian function which has been linearized with respect to the x variables about a point x^k :

$$L(x,\tilde{y},\lambda^k,\mu^k)\,|^{\text{lin}}_{x^k} = L(x^k,\tilde{y},\lambda^k,\mu^k) + \sum_{i\in I} \nabla_{x_i} L(x,\tilde{y},\lambda^k,\mu^k)\,|_{x^k} \cdot (x_i - x_i^k),$$

and note that this linearized function underestimates the Lagrangian because of the latter's convexity with respect to x.

From this observation an inequality between the inner relaxed dual and a linearized inner relaxed dual problem obtains:

$$\min_x L(x,\tilde{y},\lambda^k,\mu^k) \geq \min_x L(x,\tilde{y},\lambda^k,\mu^k)\,|^{\text{lin}}_{x^k}\,. \tag{7.8}$$

Clearly, the minimizer of the linearized problem is an extreme point, where each x_i is equal to either its lower bound x_i^L, or its upper bound x_i^U, depending on whether $\nabla_{x_i} L(x, \tilde{y}, \lambda^k, \mu^k) \mid_{x^k}$ is positive or negative:

$$\begin{aligned}
&\nabla_{x_i} L(x, \tilde{y}, \lambda^k, \mu^k) \mid_{x^k} \leq 0 \Rightarrow x_i^{\min} = x_i^U \\
&\nabla_{x_i} L(x, \tilde{y}, \lambda^k, \mu^k) \mid_{x^k} \geq 0 \Rightarrow x_i^{\min} = x_i^L, \\
&\text{where} \quad x_i^{\min} = \arg\min_x L(x, \tilde{y}, \lambda^k, \mu^k) \mid_{x^k}^{\text{lin}}, \\
&\text{and } \tilde{y} \text{ is an arbitrary fixed value of } y.
\end{aligned}$$

Using these results, the following problem can be used to underestimate the solution to the relaxed dual:

$$\begin{aligned}
&\min_{B_j \in CB, \xi \in \Re} \xi, \qquad (7.9) \\
&\text{subject to } \xi \geq \min_{y \in Y} L(x^{B_j}, y, \lambda^k, \mu^k) \mid_{x^k}^{\text{lin}}, \\
\text{subject to } &\nabla_{x_i} L(x, y, \lambda^k, \mu^k) \leq 0, \quad \text{if } x_i = x_i^U \\
&\nabla_{x_i} L(x, y, \lambda^k, \mu^k) \geq 0, \quad \text{if } x_i = x_i^L,
\end{aligned}$$

Where B_j refers to a combination of lower and upper bounds, CB is the set of all such combinations, and x^{Bj} is a vector defining the extreme point on the x-domain which corresponds to B_j.

To clarify this notation let x be a 2-vector. The set CB then contains 4 elements:

1. $B_1 = \{L, L\}$,
2. $B_2 = \{L, U\}$,
3. $B_3 = \{U, L\}$,
4. $B_4 = \{U, U\}$,

where L and U refer to "lower bound" and "upper bound", and $x^{B_2} = (x_1^L, x_2^U)$.

Minimization over CB can be achieved by solving the inner minimization over y for each $B_j \in CB$. The constraints imposed on the inner problem, termed *qualifying constraints* by Visweswaran and Floudas, result in the y-domain being partitioned into subdomains, where each subdomain is associated with a particular element of CB. Note that satisfaction of condition 4 in section 7.1 results in the feasible region defined by the qualifying constraints being convex. This simple example based on figure 7.1 will clarify the idea:

Example 7 *Demonstration of the use of qualifying constraints for the nonconvex minimization problem:*

$$\begin{aligned} \min_{x\in\Re, y\in\Re} f(x,y) &= xy \\ \textit{subject to } -2 &\leq x \leq 1 \\ -1 &\leq y \leq 2. \end{aligned}$$

Consider this problem's relaxed "dual":

$$\min_{B_l \in CB} \min_{\xi\in\Re} \xi$$

$$\textit{subject to } \xi \geq \begin{cases} \min\limits_{-1\leq y\leq 2} f(x^{B_l}, y), \\ \\ \textit{subject to } y \leq 0, \quad \textit{if } x^{B_l} = x^U \\ \qquad\qquad y \geq 0, \quad \textit{if } x^{B_l} = x^L \end{cases},$$

noting that, because this example involves no constraints where x and y variables interact, there is no notion of duality in this relaxed "dual" problem, hence the "relaxed dual problem" will be referred to as the "relaxed problem". As x is a scalar, the set CB contains only two elements, B_1 and B_2, where $x^{B_1} = x^L$, and $x^{B_2} = x^U$. In figure 7.2 the domain of the original minimization problem is divided into two regions. In region 1 the minimization problem is restricted to the domain where $y \geq 0$, in region 2 the minimization problem is restricted to the domain where $y < 0$. This is the partitioning scheme dictated by the constraint qualifications. Notice that the objective function in region 1 increases with increasing x, while the function in region 2 decreases with increasing x. From this observation it follows that the minimizer of region 1 must lie at $x = x^L = -2$ and the minimizer of region 2 must lie at $x = x^U = 1$. The partitioning of the y domain into two regions demarcated by the qualifying constraints results in a nonconvex minimization problem being replaced by two convex minimization subproblems:

$$\textit{Problem } B_1\textit{:} \min_{0\leq y\leq 2} f(x^L, y),$$

and

$$\textit{Problem } B_2\textit{:} \min_{-1\leq y\leq 0} f(x^U, y).$$

Feasible regions for these problems are marked by arrows labelled B_1 and B_2 respectively and the minimizers for the respective problems are $y = 2$ and $y = -1$. As the minimum of problem B_1 is less than that of problem B_2 the solution to the relaxed problem is the solution to problem B_1. Because $f(x,y)$ is affine in x no linearization approximation has been made, hence the solution to the relaxed problem is the global minimizer of the original problem.

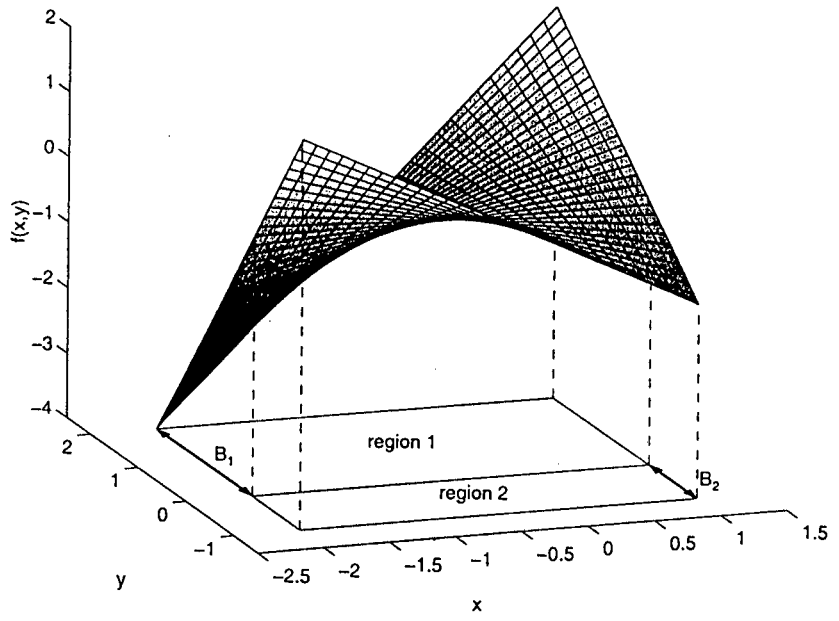

Figure 7.2: Qualifying constraints for the bilinear programming problem

7.3.2 Simplification of the Relaxed Dual Constrained by Multiple Lagrangian Functions

The previous section illustrated how the relaxed dual problem 7.7 can be simplified to a form with explicit constraints when there is only one Lagrangian constraint. This section will show how the methods used in the previous section can be used to simplify a relaxed dual with any number of Lagrangian constraints.

The representation of a relaxed dual problem with multiple constraints is a crucial element of the global optimization algorithm. The lower bound given via problem 7.9 will yield a conservative bound due to the relaxation of the dual problem and the linearization of the Lagrangians with respect to the x-variables. To generate a sequence of lower bounds that converges to a value D^L such that $f(x^*,y^*) - D^L \leq \epsilon$, where x^*,y^* is the best known solution and ϵ is a predefined convergence tolerance, provision must be made for refinement of an initially crude lower bound estimate. This is done through finer partitioning of the y-domain and the addition of constraints to the relaxed dual problem.

Consider the situation on the K^{th} iteration of an algorithm where $K-1$ Lagrangian functions are available from previous iterations' relaxed dual problems in addition to a newly generated Lagrangian function. The relaxed dual problem 7.7 may be defined as:

$$\begin{aligned} & \min_{y\in Y,\xi} \xi \\ \text{subject to} \quad \xi &\geq \min_x L(x,y,\lambda^k,\mu^k) \qquad k=1,...,K-1 \\ \xi &\geq \min_x L(x,y,\lambda^K,\mu^K). \end{aligned} \tag{7.10}$$

Simplification of this problem to a tractable one is done in a manner similar to that considered in the previous section, except here the Lagrangians from previous iterations need to be included. A central aspect to the development of this simplification is the manner in which the qualifying constraints partition the y-domain over successive iterations. The following example illustrates the partitioning process and introduces the concepts which allow 7.10 to be simplified.

Example 8 Illustration of the partitioning of the y-domain.

Consider the case where $y \in \Re^2$ and the qualifying constraints cause the y-domain to be partitioned by arbitrary hyperplanes. Let there be a Lagrangian function $L(x,y,\lambda^1,\mu^1)$ whose qualifying constraints are such that the y-domain is partitioned in the manner depicted in figure 7.3. In this diagram the four arrows labelled B_i^1 point into the y-subdomains where the linearized Lagrangian functions $L(x^{B_i^1},y,\lambda^1,\mu^1)\,|_{x^1}^{lin}$ parametrized by $x^{B_i^1}$, respectively under-

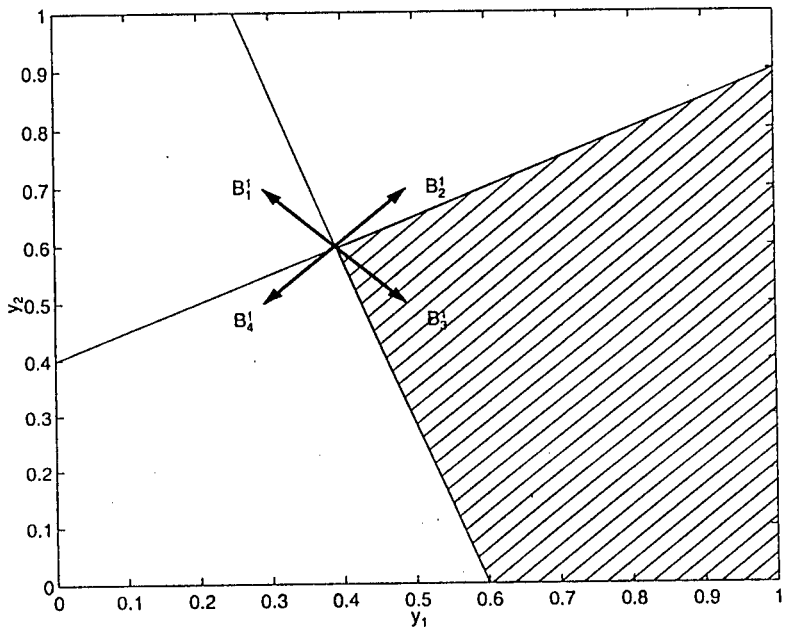

Figure 7.3: Partitioning of the y-domain for Lagrangian 1

estimate the Lagrangian $L(x,y,\lambda^1,\mu^1)$. *Note that the linearized Lagrangian* $L(x^{B_3^1},y,\lambda^1,\mu^1)\mid_{x^1}^{lin}$ *is a valid underestimator of the Lagrangian only in the shaded* y*-subdomain. A lower bound on the global optimum can be computed by first solving a minimization subproblem for each* $j=1,\ldots,4$*:*

$$\min_{\xi^{B_j^1}\in\Re} \xi^{B_j^1},$$

$$\text{subject to } \xi^{B_j^1} \geq \begin{cases} \min\limits_{y\in Y} L(x^{B_j^1},y,\lambda^1,\mu^1)\mid_{x^k}^{lin}, \\ \text{subject to } \nabla_{x_i} L(x,y,\lambda^1,\mu^1)\leq 0, \quad \text{if } x_i^{B_j^1} = x_i^{U_j^1} \\ \qquad\qquad \nabla_{x_i} L(x,y,\lambda^1,\mu^1)\geq 0, \quad \text{if } x_i^{B_j^1} = x_i^{L_j^1}, \end{cases}$$

then finding the minimum of these:

$$\min\left\{\xi^{B_1^1},\xi^{B_2^1},\xi^{B_3^1},\xi^{B_4^1}\right\}.$$

Now consider figure 7.4 which illustrates the partitioning of the y*-domain for a Lagrangian function* $L(x,y,\lambda^2,\mu^2)$. *The arrows in this diagram point into the regions where the linearized Lagrangians* $L(x^{B_i^2},y,\lambda^2,\mu^2)\mid_{x^2}^{lin}$ *underestimate the Lagrangian function* $L(x,y,\lambda^2,\mu^2)$. *The shaded region represents the region where the linearized Lagrangian* $L(x^{B_1^2},y,\lambda^2,\mu^2)\mid_{x^2}^{lin}$ *underestimates* $L(x,y,\lambda^2,\mu^2)$. *In figure 7.5 the shaded region represents the intersection of the shaded regions of figure 7.3 and figure 7.4. In this region* $L(x^{B_3^1},y,\lambda^1,\mu^1)\mid_{x^1}^{lin}$ *underestimates* $L(x,y,\lambda^1,\mu^1)$, *and* $L(x^{B_1^2},y,\lambda^2,\mu^2)\mid_{x^2}^{lin}$ *underestimates* $L(x,y,\lambda^2,\mu^2)$. *Now consider*

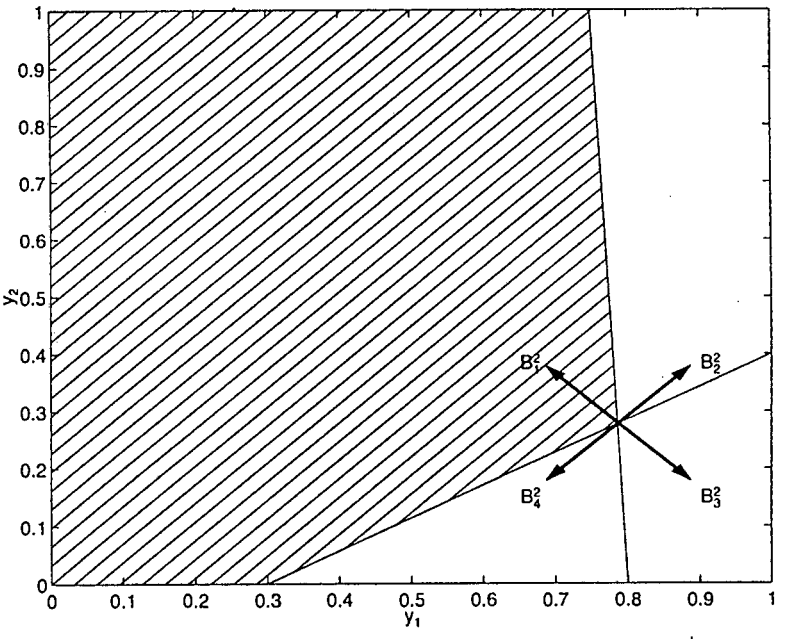

Figure 7.4: Partitioning of the y-domain for Lagrangian 2

the implication of these results for a problem which is similar to problem 7.10:

$$\begin{aligned} & \min_{y \in Y', \xi} \xi \\ \textit{subject to} \quad \xi & \geq \min_x L(x, y, \lambda^1, \mu^1) \\ \xi & \geq \min_x L(x, y, \lambda^2, \mu^2), \end{aligned} \tag{7.11}$$

where Y' is the shaded region in figure 7.5. From the above discussion a lower bound on the minimum of this problem is given by the minimum to the following problem:

$$\begin{aligned} & \min_{y \in Y', \xi} \xi \\ \textit{subject to} \quad \xi & \geq L(x^{B_3^1}, y, \lambda^1, \mu^1) \mid_{x^1}^{lin} \\ \xi & \geq L(x^{B_1^2}, y, \lambda^2, \mu^2) \mid_{x^2}^{lin} \end{aligned} \tag{7.12}$$

As this problem has explicit constraints and is convex it is amenable to computation. Having seen how 7.10 can be simplified to one region Y' of the y-domain we now consider the problem of simplifying the relaxed dual over a larger region, the shaded region Y'' in figure 7.3. By noting that $L(x^{B_3^1}, y, \lambda^1, \mu^1) \mid_{x^1}^{lin}$ is a valid underestimator of $L(x, y, \lambda^1, \mu^1)$ in part of regions B_2^2 and B_4^2 and throughout region B_3^2, we can underestimate the minimum of the problem:

$$\begin{aligned} & \min_{y \in Y'', \xi} \xi \\ \textit{subject to} \quad \xi & \geq \min_x L(x, y, \lambda^1, \mu^1) \\ \xi & \geq \min_x L(x, y, \lambda^2, \mu^2), \end{aligned} \tag{7.13}$$

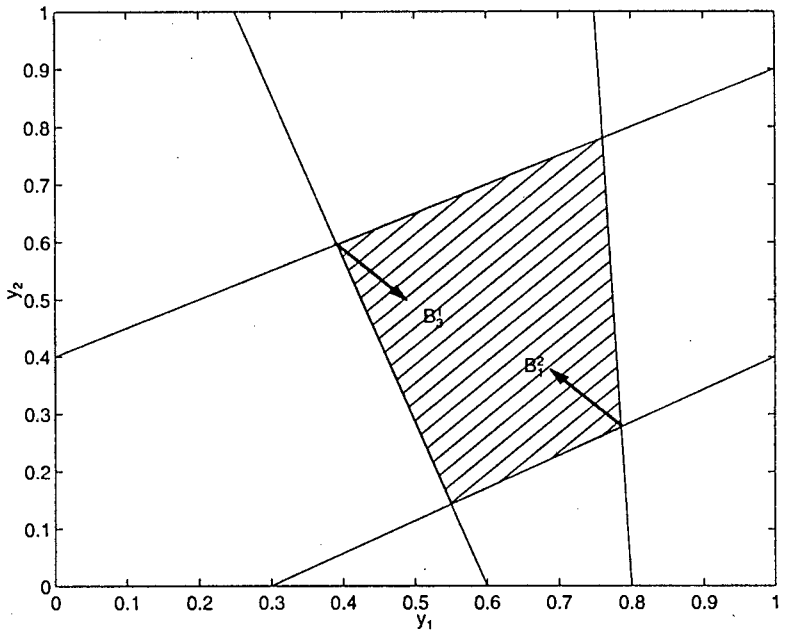

Figure 7.5: Partitioning of the y-domain for Lagrangians 1 and 2

with the minimum of the following problem:

$$\min_{B_j^2 \in CB} \left\{ \xi^{B_j^2} \right\} \tag{7.14}$$

$$\text{where } \xi^{B_j^2} = \begin{cases} \min\limits_{y \in Y'', \xi} \xi \\ \text{subject to} \quad \xi \geq L(x, y, \lambda^2, \mu^2) \mid_{x^2}^{lin} \\ \nabla_{x_i} L(x, y, \lambda^k, \mu^k)\Big|_{x^k} \leq 0 \qquad \text{if } x_i^{B_j^2} = x_i^{U_j^2} \\ \nabla_{x_i} L(x, y, \lambda^k, \mu^k)\Big|_{x^k} \geq 0, \qquad \text{if } x_i^{B_j^2} = x_i^{L_j^2} \\ \xi \geq L(x^{B_3^1}, y, \lambda^1, \mu^1) \mid_{x^1}^{lin} \quad . \end{cases}$$

Let $\xi^{B^2_{\min}}$ denote the solution to the above problem. A lower bound D^L on the global minimum over the whole y-domain can be obtained from:

$$D^L = \min \left\{ \xi^{B^2_{\min}}, \xi^{B_1^1}, \xi^{B_2^1}, \xi^{B_4^1} \right\}. \tag{7.15}$$

Note that if $\xi^{B_3^1}$ was to be included in the minimization problem in relation 7.15, the expression would still be valid. However, a tighter bound on the global optimum is derived using the expression as it stands, because $\xi^{B^2_{\min}}$, which is greater than or equal to $\xi^{B_3^1}$, underestimates the global minimum in the same region of the y-domain which is underestimated by $\xi^{B_3^1}$. In equation 7.15 , $\xi^{B_1^1}, \xi^{B_2^1}$ and $\xi^{B_4^1}$ are each uniquely associated with a region that has not yet been further partitioned. To generalize the idea behind equation 7.15 it is convenient to associate each y-region with a node which has a unique identity number $1, \ldots, k_s$. The set

of all nodes associated with y-regions that have not been further partitioned is denoted N_{k_s}. *Equation 7.15 can then be expressed as:*

$$D^L = \min_{k_t \in N_{k_s}} \left\{ \xi^{B^2_{\min}}, \xi^{k_t} \right\}.$$

In the global optimization algorithm a new Lagrangian function is defined on every iteration. The new Lagrangian, together with Lagrangians from previous iterations are then combined, in the manner described in this example, to tighten the lower bound on the global optimum. The Lagrangian functions $L(x, y, \lambda^1, \mu^1)$ *and* $L(x, y, \lambda^2, \mu^2)$ *can be seen as the Lagrangians whose multipliers where calculated on iterations 1 and 2 respectively. At the end of the first iteration the lower bound on the global minimum is* $\min\left\{\xi^{B^1_1}, \xi^{B^1_2}, \xi^{B^1_3}, \xi^{B^1_4}\right\}$. *At the end of the second iteration this lower bound is refined to* $\min\left\{\xi^{B^2_{\min}}, \xi^{B^1_1}, \xi^{B^1_2}, \xi^{B^1_4}\right\}$.

From these observations we can derive a computable approximation of the K^{th} iteration's relaxed dual problem. Let the Lagrangian function for the k^{th} iteration be defined by the Lagrange multipliers corresponding to the optimal solution of the primal problem 7.5 where y is fixed at y^k. Define $PL(k, K)$ to be the set of bounds $B^k_j \in CB$ for which the qualifying constraints for the Lagrangian function from the k^{th} iteration are satisfied at y^K. The simplified version of problem 7.10 is:

$$\min\left\{\xi^{B^K_{\min}}, \min_{k_t \in N_{k_s}} \xi^{k_t}\right\}$$

where $\xi^{B^K_{\min}} = \min_{B^K_l \in CB} \xi^{B^K_l}$;

and $\xi^{B^K_l} =$

$$\left\{ \begin{array}{c} \min_{\xi} \xi \\ \text{subject to previous iterations' Lagrangian constraints:} \\ \left. \begin{array}{ll} \xi \geq L(x, y, \lambda^k, \mu^k)\,|^{\text{lin}}_{x^k} & \\ \nabla_{x_i} L(x, y, \lambda^k, \mu^k)\Big|_{x^k} \leq 0, & \text{if } x_i^{B^k_j} = x_i^{U^k_j} \\ \nabla_{x_i} L(x, y, \lambda^k, \mu^k)\Big|_{x^k} \geq 0, & \text{if } x_i^{B^k_j} = x_i^{L^k_j}; \end{array} \right\} \forall B^k_j \in PL(k, K), \quad k = 1, .., K \\ \\ \text{and the current iteration's Lagrangian constraint:} \\ \begin{array}{ll} \xi \geq L(x, y, \lambda^K, \mu^K)\,|^{\text{lin}}_{x^K} & \\ \nabla_{x_i} L(x, y, \lambda^K, \mu^K)\Big|_{x^K} \leq 0, & \text{if } x_i^{B^K_l} = x_i^{U^K_l} \\ \nabla_{x_i} L(x, y, \lambda^K, \mu^K)\Big|_{x^K} \geq 0, & \text{if } x_i^{B^K_l} = x_i^{L^K_l}, \end{array} \end{array} \right.$$

7.4 The Global Optimization Algorithm

Figure 7.6 gives a diagrammatic overview of the global optimization algorithm.

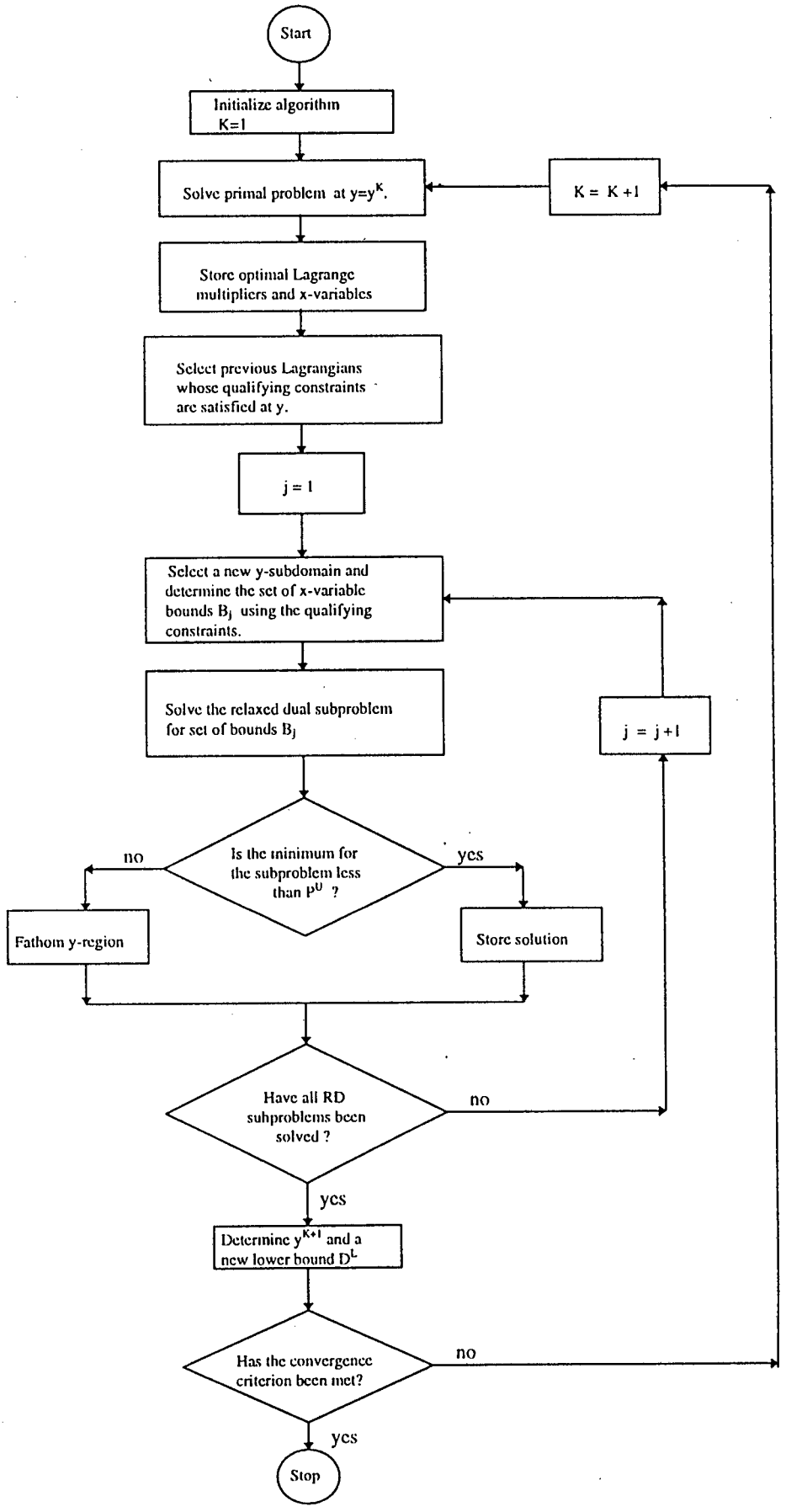

Figure 7.6: Global optimization algorithm

The general form of the global optimization algorithm may be formally stated as:

Step 0: Initialization

Set:

the lower bound on the global minimum: $D^L = -\infty$

the upper bound on the global minimum: $P^U = +\infty$

the convergence tolerance: ϵ

the iteration counter: $K = 1$

the starting point: y^1

Step 1: The Primal Problem

Solve the primal problem with $y = y^K$, and set P^K to the optimal objective value;

Evaluate the Lagrange multipliers for the primal problem, λ^K;

If $P^K < P^U$ update the upper bound on the global solution: $P^U = P^K$.

Step2: Select Previous Lagrangians

Select the set of Lagrangians from previous iterations which are pertinent to the forthcoming relaxed dual problem:

for $k = 1, \ldots, K-1$

for all $B_l^k \in CB$

Determine whether or not the qualifying constraints evaluated at y^K are satisfied for the set of bounds B_l^k, in other words check if the following relations are true:

$$\nabla_{x_i} L(x, y^K, \lambda^k, \mu^k)\Big|_{x^k} \leq 0, \qquad \text{if } x_i^{B_l^k} = x_i^{U_l^k}$$
$$\nabla_{x_i} L(x, y^K, \lambda^k, \mu^k)\Big|_{x^k} \geq 0, \qquad \text{if } x_i^{B_l^k} = x_i^{L_l^k}.$$

If these relations are satisfied then let $PL(k, K) = B_l^k$.

end

end

Step 3: The Relaxed Dual Phase

Set up and solve the relaxed dual subproblem corresponding to each bound combination:

for all $B_l^K \in CB$

Calculate $x^{U_l^K}, x^{L_l^K}$, the upper and lower bounds on the x-variables;

Solve the following relaxed dual subproblem for $\xi^{B_l^K}$ and y^* where y^* is the minimizer and $\xi^{B_l^K}$ is the minimum:

$$\begin{aligned} &\min_{y,\xi} \xi \\ &\xi \geq L(x,y,\lambda^k,\mu^k)\,|^{\text{lin}}_{x^k} \\ \text{subject to} \quad &\nabla_{x_i} L(x,y,\lambda^k,\mu^k)\big|_{x^k} \leq 0, \quad \text{if } x_i^{B_j^k} = x_i^{U_j^k} \\ &\nabla_{x_i} L(x,y,\lambda^k,\mu^k)\big|_{x^k} \geq 0, \quad \text{if } x_i^{B_j^k} = x_i^{L_j^k} \end{aligned} \Bigg\} \quad \begin{aligned} &\forall \quad B_j^k \in PL(k,K), \\ &k = 1,..,K-1 \end{aligned}$$

$$\begin{aligned} \text{and} \quad &\xi \geq L(x,y,\lambda^K,\mu^K)\,|^{\text{lin}}_{x^K} \\ &\nabla_{x_i} L(x,y,\lambda^K,\mu^K)\big|_{x^K} \leq 0, \quad \text{if } x_i^{B_l^K} = x_i^{U_l^K} \\ &\nabla_{x_i} L(x,y,\lambda^K,\mu^K)\big|_{x^K} \geq 0, \quad \text{if } x_i^{B_l^K} = x_i^{L_l^K}, \end{aligned}$$

if $\xi^{B_l^K} \geq P^U$, then fathom the y-subregion;

if $\xi^{B_l^K} < P^U$, then set $k_s = k_s + 1$, add node k_s to the set of "leaf" nodes N_{k_s}, and store the solution to the subproblem; $\xi^{k_s} = \xi^{B_l^K}$, and $y^{k_s} = y^*$.

end

Step 4: Select the y-Region for Next Iteration

Update the lower bound on the global minimum, $D^L = \min_{k_t \in N_{k_s}} \xi^{k_t}$, and select the corresponding node for the next iteration, $k_c = \arg\min_{k_t \in N_{k_s}} \xi^{k_t}$.

Remove k_c from the set of leaf nodes, N_{k_s}.

Set $y^{K+1} = y^{k_c}$.

Step 5: Check for Convergence

If $\frac{P^U - D^L}{|P^U|} \leq \epsilon$ stop, the algorithm has converged; otherwise set $K = K + 1$ and return to Step 1.

Note that for the general problem, the possibility of the primal problem being infeasible must be taken into account. However, infeasible primal problems are not considered here as they do not occur when the algorithm is applied to the Gibbs energy minimization problem.

When the global optimization algorithm is applied to the NRTL equation the algorithm is structured to be more efficient, primarily in the means by which the previous Lagrangians are selected and in the use of the qualifying constraints in the relaxed dual subproblems. Conceptually, however, the general algorithm presented here is identical to the method which

is described in section 7.5 . The global optimization algorithm is demonstrated below on a problem involving the minimization of a nonconvex polynomial function.

Example 9 Demonstration of the Global Optimization Algorithm

Consider the following minimization problem:

$$\min_{-4\leq y\leq 6} -15y - 5y^2 + y^3.$$

Figure 7.7 shows the objective function, which is labelled $f(y)$, to be nonconvex. The nonconvexities are introduced in the y^2 and y^3 terms. As the structure of this problem does not match the prerequisites of the global optimization algorithm the problem needs to be modified in some way. One way of adapting the problem is through the introduction of new variables x_1 and x_2 which are related to y via the equations:

$$\begin{aligned} x_1 - y &= 0 \\ x_2 - x_1 y &= 0. \end{aligned}$$

With these variables the problem can be written as:

$$\begin{aligned} \min_{(x_1,x_2,y)\in\Re^3} \quad & -15x_1 - 5x_2 + x_2 y \\ \textit{subject to} \quad & -4 \leq y \leq 6 \\ & x_1 - y = 0 \\ & x_2 - x_1 y = 0. \end{aligned}$$

By partitioning the variables so that x_1,$x_2 \in X$ and $y \in Y$ the problem clearly matches the structure underlying the assumptions of the global optimization algorithm. In particular the objective function is convex in the x-variables when the y-variables are held constant, and vice versa. In addition, the equality constraints are affine in the x-variables when the y-variables are held constant and vice versa.

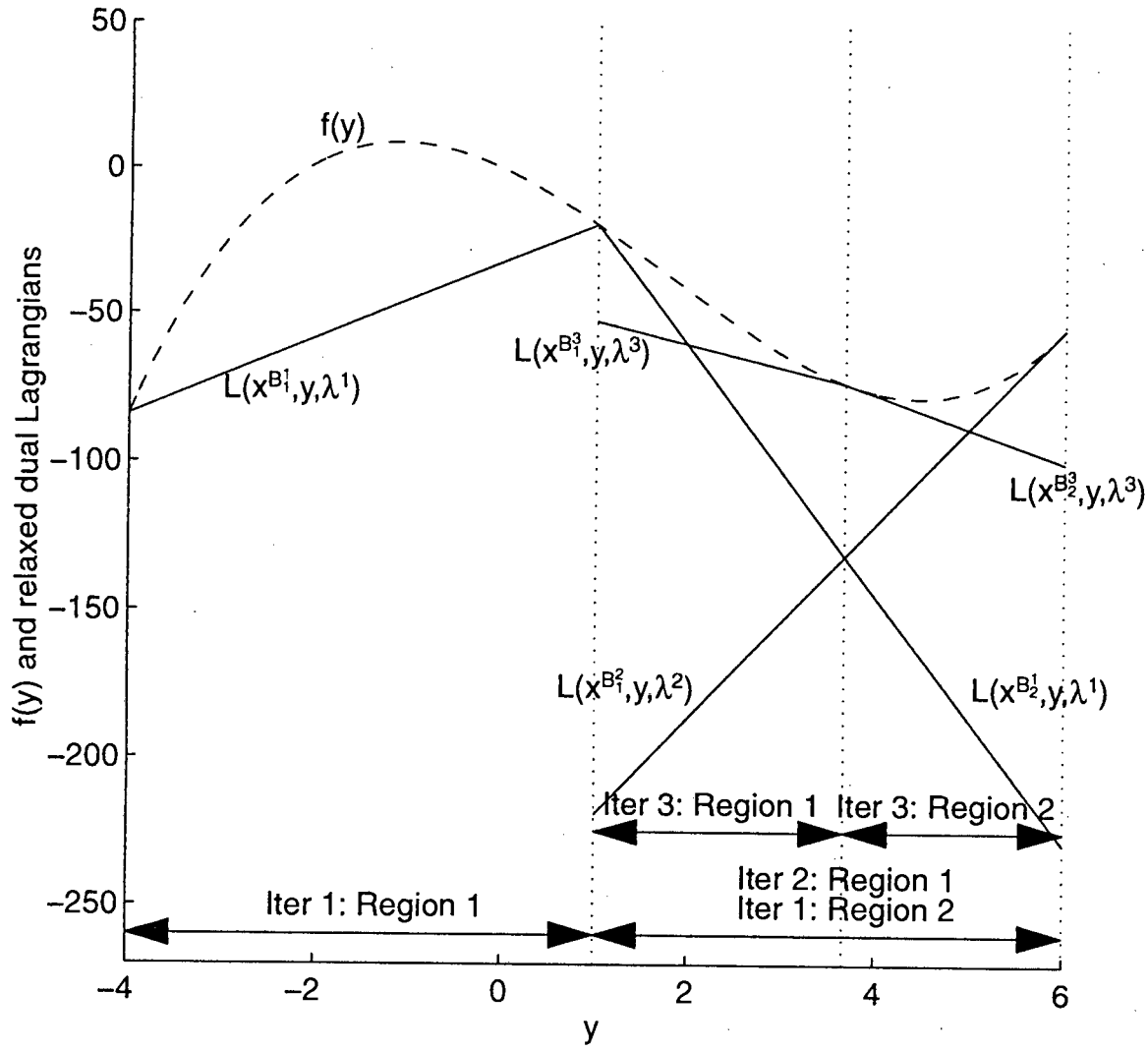

Figure 7.7: Global optimization of a polynomial function

Iteration 1: *To start the algorithm a feasible value* (y^1) *is selected. Let* $y^1 = 1$. *On iteration* k *the primal problem parametrized by* y^k *is:*

$$\begin{aligned} \min_{(x_1,x_2)\in\Re^2} \quad & -15x_1 - 5x_2 + x_2 y^k \\ \textit{subject to} \quad & x_1 - y^k = 0 \\ & x_2 - x_1 y^k = 0. \end{aligned}$$

The solution to the primal problem is completely defined by the equality constraints, yielding the optimal x*-values:*

$$\begin{aligned} x_1 &= y^k \\ x_2 &= (y^k)^2. \end{aligned}$$

Using our initial value of y^1 *we get the following result:*

$$\begin{aligned} x_1 &= 1 \\ x_2 &= 1. \end{aligned}$$

The primal objective function is evaluated here as -19. P^U, *the upper bound in the global optimum can therefore be set to* -19. *The Lagrangian function for the primal problem is written as:*

$$L(x, y^k) = -15x_1 - 5x_2 + x_2 y^k + \lambda_1^k \left(x_1 - y^k\right) + \lambda_2^k \left(x_2 - x_1 y^k\right),$$

where λ_1^k *and* λ_2^k *are the Lagrange multipliers for the equality constraints. From the Karush-Kuhn-Tucker conditions which include:*

$$\nabla_x L(x, y^k) = 0,$$

the following relations must hold at the solution to the primal problem:

$$\begin{aligned} -15 + \lambda_1^k - \lambda_2^k y^k &= 0 \\ -5 + y^k + \lambda_2^k &= 0. \end{aligned}$$

Using these equations the optimal Lagrange multipliers can be calculated. For the current problem where $y^1 = 1$ *the values of the Lagrange multipliers are:*

$$\begin{aligned} \lambda_1^1 &= 19 \\ \lambda_2^1 &= 4. \end{aligned}$$

The next stage in the algorithm is the solution of the relaxed dual problem which consists of two relaxed dual subproblems. Each subproblem is defined by its feasible y*-domain*

and the bounds at which the x-variables are held constant. The partitioning of the y-domain and the setting of the x-variables to their bounds ensures that the solution to the relaxed dual problem underestimates the global minimum. The following qualifying constraints are central to the specification of the subproblems:

$$\begin{aligned}
\frac{\partial L(x,y,\lambda^1)}{\partial x_1} &\leq 0 \quad \text{if } x_1^{B_i^1} = x_1^{U_i^1} \\
\frac{\partial L(x,y,\lambda^1)}{\partial x_1} &\geq 0 \quad \text{if } x_1^{B_i^1} = x_1^{L_i^1} \\
\frac{\partial L(x,y,\lambda^1)}{\partial x_2} &\leq 0 \quad \text{if } x_2^{B_i^1} = x_2^{U_i^1} \\
\frac{\partial L(x,y,\lambda^1)}{\partial x_2} &\geq 0 \quad \text{if } x_2^{B_i^1} = x_2^{L_i^1}.
\end{aligned}$$

In this example the superscripts on B, L, and U refer to the iteration number, while the subscripts refer to the relaxed dual subproblem number. These qualifying constraints can be simplified by taking into account the KKT conditions for the primal problem:

$$\begin{aligned}
\frac{\partial L(x,y,\lambda^k)}{\partial x_1} &= \left(-15 + \lambda_1^k - \lambda_2^k y\right) \\
&= \left(-15 + \lambda_1^k - \lambda_2^k y^k\right) + \left(-\lambda_2^k\right)\left(y - y^k\right) \\
&= -\lambda_2^k \left(y - y^k\right), \\
\frac{\partial L(x,y,\lambda^k)}{\partial x_2} &= \left(-5 + y + \lambda_2^k\right) \\
&= \left(-5 + y^k + \lambda_2^k\right) + \left(y - y^k\right) \\
&= y - y^k.
\end{aligned}$$

From the above result we note that the partitioning of the y-domain is such that in each subdomain the sign of $y - y^k$ is constant. In the present example the qualifying constraints define two regions:

$$\begin{aligned}
&\textit{region 1} \quad y \leq 1, \\
&\textit{region 2} \quad 1 \leq y
\end{aligned}$$

In the first region we note that :

$$\frac{\partial L(x,y,\lambda^1)}{\partial x_1} = -19(y-1) \geq 0,$$

and

$$\frac{\partial L(x,y,\lambda^2)}{\partial x_1} = y - 1 \leq 0.$$

As the value of the Lagrangian in this region increases with increasing x_1, the underestimating function is defined by setting x_1 to its lower bound. With increasing x_2 the

Lagrangian decreases, therefore the underestimating function is defined by setting x_2 *to its upper bound:*

$$x_1^{B_1^1} = x_1^{L_1^1}$$
$$x_2^{B_1^1} = x_2^{U_1^1}.$$

In the second region the following relation obtains:

$$\frac{\partial L(x,y,\lambda^1)}{\partial x_1} = -19(y-1) \leq 0,$$
$$\text{and} \qquad \frac{\partial L(x,y,\lambda^2)}{\partial x_1} = y - 1 \geq 0,$$

therefore the bounds on the x*-variables are set as follows:*

$$x_1^{B_2^1} = x_1^{U_1^1}$$
$$x_2^{B_2^1} = x_2^{L_2^1}.$$

To evaluate the numeric values of $x_1^{U_i^1}, x_1^{L_i^1}, x_2^{U_i^1}$, *and* $x_2^{L_i^1}$ *in each of the regions,* $i = 1, 2$, *we need to examine the constraints that pertain to these variables. In the first region defined by*

$$-4 \leq y \leq 1,$$

we note that

$$0 \leq (y)^2 \leq 16.$$

Therefore in region 1, $x_1^{U_1^1} = 1, x_1^{L_1^1} = -4, x_2^{U_1^1} = 16$, *and* $x_2^{L_1^1} = 0$. *In the second region defined by*

$$1 \leq y \leq 6,$$

the bounds on $(y)^2$ *are:*

$$1 \leq (y)^2 \leq 36.$$

Therefore in region 2, $x_1^{U_2^1} = 6$, $x_1^{L_2^1} = 1$, $x_2^{U_2^1} = 36$, *and* $x_2^{L_2^1} = 1$. *We can now define and solve the relaxed dual problems in each of the subregions.*

In region 1 the relaxed dual problem is defined as:

$$\min_{-4 \leq y \leq 1, \xi} \xi$$
$$\text{subject to} \quad \xi \geq -15x_1^{L_1^1} - 5x_2^{U_1^1} + x_2^{U_1^1} y + \lambda_1^1 \left(x_1^{L_1^1} - y \right) + \lambda_2^1 \left(x_2^{U_1^1} - x_1^{L_1^1} y \right).$$

This function is labelled $L(x^{B_1^1}, y, \lambda^1)$ *in figure 7.7. The minimizer to this problem lies at* $y = -4$ *where the relaxed dual objective has a value of -84.*

In region 2 the relaxed dual problem is defined as:

$$\begin{aligned} &\min_{1\le y\le 6,\xi} \xi \\ &\text{subject to} \quad \xi \geq -15x_1^{U_2^1} - 5x_2^{L_2^1} + x_2^{L_2^1}y + \lambda_1^1\left(x_1^{U_2^1} - y\right) + \lambda_2^1\left(x_2^{L_2^1} - x_1^{U_2^1}y\right). \end{aligned}$$

This function is labelled $L(x^{B_2^1}, y, \lambda^1)$ in figure 7.7 and its minimum in region 2 of iteration 1 lies at $y = 6$, where the relaxed dual objective has a value of -229. As $-229 \leq -84$, D^L is set to -229, and the parameter for the next primal problem is $y^2 = 6$.

Iteration 2: *The primal problem is solved using the parameter $y^2 = 6$ and the equations:*

$$\begin{aligned} x_1 &= y^2 = 6 \\ x_2 &= (y^2)^2 = 36. \end{aligned}$$

The primal objective value, -54, is less than $P^U = -19$ therefore the new upper bound $P^U = -54$. The Lagrange multipliers are calculated analytically:

$$\begin{aligned} -15 + \lambda_1^2 - \lambda_2^2 y^2 &= 0 \\ -5 + y^2 + \lambda_2^2 &= 0. \end{aligned}$$

$$\begin{aligned} \lambda_1^2 &= 9 \\ \lambda_2^2 &= -1 \end{aligned}$$

The qualifying constraints for the relaxed dual problems based on the parameters from iteration 2 are:

$$\begin{aligned} -\lambda_2^2\left(y - y^2\right) = 1(y-6) &\geq 0 \quad \text{if } x_1^{B_i^2} = x_1^{L_i^2}, \\ 1(y-6) &\leq 0 \quad \text{if } x_1^{B_i^2} = x_1^{U_i^2} \\ \text{and} \qquad y - y^2 = y - 6 &\geq 0 \quad \text{if } x_2^{B_i^2} = x_2^{L_i^2} \\ y - 6 &\leq 0 \quad \text{if } x_2^{B_i^2} = x_2^{U_i^2}. \end{aligned}$$

These constraints require that the y-domain be partitioned into the following regions:

$$\text{region 1} \quad y \leq 6 \tag{7.16}$$

$$\text{region 2} \quad 6 \leq y. \tag{7.17}$$

However, as the second region contains only a single point which is contained in the first region, this region is redundant. The qualifying constraints for iteration 1's Lagrangian

are repeated here:

$$\begin{aligned} -19(y-1) &\geq 0 \quad \text{if } x_1^{B_1^1} = x_1^{L_1^1}, && (7.18)\\ y-1 &\leq 0 \quad \text{if } x_1^{B_1^1} = x_1^{U_1^1} && (7.19)\\ -19(y-1) &\leq 0 \quad \text{if } x_1^{B_2^1} = x_1^{U_2^1}, && (7.20)\\ y-1 &\geq 0 \quad \text{if } x_1^{B_2^1} = x_1^{L_2^1}. && (7.21) \end{aligned}$$

Constraints 7.18 and 7.19 are satisfied in the region where $y \leq 1$; *constraints 7.20 and 7.21 are satisfied when* $y \geq 1$. *At* $y = y^2 = 6$ *constraints 7.20 and 7.21 are satisfied, therefore the constraint associated with region 2 of iteration 1 can be included in the current relaxed dual problem:*

$$\begin{aligned} &\min_{1\leq y\leq 6,\xi} \xi \\ \text{subject to } \xi &\geq -15x_1^{U_2^1} - 5x_2^{L_2^1} + x_2^{L_2^1} y + \lambda_1^1\left(x_1^{U_2^1} - y\right) + \lambda_2^1\left(x_2^{L_2^1} - x_1^{U_2^1} y\right) \\ \xi &\geq -15x_1^{U_1^2} - 5x_2^{U_1^2} + x_2^{U_1^2} y + \lambda_1^2\left(x_1^{U_1^2} - y\right) + \lambda_2^2\left(x_2^{U_1^2} - x_1^{U_1^2} y\right). \end{aligned}$$

These constraints are shown in figure 7.7 where they are labelled $L(x^{B_2^1}, y, \lambda^1)$ *and* $L(x^{B_1^2}, y, \lambda^2)$ *respectively. The minimizer is* $y = 3.67$ *and* $\xi = -131$. *As* $-131 \leq -84$, -131 *is the new lower bound on the global optimum* D^L, *and the parameter for iteration 3's primal problem is* $y^3 = 3.67$.

Iteration 3: *The primal problem is solved using the parameter* $y^3 = 3.67$ *and the equations:*

$$\begin{aligned} x_1 &= y^3 = 3.67 \\ x_2 &= (y^3)^2 = 13.44. \end{aligned}$$

The primal objective, -72.93, *is less than the current* P^U *therefore* $P^U = -72.93$. *The Lagrange multipliers are calculated analytically:*

$$\begin{aligned} -15 + \lambda_1^3 - \lambda_2^3 y^3 &= 0 \\ -5 + y^3 + \lambda_2^3 &= 0. \end{aligned}$$

$$\begin{aligned} \lambda_1^3 &= 19.89 \\ \lambda_2^3 &= 1.33 \end{aligned}$$

The qualifying constraints for the relaxed dual problems based on the parameters from iteration 2 are:

$$\begin{aligned} -\lambda_2^3\left(y - y^3\right) = 1.33(y - 3.67) &\geq 0 \quad \text{if } x_1^{B_i^3} = x_1^{L_i^3}, \\ 1.33(y - 3.67) &\leq 0 \quad \text{if } x_1^{B_i^3} = x_1^{U_i^3} \\ \text{and} \qquad y - y^3 = y - 3.67 &\geq 0 \quad \text{if } x_2^{B_i^3} = x_2^{L_i^3} \\ y - 3.67 &\leq 0 \quad \text{if } x_2^{B_i^3} = x_2^{U_i^3}. \end{aligned}$$

These constraints require that the y-domain be partitioned into the following regions

$$\begin{aligned} \textit{region 1} \qquad y &\leq 3.67 \qquad (7.22) \\ \textit{region 2} \qquad 3.67 &\leq y. \qquad (7.23) \end{aligned}$$

The qualifying constraints for Lagrangian functions $L(x^{B_2^1}, y, \lambda^1)$ and $L(x^{B_1^2}, y, \lambda^2)$ can be shown to hold at y^3 therefore the relaxed dual with region 2 of iteration 1 can be included in the current relaxed dual problem, which for region 1 of iteration 3 is formulated as:

$$\begin{aligned} &\min_{1 \leq y \leq 3.67, \xi} \xi \\ \textit{subject to } \xi &\geq L(x^{B_2^1}, y, \lambda^1) \\ \xi &\geq L(x^{B_1^2}, y, \lambda^2) \\ \xi &\geq L(x^{B_1^3}, y, \lambda^3) \end{aligned}$$

where

$$\begin{aligned} L(x^{B_1^3}, y, \lambda^3) &= -15x_1^{L_1^3} - 5x_2^{U_1^3} + x_2^{U_1^3} y + \lambda_1^3 \left(x_1^{L_1^3} - y \right) + \lambda_2^3 \left(x_2^{U_1^3} - x_1^{L_1^3} y \right) \\ x_1^{L_1^3} &= 1 \\ x_2^{U_1^3} &= 13.44. \end{aligned}$$

In figure 7.7 the minimum $\xi = -72.93$ can be seen to lie at $y = 3.67$. As ξ is greater than the upper bound $P^U = -84$, it is certain that the global minimum does not lie in the y-domain of iteration 3 region 1 and the region is therefore fathomed. The relaxed dual subproblem for region 2 of iteration 3 is defined as

$$\begin{aligned} &\min_{3.67 \leq y \leq 6, \xi} \xi \\ \textit{subject to } \xi &\geq L(x^{B_2^1}, y, \lambda^1) \\ \xi &\geq L(x^{B_1^2}, y, \lambda^2) \\ \xi &\geq L(x^{B_2^3}, y, \lambda^3) \end{aligned}$$

where the Lagrangian is defined as:

$$\begin{aligned} L(x^{B_2^3}, y, \lambda^3) &= -15x_1^{U_2^3} - 5x_2^{L_2^3} + x_2^{L_2^3} y + \lambda_1^3 \left(x_1^{U_2^3} - y \right) + \lambda_2^3 \left(x_2^{L_2^3} - x_1^{U_2^3} y \right) \\ x_1^{U_2^3} &= 6 \\ x_2^{L_2^3} &= 13.44. \end{aligned}$$

The minimum lies at $y = 4.97$, and $\xi = -87.77$. Refering to figure 7.7, the regions that have not been further partitioned are seen to be "iter1: region 1" , "iter 3: region

1", and "iter 3: region 2". Region 1 of iteration 3 has been fathomed, therefore the new lower bound on the global minimum D^L is the smaller of the underestimator of the global minimum in region 1 of iteration 1 (-84) and the underestimator of the global minimum in iteration 3's region 2 (-87.77). Consequently D^L =-87.77 and the parameter for iteration 4's primal is $y^4 = 4.97$.

Comments: *This process continues while the lower bound defined by the Lagrangians of the relaxed dual problems differs from the least upper bound defined by the primal problems. In figure 7.7, the global minimum for the problem can be seen to lie at $y = -4$. Although on the second iteration the primal problem is evaluated at this point, $y^2 = -4$, the main effort in the global optimization procedure is the refinement of the underestimating function to a degree which is sufficient to prove that a known local minimum is an ϵ-global minimum.*

7.5 Application to the NRTL Equation

The application of the global optimization algorithm to a particular problem cannot be done blindly, as is the case with local nonlinear programming packages, rather it requires consideration of the following issues:

1. transformation of the problem so that it conforms to the algorithm's requirements;
2. partitioning the variables into x- and y- types;
3. definition the algorithm's subproblems: the primal problem and the relaxed dual problem;
4. partitioning of the y variable space.

McDonald and Floudas' resolution of these issues for the NRTL equation and the phase equilibrium problem is discussed in this section.

7.5.1 Transformation and Partitioning

The dimensionless Gibbs energy function, $\hat{G} = \frac{G}{RT}$, defined using the NRTL equation is:

$$\hat{G}(n) = \sum_{i \in C} \sum_{k \in P} n_{ik} \left\{ \frac{G^f_{ik}}{RT} + \ln \left(\frac{n_{ik}}{\sum_{j \in C} n_{jk}} \right) \right\}$$

$$+\sum_{i\in C}\sum_{k\in P} n_{ik}\left\{\frac{\sum_{j\in C}\tau_{ji}\gamma_{ji}n_{jk}}{\sum_{j\in C}\gamma_{ji}n_{jk}}+\sum_{j\in C}\frac{\gamma_{ij}n_{jk}}{\sum_{j\in C}\gamma_{lj}n_{lk}}\left\{\tau_{ij}-\frac{\sum_{l\in C}\tau_{lj}\gamma_{lj}n_{lk}}{\sum_{l\in C}\gamma_{lj}n_{lk}}\right\}\right\},$$

where, C is the set of chemical species, P is the set of hypothetical phases, and τ and γ are NRTL parameters. McDonald and Floudas [44] show that the function can be significantly simplified to yield the following:

$$\hat{G}(n) = \sum_{k\in P} C_k + \sum_{k\in P}\sum_{i\in C} n_{ik}\left\{\sum_{j\in C}\frac{\gamma_{ij}\tau_{ij}n_{jk}}{\sum_{l\in C}\gamma_{lj}n_{lk}}\right\} \tag{7.24}$$

$$\text{where}\quad C_k = \sum_{i\in C} n_{ik}\left\{\frac{G^f_{ik}}{RT}+\ln\left(\frac{n_{ik}}{\sum_{j\in C}n_{jk}}\right)\right\}.$$

The terms in equation 7.24 to the left of the plus sign are all convex; whereas those to the right are nonconvex. Restructuring the problem to conform to the global optimization algorithm's prerequisites starts with the transformation of variables, and the partitioning of variables into x- and y-types. Recall that this partitioning should yield a function that is convex in x for fixed y; and convex in y for fixed x, and the gradient of this function with respect to x should be affine in y. To achieve this, new variables are introduced:

$$\Psi_{ik} = \frac{n_{ik}}{\sum_{j\in C}\gamma_{ji}n_{jk}} \qquad \forall\ i\in C,\ k\in P.$$

The transformed objective function now becomes:

$$\hat{G}(n) = \sum_{k\in P} C_k + \sum_{k\in P}\sum_{i\in C} n_{ik}\left\{\sum_{j\in C}\gamma_{ij}\tau_{ij}\Psi_{jk}\right\}.$$

Additional constraints now have to be introduced as a result of the transformation:

$$\Psi_{ik}\left\{\sum_{j\in C}\gamma_{ji}n_{jk}\right\} = n_{ik} \quad \forall\ i\in C,\ k\in P.$$

The result of the transformation is such that the objective function and constraints are convex in the n for fixed Ψ, and convex in Ψ for fixed n. In addition, the gradients of these functions with respect to n are affine in Ψ, and *vice versa*. By partitioning the variables so that the y-variable set contains the mole vector n and the x-variable set contains the new variables Ψ,

$$x \leftarrow \{\Psi_{ik}\} \qquad y \leftarrow \{n_{ik}\},$$

the prerequisites for the global optimization algorithm are fulfilled.

7.5.2 Primal Problem

In this application the primal problem 7.5, parametrized by a value of y fixed at $\bar{y}$, is:

$$\min_{\Psi_{ik}} \sum_{k\in P}\bar{C}_k + \sum_{k\in P}\sum_{i\in C}\bar{n}_{ik}\left\{\sum_{j\in C}\gamma_{ij}\tau_{ij}\Psi_{jk}\right\} \tag{7.25}$$

$$\text{subject to } \Psi_{ik}\left\{\sum_{j\in C}\gamma_{ji}\bar{n}_{jk}\right\} = \bar{n}_{ik} \quad \forall\ i \in C,\ k \in P.$$

The material balance and positivity constraints on the mole vector are excluded from the primal problem as these constraints involve only y-variables, which are fixed in the primal. These constraints will be encountered again when the relaxed dual problem is considered. A careful examination of the primal problem's constraints shows that the primal, in this case, involves no minimization because each Ψ_{ik} is completely defined by the appropriate equality constraint.

In addition to supplying an upper bound to the problem, an essential role of the primal problem is to provide the values of the Lagrange multipliers which parametrize the Lagrangians in the relaxed dual phase of the algorithm. Here we consider the calculation of these multipliers. The Lagrangian function for problem 7.25 is:

$$\begin{aligned} L(x,\bar{y},\lambda) \;=\; & \sum_{k\in P}\bar{C}_k + \sum_{k\in P}\sum_{i\in C}\bar{n}_{ik}\left\{\sum_{j\in C}\gamma_{ij}\tau_{ij}\Psi_{jk}\right\} \\ & + \sum_{i\in C}\sum_{k\in P}\lambda_{\Psi_{ik}}\left\{\Psi_{ik}\sum_{j\in C}\gamma_{ji}\bar{n}_{jk} - \bar{n}_{ik}\right\}, \end{aligned} \tag{7.26}$$

where $\lambda_{\Psi_{ik}}$ is the multiplier associated with the equality constraint defining Ψ_{ik}. Optimality in the primal problem means that the KKT optimality conditions must apply:

$$\nabla_{\Psi_{ik}} L(x,\bar{y},\lambda) = \sum_{j\in C}\bar{n}_{jk}\gamma_{ji}\tau_{ji} + \bar{\lambda}_{\Psi_{ik}}\sum_{j\in C}\gamma_{ji}\bar{n}_{jk} = 0 \qquad \forall\ i \in C,\ k \in P.$$

The Lagrange multipliers can then be calculated analytically:

$$\bar{\lambda}_{\Psi_{ik}} = -\frac{\sum_{j\in C}\bar{n}_{jk}\gamma_{ji}\tau_{ji}}{\sum_{j\in C}\gamma_{ji}\bar{n}_{jk}} \qquad \forall\ i \in C,\ k \in P.$$

7.5.3 The Relaxed Dual Problem and the y-Space Partition

While the primal problem determines upper bounds on the global solution, the relaxed dual problem provides the lower bounds. An important consideration in the design of the global optimization algorithm is the manner in which the relaxed dual problem's qualifying constraints partition the $y-$domain. In the general global optimization algorithm the y-space partitioning is defined implicitly via the qualifying constraints. This partitioning can be manipulated by altering the form of the Lagrangian function in such a way that the qualifying constraints in the relaxed dual subproblems can be replaced by simple bounds on the y-variables. When the qualifying constraints are in this form the task of identifying the correct set of Lagrangian equations for the relaxed dual subproblems is simplified.

The basic form of the Lagrangian function 7.26 can be recast through the rearrangement of terms:

$$L(\Psi, n, \bar{\lambda}) = \sum_{i\in C}\sum_{k\in P} \Psi_{ik} \left\{ \sum_{j\in C} \gamma_{ji} \left[\tau_{ji} + \bar{\lambda}_{\Psi_{ik}}\right] \left[n_{jk} - \bar{n}_{jk}\right] \right\} + \sum_{k\in P} C_k - \sum_{i\in C}\sum_{k\in P} n_{ik}\bar{\lambda}_{\Psi_{ik}} \quad (7.27)$$

The qualifying constraints which result in the partitioning of the y-domain are derived from the gradients of the Lagrangian:

$$\nabla_{\Psi_{ik}} L(\Psi, n, \bar{\lambda}) = \sum_{j\in C} \gamma_{ij} \left[\tau_{ji} + \bar{\lambda}_{\Psi_{ik}}\right] \left[n_{jk} - \bar{n}_{jk}\right]. \quad (7.28)$$

Recall that the partitioning of the y-space is achieved by constraining the above gradients to be either negative or positive, depending on the bounds at which the x-values are set. Equation 7.28 , however, is unsuitable as a qualifying constraint because the borders demarcating the subregions would be defined by equations such as:

$$\nabla_{\Psi_{ik}} L(\Psi, n, \bar{\lambda}) = 0.$$

Geometrically, these equations define arbitrary hyperplanes in the y-domain, and the regions they enclose are unstructured polytopes. The lack of structure in such a partitioning scheme would make the task of determining the correct qualifying constraints from previous iterations a difficult one. McDonald and Floudas developed a simpler partitioning scheme by augmenting the set of x-variables. This augmented set contains the variables:

$$\Psi_{ijk} \qquad i \in C, \quad j \in C, \quad k \in P.$$

Using these variables the Lagrangian becomes:

$$\begin{aligned} L(x, y, \bar{\lambda}) &= \sum_{i\in C}\sum_{k\in P} \Psi_{ijk} \left\{ \sum_{j\in C} \gamma_{ji} \left[\tau_{ji} + \bar{\lambda}_{\Psi_{ik}}\right] \left[n_{jk} - \bar{n}_{jk}\right] \right\} && (7.29) \\ &\quad + \sum_{k\in P} C_k - \sum_{i\in C}\sum_{k\in P} n_{ik}\bar{\lambda}_{\Psi_{ik}} && (7.30) \\ &= \sum_{i\in C}\sum_{k\in P} \left\{ \sum_{j\in C} \Psi_{ijk}\gamma_{ji} \left[\tau_{ji} + \bar{\lambda}_{\Psi_{ik}}\right] \left[n_{jk} - \bar{n}_{jk}\right] \right\} \\ &\quad + \sum_{k\in P} C_k - \sum_{i\in C}\sum_{k\in P} n_{ik}\bar{\lambda}_{\Psi_{ik}}. && (7.31) \end{aligned}$$

The gradients of the modified Lagrangian with respect to the modified set of x-variables are:

$$\nabla_{\Psi_{ijk}} L(x, y, \bar{\lambda}) = \gamma_{ij} \left[\tau_{ji} + \bar{\lambda}_{\Psi_{ik}}\right] \left[n_{jk} - \bar{n}_{jk}\right]. \quad (7.32)$$

Each qualifying constraint is now a function of a single y-variable, with the result that the qualifying constraints partition the y-domain into simple n-rectangles.

7.5.4 Selecting Previous Lagrangians

As explained in section 7.3.2, the relaxed dual problem includes Lagrangians from previous iterations whose qualifying constraints are satisfied in the current iteration's y-space. On iteration K of the general global optimization algorithm these Lagrangians are determined by evaluating the qualifying constraints of each Lagrangian at $\bar{n}^K$. As a large number of Lagrangians may exist in the latter stages of the iterative process, determining the valid Lagrangians may become time consuming. In this section, a more efficient method of finding valid Lagrangians is demonstrated.

The qualifying constraints for the Lagrangian from the $(k)^{th}$ iteration parametrized by $\Psi^{B_l^{(k)}}$ may be stated as follows:

$$\left.\begin{array}{lll} \gamma_{ij}\left[\tau_{ji}+\bar{\lambda}^{(k)}_{\Psi_{ik}}\right]\left[n_{jk}-\bar{n}^{(k)}_{jk}\right]\geq 0 & \text{if} & \Psi^{B_l^{(k)}}_{ijk}=\Psi^{L_l^{(k)}}_{ijk} \\ \gamma_{ij}\left[\tau_{ji}+\bar{\lambda}^{(k)}_{\Psi_{ik}}\right]\left[n_{jk}-\bar{n}^{(k)}_{jk}\right]<0 & \text{if} & \Psi^{B_l^{(k)}}_{ijk}=\Psi^{U_l^{(k)}}_{ijk} \end{array}\right\} \quad \forall \quad i,j\in C,\ k\in P. \tag{7.33}$$

As these inequalities define the y-domain of one of iteration (k)'s relaxed dual subproblems, it is clear that finding those qualifying constraints which are satisfied when $n=n^K$ simply involves determining all the y-domains from previous relaxed dual problems which contain $\bar{n}^K$. To efficiently identify these regions, McDonald and Floudas associate a node with each y-space subdomain, and use a tree structure to describe the genealogy of these nodes. The first node in the algorithm, termed the root node, is associated with the full y domain. In the first relaxed dual problem this domain is partitioned into $2^{|C|\times|P|}$ subdomains, where $|\cdot|$ denotes the cardinality of a set. The root node is therefore the parent of $2^{|C|\times|P|}$ nodes. These nodes spawn further nodes; in this manner the tree structure branches with the solution of each relaxed dual problem.

Example 10 *Consider the case where a region of the y-space is partitioned into two subregions in the relaxed dual problem, and let the algorithm take the course:*

Iteration 1: node 0, the root node, spawns nodes 1 and 2;

Iteration 2: node 1 spawns nodes 3 and 4;

Iteration 3: node 2 spawns nodes 5 and 6;

Iteration 4: node 4 spawns nodes 7 and 8.

The tree structure is shown in figure7.8. In practice, this tree is stored in an array p, where

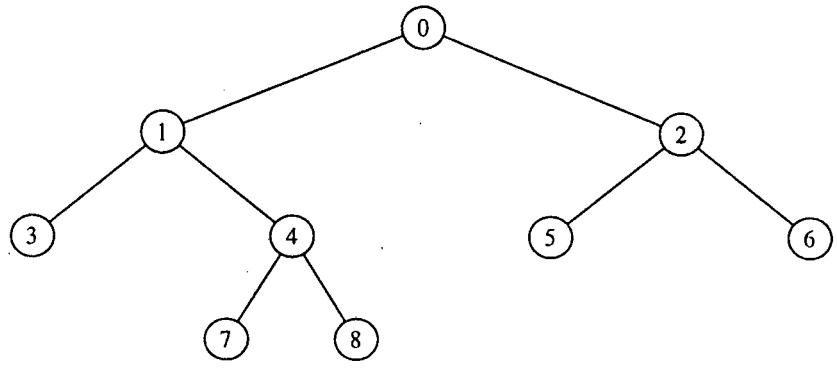

Figure 7.8: Tree structure relating subdomains

$p(k_t)$ is the parent of node k_t. The Lagrangians applicable to the relaxed dual can then be traced by traversing the tree, via child-parent links, from current node, k_c, to root node, k_r. Let, for example, the selected node, k_c, for iteration 5 be node 7. The parent of 7 is node 4, $p(7) = 4$, the parent of 4 is node 1, $p(4) = 1$, and the parent of 1 is the root node, $p(1) = 0$. The set of previous Lagrangians applicable to the current problem, denoted PL, is $\{4, 2, 1\}$ as the Lagrangians for nodes 4,1 and 0 were generated on iterations 4,2 and 1 respectively.

7.5.5 Calculating the x-Variable Bound Values

In the formulation of the relaxed dual problem, the x-variables are set to a combination of upper and lower bounds. It is at these points that the Lagrangian, being affine in Ψ, is minimized with regard to the x-variables. Each y-region is associated with a unique combination of bounds. In the previous section we stated the equations 7.33 which relate each y-space subregion to a particular set of bounds on the x-variables. In this section we will examine how, given a y-space region, the *numerical* values of the upper and lower bounds on the x-variables may be determined. The following minimization problems define the minimum and maximum values for each Ψ_{ik} when n is constrained to lie within an n-rectangle:

$$\begin{aligned} \min_n \Psi_{ik} &= \frac{n_{ik}}{\sum_{j\in C}\gamma_{ji}n_{jk}} \\ \text{subject to} \quad n_{jk}^L &\le n_{jk} \le n_{jk}^U \qquad \forall\ j \in C, k \in P, \end{aligned} \tag{7.34}$$

$$\begin{aligned} \min_n -\Psi_{ik} &= \frac{-n_{ik}}{\sum_{j\in C}\gamma_{ji}n_{jk}} \\ \text{subject to} \quad n_{jk}^L &\le n_{jk} \le n_{jk}^U \qquad \forall\ j \in C, k \in P. \end{aligned} \tag{7.35}$$

As the linear fractional objective functions of both problems are quasilinear, which means that they increase or decrease monotonically with each n_{ik}, the optimality conditions are in each case satisfied at only one point, hence the Karush-Kuhn-Tucker local optimality conditions can be used to evaluate the globally valid upper and lower bounds for these problems. The upper and lower bounds for Ψ_{ik}, analytically defined via the KKT conditions for the respective problems are:

$$\Psi_{ik}^L = \frac{n_{ik}^L}{n_{ik}^L + \sum_{j\neq i}\gamma_{ji}n_{jk}^U}, \Psi_{ik}^U = \frac{n_{ik}^U}{n_{ik}^U + \sum_{j\neq i}\gamma_{ji}n_{jk}^L}.$$

Given the y-space subregion defined by the n-rectangle:

$$\mathcal{B} = \left\{ n \mid n_{jk}^L \le n_{jk} \le n_{jk}^U \quad \forall \quad j \in C, k \in P \right\},$$

the bounds on the x-variables associated with this region can be determined through the constraint qualifications:

$$\begin{aligned} \hat{\Psi}_{ijk} &= \Psi_{jk}^U \quad \text{if} \quad \nabla_{\Psi_{ijk}} L(x, y, \bar{\lambda})\Big|_{\Psi_{ijk}} \le 0, \\ \hat{\Psi}_{ijk} &= \Psi_{jk}^L \quad \text{if} \quad \nabla_{\Psi_{ijk}} L(x, y, \bar{\lambda})\Big|_{\Psi_{ijk}} \ge 0. \end{aligned}$$

Substituting this application's Lagrangian function into these constraints we get:

$$\begin{aligned} \hat{\Psi}_{ijk} &= \Psi^U_{jk} \quad \text{if} \quad \gamma_{ij}\left[\tau_{ji} + \bar{\lambda}_{\Psi_{ik}}\right]\left[n_{jk} - \bar{n}_{jk}\right] \leq 0, \\ \hat{\Psi}_{ijk} &= \Psi^L_{jk} \quad \text{if} \quad \gamma_{ij}\left[\tau_{ji} + \bar{\lambda}_{\Psi_{ik}}\right]\left[n_{jk} - \bar{n}_{jk}\right] \geq 0. \end{aligned} \tag{7.36}$$

7.5.6 The Global Optimization Algorithm

Some notation essential for the description of the global optimization algorithm is introduced:

- $\mathcal{R}_{k_s} = [n^L_{k_s}, n^U_{k_s}]$ is the interval vector that defines the y-domain associated with node k_s.
- $\mathcal{B}_{B_l} = [n^L_{B_l}, n^U_{B_l}]$ is the interval vector that defines the y-domain associated with a relaxed dual problem with x-bound combination B_l.

The global optimization algorithm developed by McDonald and Floudas for the NRTL phase equilibrium problem is described below.

Step 0: Initialization

Set the lower bound on the global minimum: $D^L = -\infty$.

Set the upper bound on the global minimum: $P^U = +\infty$.

Set the convergence tolerance: ϵ.

Set the iteration counter: $K = 1$ the starting point: $\bar{n}^1$.

Step 1: The Primal Problem

Evaluate the primal problem at $\bar{n}^K$, yielding $\hat{G}(\bar{n}^K)$ the value of the primal objective, and the Lagrange multipliers $\bar{\lambda}^K$.

If $\hat{G}(\bar{n}^K) < P^U$ locally minimize the equilibrium problem to give $\hat{G}^*$. Update the upper bound on the global solution: $P^U = \min\{P^U, \hat{G}(\bar{n}^K), \hat{G}^*\}$.

Step 2: Select Previous Lagrangians

Select the set of Lagrangians from previous iterations, $PL(K)$, which are pertinent to the forthcoming relaxed dual problem:

Set $PL(K) = \emptyset$, $k_t = k_c$, where k_c is the current node under consideration;

while $(k_t \neq k_r)$, where k_r is the *root* node,

do $K_P = I_{k_t}$

$PL(K) = PL(K) \cup K_P$

$k_t = p(k_t)$

end do

end while

Step 3: The Relaxed Dual Phase

Partition the region defined by $\mathcal{R}_{k_c}$ by placing hyperplanes through n^{k_c}; store these bounds in $\mathcal{B}_{B_l}$.

for all $B_l^K \in CB$

calculate the bounds on the x-variables, $\hat{\Psi}^k$ and $\hat{\Psi}^K$ using equations 7.36 and solve the relaxed dual subproblem to give ξ^* and n^* :

$$\begin{aligned} \min_{n,\xi} \ & \xi \\ \text{subject to} \quad \xi &\geq L(\hat{\Psi}^k, n, \lambda^k) \qquad \forall \quad k \in PL(K) \\ \xi &\geq L(\hat{\Psi}^K, n, \lambda^K) \\ n_{B_l}^L &\leq n \leq n_{B_l}^U \\ An &= b \\ n &\geq 0 \end{aligned}$$

if $\xi^* \geq P^U$, then fathom the y-subregion;

if $\xi^* < P^U$, then set $k_s = k_s + 1$, $N_{k_s} = N_{k_s - 1} \cup k_s$, $p(k_s) = k_c$, $I_{k_s} = K$, $\xi^{k_s} = \xi^*$, $\bar{n}^{k_s} = n^*$ and

$\mathcal{R}_{k_s} = \mathcal{B}_{B_l}$.

end

Step 4: Select the y-Region for Next Iteration

Update the lower bound on the global minimum, $D^L = \min_{k_t \in N_{k_s}} \xi^{k_t}$, and select the corresponding node for the next iteration, $k_c = \arg \min_{k_t \in N_{k_s}} \xi^{k_t}$.

Remove k_c from the set of leaf nodes, N_{k_s}.

Set $n^{K+1} = \bar{n}^{k_c}$.

Step 5: Check for Convergence

If $\frac{P^U - D^L}{|P^U|} \leq \epsilon$ stop, the algorithm has converged; otherwise set $K = K + 1$ and return to Step 1.

7.6 Implementation

The global optimization algorithm was implemented in FORTRAN. The source code for this algorithm is to be found in appendix B. Implementational issues regarding this program are discussed in this section.

7.6.1 Infeasible Relaxed Dual Subproblems

In each iteration of the global optimization algorithm the variable space is partitioned into n-rectangles before the relaxed dual subproblems are solved. It is evident however that the molar variables cannot arbitrarily be assigned to these subregions as some partitions may result in problems that cannot satisfy the mass balance constraints. While in principle this is not a problem as the infeasible problems can be simply discarded once MINOS has determined them infeasible, the computational work involved in attempting to solve infeasible problems may not be insignificant, and a way of minimizing the number of infeasible subproblems is thus required.

In phase equilibrium problems this can be achieved by eliminating the variables for one of the phases modelled by the NRTL equation. In other words, if water is a component in the system, the moles of water in phase 2 may be expressed as:

$$n_{H_2O,2} = n_{H_2O,T} - n_{H_2O,1}$$

where $n_{H_2O,1}$ and $n_{H_2O,2}$ are respectively the number of moles of water in phase 1 and 2, and $n_{H_2O,T}$ is the total number of moles. Although the variable elimination approach can adequately deal with phase equilibrium problems, the method cannot be applied to problems involving chemical reactions. A more general approach which was developed to eliminate infeasible dual subproblems is described below.

If there are m mass independent balance constraints and n molar variables then the number of variables that can be assigned independently is $n - m$. These independent variables are termed "nonbasis". The other "basis" m variables are related to the nonbasis variables via the m mass balance constraints. Infeasibility in the relaxed dual subproblems can be detected by using the upper and lower bounds on the nonbasis variables and the mass balance constraints to determine the upper and lower bounds on the basis variables. This procedure is implemented using LINPACK subroutines DGESL and DGEFA. The use of this method on problems involving chemical reactions can eliminate many but not all infeasible subproblems.

7.6.2 Solving the Relaxed Dual Subproblems

MINOS 5.4 was used as a subroutine to solve the nonlinearly constrained convex relaxed dual subproblems. In an attempt to prevent MINOS from determining suboptimal solutions to these subproblems the convergence tolerances were set very tight. This approach, which is not entirely satisfactory as it does not prevent MINOS from terminating prematurely, produced acceptable results in practice. A better approach would be to use a convex programming algorithm that terminates on the basis of the ϵ-global optimality of the solution, the termination criterion used for the global optimization algorithm itself.

7.7 Test Problems

Test problems from McDonald and Floudas' paper were solved to assess the implementation of the global optimization algorithm. The data files containing NRTL parameters and other thermodynamic data for these problems are presented in appendix E.

7.7.1 Problem 1: n-Butyl Acetate + Water

This problem involving only two species is useful as it facilitates a graphical representation of the global optimization process. The problem involves 0.5 moles of each species in a system at a temperature of 298K and a pressure of 1atm. The datafile for this problem is "bute-wate.dat". A graphical representation of the Gibbs energy surface appears in figure 7.9 where the Gibbs energy is plotted as a function of n_{11} and n_{21}, which are respectively the moles of n-butyl acetate and water in phase 1. In this figure the mass balance constraints have been taken into account by explicitly relating n_{i2} to n_{i1} via the relations: $n_{12} = n_{1T} - n_{11}$, and $n_{22} = n_{2T} - n_{21}$. Features of interest on the Gibbs energy surface are:

1. the valley of local minima running along the locus of equimolar composition. This is the composition of the feed to the system. Along this locus, both phases are identical in composition, and the physical system exhibits a single phase state. This false single phase, known as the trivial solution, is a frequent cause of error in phase equilibrium computations carried out by local search techniques.

2. local minima at $n_{11} = 0.4979$, $n_{21} = 0.0345$, and at $n_{11} = 0.0021$, $n_{21} = 0.4655$ where $\hat{G} = -0.0196$.

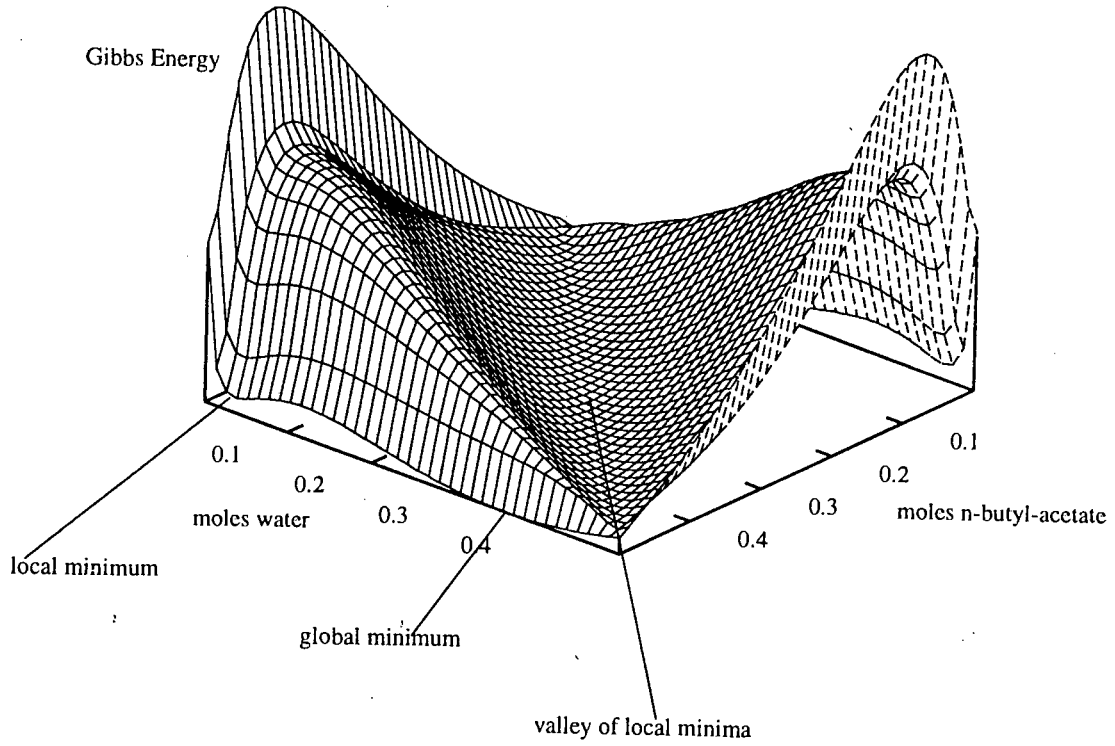

Figure 7.9: Gibbs energy surface: n-butyl acetate + water

3. global minima at $n_{11} = 0.4993$, $n_{21} = 0.3441$, and at $n_{11} = 0.0007$ $n_{21} = 0.1559$ where $\hat{G} = -0.2020$.

Notice that, without compromising the rigor of the formulation, the domain of the search variables can be reduced by specifying phase 1 to be rich in n-butyl acetate. In other words the problem can be formulated as:

$$\begin{aligned} \min_{n_{11},n_{21}} \hat{G}(n) & \\ \text{subject to } n_{12} &= 0.5 - n_{12} \\ n_{22} &= 0.5 - n_{22} \\ 0 &\leq n_{11} \leq 0.25 \\ 0 &\leq n_{21} \leq 0.5 \end{aligned}$$

Table 7.2 shows the key statistics on the global optimization of this problem: the iteration number, the upper and lower bounds on $\hat{G}$, the convergence test criterion, the total number of relaxed duals solved, and the number of unfathomed leaf nodes.

Iter	Phase 1 Moles		Phase 2 Moles		Objective
	n-Butyl Acetate	Water	n-Butyl Acetate	Water	
1	0.25	0.25	0.25	0.25	-.175799420E-01
32	0.7135E-03	0.1559	0.4993	0.3441	-.202001744E-01

Table 7.1: Local Minima found by the Global Optimization Algorithm: Problem 1

Iter	UpperBound P^U	LowerBound D^L	ConvergenceTest $\frac{P^U - D^L}{\lvert P^U \rvert}$	Sub-Duals	Leaf Nodes
1	-0.175799420E-01	-0.292125395	0.156169715E+02	4	4
2	-0.175799420E-01	-0.166008562	0.844306652E+01	5	4
3	-0.175799420E-01	-0.158945020	0.804127103E+01	7	5
20	-0.175799420E-01	-0.366072621E-01	0.108233123E+01	67	42
40	-0.202001744E-01	-0.264791353E-01	0.310836960E+00	131	43
60	-0.202001744E-01	-0.232376450E-01	0.150368529E+00	201	39
80	-0.202001744E-01	-0.210534484E-01	0.422409217E-01	261	30
100	-0.202001744E-01	-0.203972881E-01	0.975801815E-02	331	12
120	-0.202001744E-01	-0.202036688E-01	0.172988815E-03	405	17
140	-0.202001744E-01	-0.202003128E-01	0.685020870E-05	473	21
160	-0.202001744E-01	-0.202001830E-01	0.423000395E-06	525	10
180	-0.202001744E-01	-0.202001748E-01	0.191884195E-07	595	20
200	-0.202001744E-01	-0.202001745E-01	0.144482600E-08	651	12
222	-0.202001744E-01	-0.202001744E-01	0.863357548E-10	713	21

Table 7.2: Global Optimization Statistics: Problem 1

For this problem, as there are 2 independent y variables the maximum number of relaxed dual subproblems on any iteration is four. Notice, however, that frequently less that four relaxed duals were solved on an iteration. This is due to the parameter $\bar{n}$ being on the boundary of the rectangle whose partitioning it defines. Local minima in table 7.1 show that the globally optimal solution was found early in the run, on iteration 32.

Guaranteeing the solution by raising the lower bound to within the convergence tolerance, is therefore the reason for the large number of iterations needed to satisfy a convergence criterion $\epsilon = 10^{-10}$. An interesting point about the number of unfathomed leaf nodes in this example, is its tendency to remain at a fairly constant level. This observation, unfortunately, does not apply in general, in other examples different behaviour can be observed.

The convergence criterion involves guaranteeing the value of the minimum, not the minimizer, and is in this sense inappropriate for the phase equilibrium problem where the value of the minimizer is all-important. Through the fathoming process the compositions that are definitely not global minimizers are discarded, enabling the sufficiency of the value ϵ to

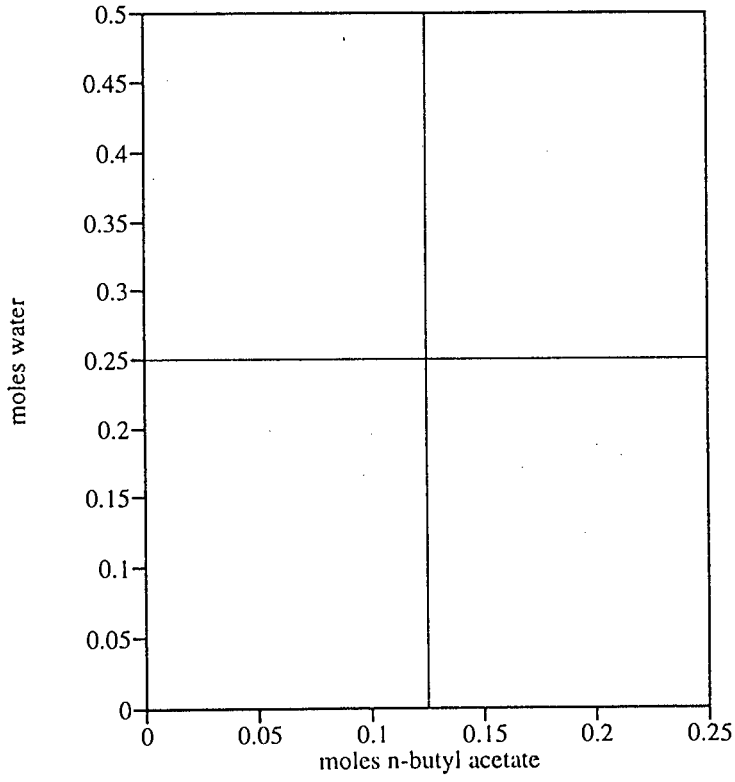

Figure 7.10: Partitioning and fathoming for problem 1: iteration 1

guarantee the *minimizer* to be assessed by examining the regions which are unfathomed at various stages of the optimization. Figures 7.10 to 7.14 represent the partitioned y-domain at representative iterations, the shaded regions in these figures being the fathomed subregions at the specified iteration. The first figure in the series shows how on the first iteration the y-domain is partitioned into 4 subdomains. A comparison of figure 7.11 with figure 7.9 reveals the expected result that the regions to be fathomed first are coincident with the highlands of the Gibbs energy surface. At a more advanced stage in the minimization, figure 7.12 shows an unfathomed band in the region of the trivial solution valley. Figure 7.13 shows the existence of a number of unfathomed regions at iteration 100, in the region of the false local minima, and near the global minimum. In the advanced stages of the minimization the effect of ill-conditioning becomes apparent in the elongated shape of the subregions near the global optimum. Because the rectangles are longer than they are wide, the value of the global n_1^1minimizers can be guaranteed to a far greater accuracy than the n_2^1 minimizers. By iteration 140 almost the entire domain has been fathomed, the only unfathomed region being the small area containing the global solution. In this example a convergence criterion of 0.5×10^{-5} appears to be sufficient to guarantee a good solution.

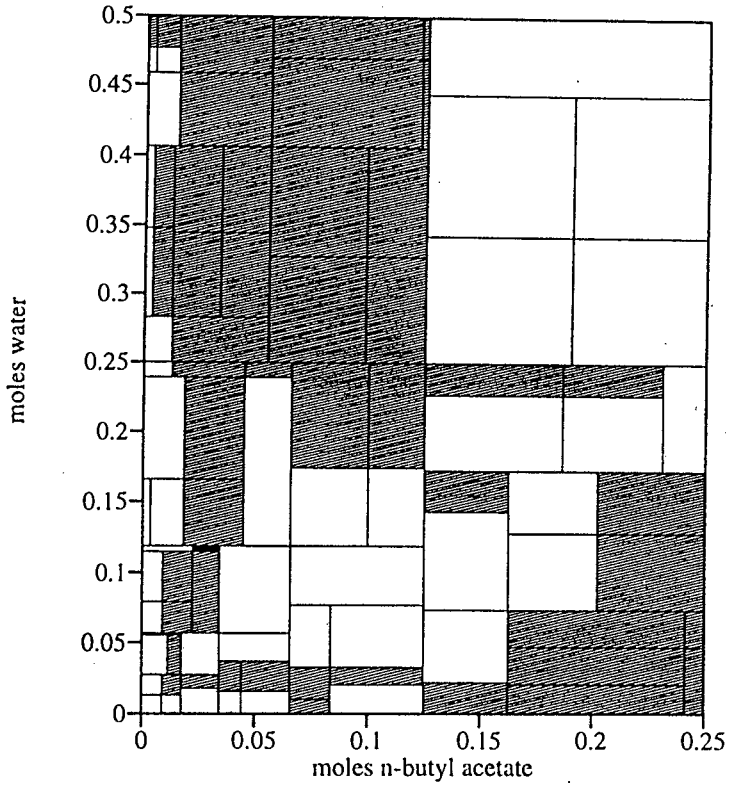

Figure 7.11: Partitioning and fathoming for problem 1: iteration 40

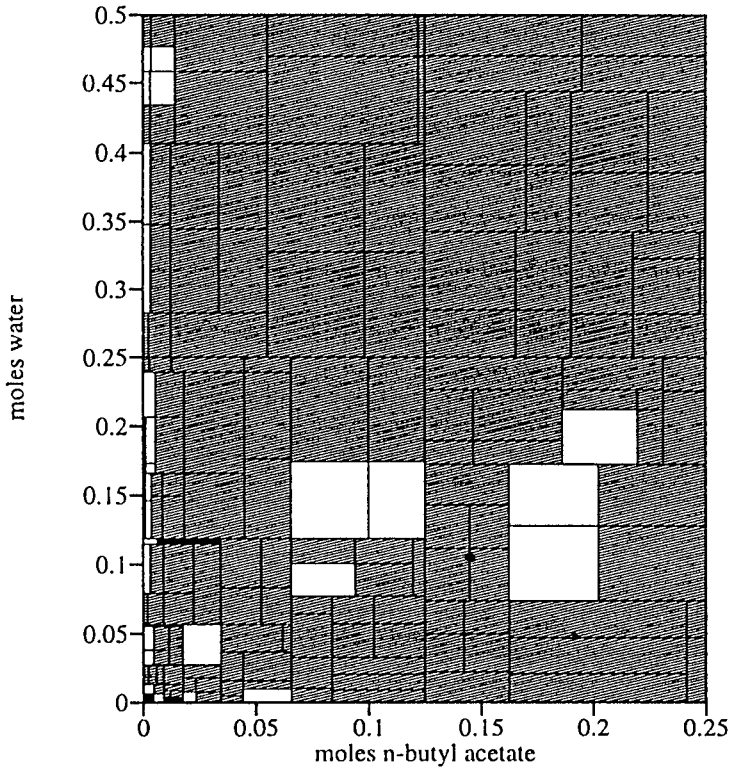

Figure 7.12: Partitioning and fathoming for problem 1: iteration 80

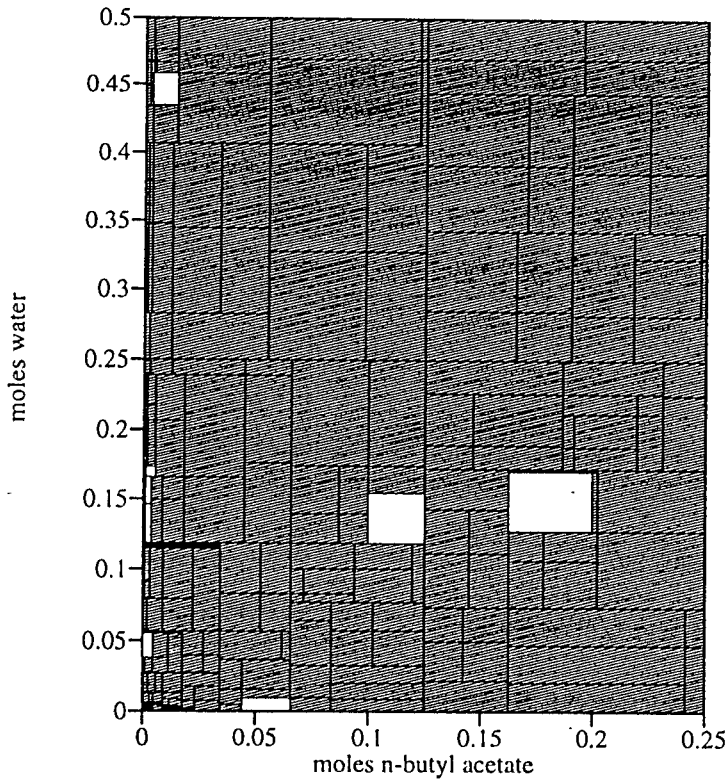

Figure 7.13: Partitioning and fathoming for problem 1: iteration 100

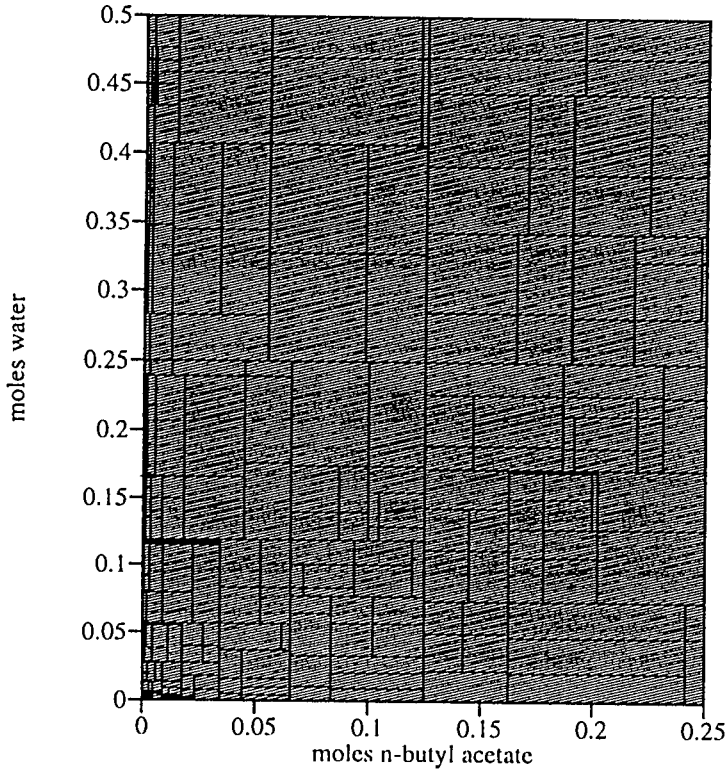

Figure 7.14: Partitioning and fathoming for problem 1: iteration 140

7.7.2 Problem 2: LL Equilibrium for Toluene + Water + Aniline

This example, investigated by Castillo and Grossmann [12] , involves a ternary system at a temperature of 298 K and a pressure of 1 atm. The parameters for the NRTL equation used to model both liquid phases were obtained from Bender and Block [3] , and are listed in datafile "tolu-wate-anil.dat" in appendix E. In this example there are three mass balance constraints and 6 variables, hence there are 3 nonbasis variables. On every iteration the current box region is partitioned into a maximum of 2^3 subdomains, therefore the maximum number of relaxed dual subproblems that need to be solved on any iteration is 8. Statistics from the global optimization run are recorded in table 7.4, local solutions found during the course of the optimization are shown in table 7.3. The global minimum was discovered on

		Iter1	Iter 7
Phase 1 Moles	Toluene	0.1498	0.1233E-04
	Water	0.9990E-01	0.1343
	Aniline	0.2497	0.6696E-03
Phase 2 Moles	Toluene	0.1498	0.2995
	Water	0.9990E-01	0.6551E-01
	Aniline	0.2497	0.4987
Objective		-0.324348794	-0.352497801

Table 7.3: Local Minima found by the Global Optimization Algorithm: Problem 2

iteration 7. 378 iterations were required to confirm the ϵ-global optimality of this solution for $\epsilon = 10^{-10}$. The number of leaf nodes for this problem remained small relative to the total number of relaxed dual problems solved.

7.7.3 Problem 3: LL Equilibrium for n-Propanol + n-Butanol + Water

The NRTL parameters for this system originate from Block and Hegner [4]. Two of the feed compositions, which were used by Walraven and van Rompay [80] to test their phase splitting algorithm are used to define two test problems for the n-propanol, n-butanol and water system.

Iter	Upper Bound P^U	Lower Bound D^L	Convergence Test $\frac{P^U-D^L}{\lvert P^U\rvert}$	Sub-Duals	Leaf Nodes
1	-0.324348794	-0.771844626	0.137967472E+01	8	8
2	-0.324348794	-0.595710744	0.836636221	9	8
3	-0.324348794	-0.571616176	0.762350242	13	10
10	-0.352497801	-0.455344245	0.291764780	57	34
20	-0.352497801	-0.398849130	0.131493952	121	50
40	-0.352497801	-0.369187256	0.473462675E-01	237	35
60	-0.352497801	-0.359552748	0.200141577E-01	337	17
80	-0.352497801	-0.352726363	0.648406443E-03	451	10
100	-0.352497801	-0.352500207	0.682466500E-05	545	28
200	-0.352497801	-0.352497809	0.227286787E-07	945	86
300	-0.352497801	-0.352497801	0.826720528E-09	1289	124
378	-0.352497801	-0.352497801	0.959802453E-10	1523	119

Table 7.4: Global Optimization Statistics: Problem 2

Feed 1

The datafile for this problem is "prol-buol-wat1.dat". This feed composition lies well within the immiscibility region. Local solutions found by the global optimization algorithm are reported in table 7.5.

		Iter 1	Iter 17
Phase 1 Moles	n-Propanol	0.2000E-01	0.4904E-02
	n-Butanol	0.8000E-01	0.9513E-02
	Water	0.4000E+00	0.4153E+00
Phase 2 Moles	n-Propanol	0.2000E-01	0.3510E-01
	n-Butanol	0.8000E-01	0.15049E+00
	Water	0.4000E+00	0.3847E+00
Objective		-.222036392E+00	-.226149289E+00

Table 7.5: Local Minima found by the Global Optimization Algorithm: Problem 3 Feed 1

The global solution found on iteration 17 was confirmed to be an $\epsilon-$global optimum in 803 iterations, where $\epsilon = 10^{-5}$. As the difference between the global and trivial solutions is greater than 10^{-3}, $\epsilon = 10^{-3}$ would have been a sufficiently tight tolerance to avoid the trivial solution. Statistics in table 7.6 show that the number of leaf nodes stabilizes after iteration 200.

Iter	Upper Bound P^U	Lower Bound D^L	Convergence Test $\frac{P^U - D^L}{\lvert P^U \rvert}$	Sub-Duals	Leaf Nodes
1	-0.222036392	-0.582488762	0.162339321E+01	8	8
2	-0.222036392	-0.518933369	0.133715458E+01	10	9
3	-0.222036392	-0.501450125	0.125841413E+01	14	11
10	-0.222036392	-0.321933400	0.449912771	50	35
20	-0.226149289	-0.269112028	0.189975122	96	63
40	-0.226149289	-0.247600687	0.948550317E-01	202	133
60	-0.226149289	-0.241268893	0.668567399E-01	294	165
80	-0.226149289	-0.238039220	0.525755845E-01	390	178
100	-0.226149289	-0.235741009	0.424132221E-01	478	195
200	-0.226149289	-0.229260158	0.137558189E-01	911	217
400	-0.226149289	-0.226498582	0.154452360E-02	1845	141
600	-0.226149289	-0.226159098	0.433738793E-04	2861	130
803	-0.226149289	-0.226149514	0.994967010E-06	3856	105

Table 7.6: Global Optimization Statistics: Problem 3 Feed 1

Feed 2

The data defining this problem are to be found in the datafile "prol-buol-wat2.dat" in appendix E. This feed composition lies very close to the plait point of the system, a region in which it is notoriously difficult to correctly compute phase equilibrium. Table 7.7 shows the local solutions that were found. Unlike feed 1, the global solution was not found in the early stages of the optimization but on the 510th iteration.

		Iter1	Iter 510
Phase 1 Moles	n-Propanol	0.7400E-01	0.2002E-01
	n-Butanol	0.2600E-01	0.6358E-02
	Water	0.4000E+00	0.1451E+00
Phase 2 Moles	n-Propanol	0.7400E-01	0.1280E+00
	n-Butanol	0.2600E-01	0.4564E-01
	Water	0.4000E+00	0.65490E+00
Objective		-.270812067E+00	-.270813132E+00

Table 7.7: Local Minima found by the Global Optimization Algorithm: Problem 3 Feed 2

This result is due to the narrow margin, of the order of 10^{-6}, between the objective function evaluated at the global and trivial solutions. A local search is initiated only when a primal

Iter	Upper Bound P^U	Lower Bound D^L	Convergence Test $\frac{P^U-D^L}{\lvert P^U\rvert}$	Sub-Duals	Leaf Nodes
1	-0.270812067	-0.577842622	0.113374031E+01	8	8
10	-0.270812067	-0.344260120	0.271214110	46	32
100	-0.270812067	-0.280740645	0.366622456E-01	528	354
500	-0.270812067	-0.272845159	0.750739115E-02	2348	1240
1000	-0.270813132	-0.271773101	0.354476526E-02	4506	2309
6000	-0.270813132	-0.270917786	0.386442608E-03	26181	12133

Table 7.8: Global Optimization Statistics: Problem 3 Feed 2

solution produces a minimum which improves on the previous best minimum. Because of the narrow margin between the different local minima in this example such an improvement is unlikely to occur unless the primal objective is evaluated at a point very near to the local minimum itself. The statistics from the global optimization show further signs of the difficulty of this problem:

1. a proliferation of leaf nodes which shows no sign of levelling off even after 6000 iterations and 26181 relaxed dual problems;

2. an upper/lower bound convergence rate which is extremely slow compared to that in feed 1.

This example demonstrates that despite the method's finite convergence property, its memory and iteration requirements may become excessive, even for a small problem.

7.7.4 Problem 4: LL Equilibrium for Ethanol + Ethyl Acetate + Water

The NRTL parameters for this ternary system are listed in the file "etOH-etAc-wate.dat" in appendix E. The local solution data in table 7.9 shows that the global solution was determined on iteration 6. Table 7.10 shows the ϵ-optimality of this solution was guaranteed for $\epsilon = 10^{-5}$, in 351 iterations.

The number of leaf nodes in this problem stabilized below 100 early in the run.

		Iter1	Iter 6
Phase 1 Moles	Ethanol	0.2000E-01	0.2376-01
	Ethyl Acetate	0.1500	0.2622
	Water	0.3300	0.1279
Phase 2 Moles	Ethanol	0.2000E-01	0.1624E-01
	Ethyl Acetate	0.1500	0.3785E-01
	Water	0.3300	0.5321
Objective		-0.192717963	-0.213142208

Table 7.9: Local Minima found by the Global Optimization Algorithm: Problem 4

Iter	Upper Bound P^U	Lower Bound D^L	Convergence Test $\frac{P^U - D^L}{\lvert P^U \rvert}$	Sub-Duals	Leaf Nodes
1	-0.192717963	-0.588264320	0.205246232E+01	8	8
2	-0.192717963	-0.529071287	0.174531382E+01	10	9
3	-0.192717963	-0.524976211	0.172406475E+01	18	14
10	-0.213142208	-0.351548835	0.649362831	50	29
20	-0.213142208	-0.261638558	0.227530486	96	57
40	-0.213142208	-0.239020477	0.121413162	194	90
60	-0.213142208	-0.230598274	0.818986828E-01	302	96
80	-0.213142208	-0.223668016	0.493839695E-01	388	84
100	-0.213142208	-0.218041910	0.229879505E-01	484	94
200	-0.213142208	-0.213468151	0.152922732E-02	995	71
300	-0.213142208	-0.213152400	0.478167651E-04	1509	76
351	-0.213142208	-0.213144252	0.959076781E-05	1779	88

Table 7.10: Global Optimization Statistics: Problem 4

7.7.5 Problem 5: LL Equilibrium for n-Butanol + Water + n-Butyl Acetate

Datafile "bute-wate-buAc.dat" in appendix E lists the data for this ternary system. Local solution data in table 7.11 show that the global optimum was determined in 6 iterations and confirmed for $\epsilon = 10^{-5}$ in 249 iterations. Data in table 7.12 show the performance of the algorithm.

		Iter 1	Iter 6
Phase 1 Moles	n-Butanol	0.7000E-01	0.3973E-02
	Water	0.3200	0.4734
	n-Butyl-Acetate	0.1100E+00	0.1091E-02
Phase 2 Moles	n-Butanol	0.7000E-01	0.1360E+00
	Water	0.3200	0.1666
	n-Butyl-Acetate	0.1100	0.2189
Objective		-0.224828753	-0.264923144

Table 7.11: Local Minima found by the Global Optimization Algorithm: Problem 5

Iter	UpperBound P^U	LowerBound D^L	ConvergenceTest $\frac{P^U - D^L}{\lvert P^U \rvert}$	Sub-Duals	Leaf Nodes
1	-0.224828753	-0.790526490	0.251612719E+01	8	8
2	-0.224828753	-0.645106254	0.186932275E+01	9	8
3	-0.224828753	-0.575547941	0.155993922E+01	13	10
10	-0.264923144	-0.407148467	0.536855033	45	27
20	-0.264923144	-0.313813093	0.184543896	109	45
30	-0.264923144	-0.291148421	0.989920195E-01	161	43
40	-0.264923144	-0.284124903	0.724804911E-01	219	38
50	-0.264923144	-0.276847282	0.450098021E-01	287	37
60	-0.264923144	-0.271477319	0.247399120E-01	336	35
70	-0.264923144	-0.267531004	0.984383615E-02	390	25
80	-0.264923144	-0.266738036	0.685063899E-02	452	48
90	-0.264923144	-0.265572290	0.245031790E-02	510	39
100	-0.264923144	-0.265315375	0.148054910E-02	572	59
150	-0.264923144	-0.264987923	0.244520559E-03	760	88
249	-0.264923144	-0.264925608	0.930093148E-05	1142	136

Table 7.12: Global Optimization Statistics: Problem 5

7.7.6 Problem 6: LV Esterification Reaction

This problem involves the following esterification reaction which occurs in the liquid and vapour phases:

$$C_2H_5OH + CH_3COOH \rightleftharpoons CH_3COOC_2H_5 + H_2O.$$

Numerous authors besides McDonald and Floudas, Sanderson and Chien [65], Castillo and Grossmann [12], Lantange et al. [39], Castier et al. [11], Gautam and Seider [23], and Paules and Floudas [57], have used this example to test a variety of equilibrium computation procedures. The NRTL parameters from McDonald and Floudas are listed in "etOH-Acet-EtAc-Wate.dat", appendix E. The gas phase is modelled using the ideal gas law. There are 8 variables in this problem, 3 mass balance equations, hence there are 5 nonbasis variables. Only four variables correspond to the liquid phase, however, and as the gas phase component of the Gibbs energy function is convex the box region needs to be partitioned into a maximum of only 2^4 subregions for each relaxed dual subproblem. The global minimum in table 7.13 is the only local minimum that was determined by the global optimization algorithm. Statistics in table 7.14 show the progress of the method and the computational effort required to guarantee the global optimality of the solution.

		Iter 2
Vapour Phase Moles	Ethanol	0.7442E-01
	Acetic Acid	0.6651E-01
	Ethyl Acetate	0.4197E+00
	Water	0.3906E+00
Liquid Phase Moles	Ethanol	0.1937E-02
	Acetic Acid	0.9845E-02
	Ethyl Acetate	0.3968E-02
	Water	0.3303E-01
Objective		-.907704620E+02

Table 7.13: Local Minima found by the Global Optimization Algorithm: Problem 6

7.7.7 Discussion

In all cases the global optimization algorithm successfully found the global optimum and is therefore a very reliable means of computing the global optimum. The practical value of this algorithm is questionable due to the extremely large computational requirements. In all test examples besides feed 2 in example 3 the global minimum was found in the early iterations; the computational effort required to prove the global optimality of the solution

Iter	Upper Bound P^U	Lower Bound D^L	Convergence Test $\frac{P^U-D^L}{\lvert P^U\rvert}$	Sub-Duals	Leaf Nodes
1	0.100000000E+11	-0.926600885E+02	0.100000001E+01	16	6
2	-0.907704620E+02	-0.915289605E+02	0.835622521E-02	17	4
3	-0.907704620E+02	-0.914449804E+02	0.743103372E-02	25	4
100	-0.907704620E+02	-0.907803370E+02	0.108790450E-03	829	226
200	-0.907704620E+02	-0.907741243E+02	0.403467923E-04	1541	436
400	-0.907704620E+02	-0.907716131E+02	0.126807547E-04	2804	631
600	-0.907704620E+02	-0.907710147E+02	0.608853012E-05	4075	846
800	-0.907704620E+02	-0.907707657E+02	0.334576874E-05	5260	887
1000	-0.907704620E+02	-0.907706414E+02	0.197576965E-05	6536	991
1200	-0.907704620E+02	-0.907705769E+02	0.126499889E-05	7791	1123
1306	-0.907704620E+02	-0.907705528E+02	0.999444107E-06	8413	1169

Table 7.14: Global Optimization Statistics: Problem 6

was the cause for the exorbitant number of iterations required.

The inefficiency of the global optimization approach poses the question of how the algorithm could be improved, without compromising its rigorous nature. One of the reasons for the large computation time is the branch and bound nature of the approach which only terminates once a sufficiently fine partitioning about the optimal solution has been achieved. An approach that may be useful in this regard would make use of convexity information to fathom regionally convex regions of the Gibbs energy surface. For the Gibbs energy minimization problem where the constraints are linear this approach would involve showing that the reduced Hessian is positive definite in the region around the local minimum. As there is no precise way of doing this, some type of approximation would be required where the extent of the locally convex region would be underestimated. Neumaier [54] suggests an interval analysis approach to testing the convexity of a region. In section 7.8 a new convexity test is developed, and its benefits when applied to the global optimization algorithm are assessed.

7.8 A Regional Convexity Test

One of the problems observed in the global optimization algorithm is the very fine partitioning of the y-domain which is needed to guarantee the ϵ-optimality of a proposed solution. In the chemical equilibrium problem the global minimum is frequently in the relative interior of the feasible domain and is therefore the minimizer of a locally convex region. As standard nonlinear programming methods may be used to find the global minimizer of a convex minimization problem, the global minimum of a region in the y-domain where the problem

is convex may be found in this way. If a region can be proven convex, the relaxed dual approach to determining a lower bound on the global minimum for this region can be superseded by a minimization of the Gibbs energy function itself. The difficulty in implementing this approach, however, is in establishing whether or not a specified region is convex.

An $n \times n$ symmetric matrix H is said to be positive definite if for all $y \in \Re^n$, $y^T H y > 0$. A function $f(x)$ which has continuous second partial derivatives can be shown to be convex at a point x^* if the Hessian matrix $H(x^*)$ is positive definite. The function is convex when it is convex for all points in its domain. Consider now a linearly constrained minimization problem:

$$\min f(x)$$
$$\text{subject to } Ax = b$$

The projected Hessian matrix $\tilde{H}(x)$ is defined as $Z^T H(x) Z$, where Z is a matrix whose columns form a basis for the null-space of A. A sufficient condition for $f(x)$ to be convex relative to the feasible domain is that the projected Hessian must be positive definite for all feasible x. Let x^L and x^U be the lower and upper bounds on x which define an n-rectangle $\mathcal{B} = \{x : x^L \leq x \leq x^U\}$. Described in this section is a sufficient condition to validate the following:

$$Z^T H(x) Z > 0 \quad ,\forall x \in \mathcal{B} \tag{7.37}$$

where $>$ means "positive definite".

7.8.1 A Test for Positive Definiteness

Before considering a sufficient test for 7.37, a positive definiteness test for a symmetric matrix is introduced. Consider a symmetric matrix:

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \cdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{bmatrix}.$$

If the k^{th} leading principle submatrix:

$$\begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1k} \\ m_{21} & m_{22} & \cdots & m_{2k} \\ \vdots & \cdots & \ddots & \vdots \\ m_{k1} & m_{k2} & \cdots & m_{kk} \end{bmatrix}$$

is denoted, $M[1,\ldots,k]$, the k^{th} principle minor of this matrix is defined as the determinant of $M[1,\ldots,k]$. A symmetric matrix is positive definite if and only if all its principle minors are positive [58].

A convenient way of determining whether the principle minors are positive is described by Cottle [13]. The method is based on the formula for the determinant:

$$\begin{aligned} \det M &= \det B \det(D - C^T B^{-1} C) \\ \text{where} \quad M &= \begin{bmatrix} B & C \\ C^T & D \end{bmatrix}. \end{aligned} \tag{7.38}$$

The matrix $D - C^T B^{-1} C$ is called the Schur complement of B in M and is denoted M/B. Consider M/m_{11}, which is denoted $M^{(1)}$:

$$M^{(1)} = \begin{bmatrix} m_{22} - \frac{m_{21}m_{12}}{m_{11}} & m_{23} - \frac{m_{21}m_{13}}{m_{11}} & \cdots & m_{2n} - \frac{m_{21}m_{1n}}{m_{11}} \\ m_{32} - \frac{m_{31}m_{12}}{m_{11}} & m_{33} - \frac{m_{31}m_{13}}{m_{11}} & & \vdots \\ \vdots & & \ddots & \vdots \\ m_{n2} - \frac{m_{n1}m_{12}}{m_{11}} & \cdots & \cdots & m_{nn} - \frac{m_{n1}m_{1n}}{m_{11}} \end{bmatrix}.$$

The leading element of this matrix, $m_{11}^{(1)} = m_{22} - \frac{m_{21}m_{12}}{m_{11}}$, is the Schur complement $M[1,2]/M[1]$. Furthermore, $M^{(1)}[1] = m_{11}^{(1)} = \frac{\det M[1,2]}{\det M[1]}$, therefore if the first principal minor of M is positive, the second is positive only if $m_{11}^{(1)}$ is positive. Note that $\det M^{(1)}[1,2] = \frac{\det M[1,\ldots,3]}{\det M[1]}$ and that by taking the Schur complement $M^{(2)} = M^{(1)}/m_{11}^{(1)}$ the leading element of $M^{(2)}$ can be shown to be equivalent to:

$$\frac{\det M^{(1)}[1,2]}{\det M^{(1)}[1]} = \frac{\frac{\det M[1,\ldots,3]}{\det M[1]}}{\frac{\det M[1,2]}{\det M[1]}} = \frac{\det M[1,\ldots,3]}{\det M[1,2]}$$

Continuing the process whereby $M^{(k+1)} = M^{(k)}/m_{11}^{(k)}$, and provided no element $m_{11}^{(j)}$is zero for $j < k < n$, the following general relation obtains:

$$m_{11}^{(k)} = \frac{\det M[1,\ldots,k+1]}{\det M[1,\ldots,k]}.$$

This scheme is simple to implement; finding $M^{(k+1)}$ from $M^{(k)}$ can be done using the following relation:

$$m_{i,j}^{(k+1)} = m_{i+1,j+1}^{(k)} - \frac{m_{i+1,1}^{(k)} m_{1,j+1}^{(k)}}{m_{11}^{(k)}}.$$

Via this method a symmetric matrix is proven to be positive definite if no element $m_{11}^{(k)}$ is zero or negative for $k = 1,\ldots,n-1$.

7.8.2 An Interval Matrix Positive Definiteness Test

Now we consider the problem of determining if all the projected Hessian matrices parametrized by x are positive definite in $\mathcal{B}$. This is done through the use of an interval matrix $\mathcal{M}$ whose elements are all intervals $[m_{ij}^L, m_{ij}^U]$:

$$\mathcal{M} = \begin{bmatrix} [m_{11}^L, m_{11}^U] & \cdots & [m_{1n}^L, m_{1n}^U] \\ \vdots & \ddots & \vdots \\ [m_{n1}^L, m_{n1}^U] & \cdots & [m_{nn}^L, m_{nn}^U] \end{bmatrix}.$$

Firstly, we assume that we can identify an interval matrix that contains the projected Hessian matrix evaluated at any point within the n-rectangle:

$$\tilde{H}(x) \in \mathcal{M} \quad \forall x \in \mathcal{B}.$$

The determination of such an interval matrix for the NRTL equation's Hessian will be discussed in section 7.8.3. If every matrix contained within this interval matrix can be shown to be positive definite, from our assumption, we can then conclude that the projected Hessian itself is positive definite throughout the region.

The positive definiteness test for a symmetric matrix discussed in section 7.8.1 is the basis for the positive definiteness test on the interval matrix. Let the lower and upper limits of the interval matrix $\mathcal{M}$ constitute the matrices M^L and M^U respectively:

$$M^L = \begin{bmatrix} m_{11}^L & \cdots & m_{1n}^L \\ \vdots & \ddots & \vdots \\ m_{n1}^L & \cdots & m_{nn}^L \end{bmatrix},$$

$$M^U = \begin{bmatrix} m_{11}^U & \cdots & m_{1n}^U \\ \vdots & \ddots & \vdots \\ m_{n1}^U & \cdots & m_{nn}^U \end{bmatrix}.$$

Starting with interval matrix $\mathcal{M}$ the proposed procedure generates a series of interval matrices $\mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(n-1)}$ which are computed in a conservative manner to ensures that they respectively contain $\tilde{H}^{(1)}(x), \ldots, \tilde{H}^{(n-1)}(x)$ for any $x \in \mathcal{B}$. If at any stage k, $m_{11}^{L(k)}$ is found to be negative then, due to the conservativeness of the approach, nothing can be said about the positive definiteness of $\tilde{H}(x)$ for $x \in \mathcal{B}$. On the other hand, if $m_{11}^L, m_{11}^{L(1)}, \ldots, m_{11}^{L(n-1)}$ are all positive then $\tilde{H}(x)$ is guaranteed to be positive definite for all $x \in \mathcal{B}$. The algorithm is stated as:

Step 1: If $m_{ii}^{(0)L} \equiv m_{ii}^L > 0$ for all $i = 1, \ldots, n$ go to step 2, otherwise stop; positive definiteness cannot be shown.

Step 2: for $k = 1, \ldots, n-1$

Determine $M^{(k)L}$ and $M^{(k)U}$ using $M^{(k-1)L}$ and $M^{(k-1)U}$, carrying out the calculation in such a way that $M^{(k)L}$ and $M^{(k)U}$ respectively contain the smallest and largest elements which can be derived from elements contained in the interval matrix: $\mathcal{M}^{(k-1)}$:

$$\text{let } r_{ij}^{(k-1)L} = \min \left\{ \begin{array}{ll} m_{i+1,1}^{(k-1)L} m_{1,j+1}^{(k-1)L}, & m_{i+1,1}^{(k-1)L} m_{1,j+1}^{(k-1)U}, \\ m_{i+1,1}^{(k-1)U} m_{1,j+1}^{(k-1)L}, & m_{i+1,1}^{(k-1)U} m_{1,j+1}^{(k-1)U} \end{array} \right\}$$

$$\text{let } r_{ij}^{(k-1)U} = \max \left\{ \begin{array}{ll} m_{i+1,1}^{(k-1)L} m_{1,j+1}^{(k-1)L}, & m_{i+1,1}^{(k-1)L} m_{1,j+1}^{(k-1)U}, \\ m_{i+1,1}^{(k-1)U} m_{1,j+1}^{(k-1)L}, & m_{i+1,1}^{(k-1)U} m_{1,j+1}^{(k-1)U} \end{array} \right\}$$

$$\text{if } r_{ij}^{(k-1)U} \geq 0 \text{ let } m_{ij}^{(k)L} = m_{i+1,j+1}^{(k-1)L} - \frac{r_{ij}^{(k-1)U}}{m_{11}^{(k-1)L}}$$

$$\text{if } r_{ij}^{(k-1)U} < 0 \text{ let } m_{ij}^{(k)L} = m_{i+1,j+1}^{(k-1)L} - \frac{r_{ij}^{(k-1)U}}{m_{11}^{(k-1)U}}$$

$$\text{if } r_{ij}^{(k-1)L} \geq 0 \text{ let } m_{ij}^{(k)U} = m_{i+1,j+1}^{(k-1)U} - \frac{r_{ij}^{(k-1)L}}{m_{11}^{(k-1)U}}$$

$$\text{if } r_{ij}^{(k-1)L} < 0 \text{ let } m_{ij}^{(k)U} = m_{i+1,j+1}^{(k-1)U} - \frac{r_{ij}^{(k-1)L}}{m_{11}^{(k-1)L}}$$

Check positivity of $m_{11}^{(k)L}$:

if $m_{11}^{(k)L} \geq 0$ continue, otherwise stop; positive definiteness cannot be shown.

end

Step 3 Matrix $\tilde{H}(x)$ is positive definite for all $x \in \mathcal{B}$.

To implement this method, the matrix $\mathcal{M}$ must be determined. In section 7.8.3 this is discussed.

7.8.3 Derivation of the Interval Reduced Hessian Matrix

Here we derive the pair of matrices M^L and M^U whose respective entries are lower and upper bounds on the entries of the NRTL-Gibbs energy equation's projected Hessian matrix. Consider the dimensionless NRTL based Gibbs energy function:

$$\begin{aligned} f(n) &= f_1(n) + f_2(n) \\ f_1(n) &= \sum_{k \in P} \sum_{i \in C} n_{ik} \left(\frac{G_{ik}^f}{RT} + \ln \frac{n_{ik}}{\sum_{j \in C} n_{jk}} \right) \\ f_2(n) &= \sum_{k \in P} \sum_{i \in C} n_{ik} \sum_{j \in C} \frac{\gamma_{ij} \tau_{ij} n_{jk}}{\sum_{p \in C} n_{pk} \gamma_{pj}}, \end{aligned}$$

where $\frac{Gf}{RT}, \gamma$ and τ are constants, and γ is positive. To simplify the notation:

$$\begin{aligned} \text{let } \theta_{ij} &= \gamma_{ij}\tau_{ij} \\ \text{and } g_{ik}(n) &= \sum_{j \in C} \frac{\theta_{ij} n_{jk}}{\sum_{p \in C} n_{pk}\gamma_{pj}}. \end{aligned}$$

The function f_2 can now be expressed as:

$$f_2(n) = \sum_{k \in P} \sum_{i \in C} n_{ik} g_{ik}(n).$$

The first and second partial derivatives of f_2 are:

$$\begin{aligned} \frac{\partial f_2}{\partial n_{lk}} &= g_{lk}(n) + \sum_{i \in C} n_{ik} \frac{\partial g_{ik}(n)}{\partial n_{lk}} \\ \frac{\partial^2 f_2}{\partial n_{lk} \partial n_{mk}} &= \frac{\partial g_{lk}(n)}{\partial n_{mk}} + \frac{\partial g_{mk}(n)}{\partial n_{lk}} + \sum_{i \in C} n_{ik} \frac{\partial^2 g_{ik}(n)}{\partial n_{lk} \partial n_{mk}}, \end{aligned}$$

where,

$$\begin{aligned} \frac{\partial g_{ik}(n)}{\partial n_{lk}} &= \frac{\theta_{il}}{\sum_{p \in C} n_{pk}\gamma_{pl}} + \sum_{j \in C} \frac{\theta_{ij} n_{jk}(-\gamma_{lj})}{\left(\sum_{p \in C} n_{pk}\gamma_{pj}\right)^2} \\ \frac{\partial^2 g_{ik}(n)}{\partial n_{lk} \partial n_{mk}} &= \frac{\theta_{il}(-\gamma_{ml})}{\left(\sum_{p \in C} n_{pk}\gamma_{pl}\right)^2} + \frac{\theta_{im}(-\gamma_{lm})}{\left(\sum_{p \in C} n_{pk}\gamma_{pm}\right)^2} + \sum_{j \in C} \frac{\theta_{ij} n_{jk}(2\gamma_{lj}\gamma_{mj})}{\left(\sum_{p \in C} n_{pk}\gamma_{pj}\right)^3}. \end{aligned}$$

To construct the interval Hessian matrix, lower and upper bounds on the second partial derivatives of f_2 need to be established where n is confined to an n-rectangle $\mathcal{B} = [n^L, n^U]$ where $\mathcal{B}$ is an interval vector and n^L and n^U represent the lower and upper limits on the n vector. Noting that in the NRTL equation γ is always a non-negative number, lower and upper limits on the sum, $s_{jk} = \sum_{p \in C} n_{pk}\gamma_{pj}$, can be established:

$$\begin{aligned} s_{jk}^L &= \sum_{p \in C} n_{pk}^L \gamma_{pj}, \\ s_{jk}^U &= \sum_{p \in C} n_{pk}^U \gamma_{pj}, \\ \text{where } s_{jk}^L &\leq s_{jk}(n) \leq s_{jk}^U \text{ for all } n \in \mathcal{B}. \end{aligned}$$

Using the notation of section 7.5.5 where:

$$\Psi_{jk} = \frac{n_{jk}}{\left(\sum_{p \in C} n_{pk}\gamma_{pj}\right)}$$

equations 7.34 and 7.35 can be used to define lower and upper bounds on Ψ:

$$\Psi^L_{jk} = \frac{n^L_{jk}}{n^L_{jk} + \sum_{p \in C, p \neq j} n^U_{pk}\gamma_{pj}}$$

$$\Psi^U_{jk} = \frac{n^U_{jk}}{n^U_{jk} + \sum_{p \in C, p \neq j} n^L_{pk}\gamma_{pj}}.$$

An interval for $\frac{\partial g_{ik}(n)}{\partial n_{lk}}$ may be computed using the following formulae:

$$\frac{\partial g^L_{ik}(n)}{\partial n_{lk}} = v^L_{il} + \sum_{j \in J^+_i} \theta_{ij}(-\gamma_{lj})\frac{\Psi^U_{ik}}{s^L_{jk}} + \sum_{j \in J^-_i} \theta_{ij}(-\gamma_{lj})\frac{\Psi^L_{ik}}{s^U_{jk}}$$

$$\text{where } v^L_{il} = \begin{cases} \dfrac{\theta_{il}}{s^U_{lk}} & \text{if } l \in J^+_i \\ \dfrac{\theta_{il}}{s^L_{lk}} & \text{if } l \in J^-_i \end{cases}$$

$$J^+_i = \{j : \theta_{ij} \geq 0, j \in C\}$$

$$J^-_i = \{j : \theta_{ij} < 0, j \in C\}$$

$$\frac{\partial g^U_{ik}(n)}{\partial n_{lk}} = v^U_{il} + \sum_{j \in J^+} \theta_{ij}(-\gamma_{lj})\frac{\Psi^L_{ik}}{s^U_{jk}} + \sum_{j \in J^-} \theta_{ij}(-\gamma_{lj})\frac{\Psi^U_{ik}}{s^L_{jk}}$$

$$\text{where } v^U_{il} = \begin{cases} \dfrac{\theta_{il}}{s^L_{lk}} & \text{if } l \in J^+_i \\ \dfrac{\theta_{il}}{s^U_{lk}} & \text{if } l \in J^-_i \end{cases}.$$

Similarly an interval for $\frac{\partial^2 g_{ik}(n)}{\partial n_{lk}\partial n_{mk}}$ may be computed using:

$$\frac{\partial^2 g^L_{ik}(n)}{\partial n_{lk}\partial n_{mk}} = w^L_{iml} + w^L_{ilm} + \sum_{j \in J^+_i} \theta_{ij}(2\gamma_{lj}\gamma_{mj})\frac{\Psi^L_{ij}}{\left(s^U_{jk}\right)^2} + \sum_{j \in J^-_i} \theta_{ij}(2\gamma_{lj}\gamma_{mj})\frac{\Psi^U_{ij}}{\left(s^L_{jk}\right)^2}$$

$$\text{where } w^L_{ilm} = \begin{cases} \dfrac{\theta_{il}(-\gamma_{ml})}{\left(s^L_{jl}\right)^2} & \text{if } l \in J^+_i \\ \dfrac{\theta_{il}(-\gamma_{ml})}{\left(s^U_{jl}\right)^2} & \text{if } l \in J^-_i \end{cases}$$

$$\frac{\partial^2 g^U_{ik}(n)}{\partial n_{lk}\partial n_{mk}} = w^U_{iml} + w^U_{ilm}$$

$$+\sum_{j\in J_i^+}\theta_{ij}(2\gamma_{lj}\gamma_{mj})\frac{\Psi_{ij}^U}{\left(s_{jk}^L\right)^2}+\sum_{j\in J_i^-}\theta_{ij}(2\gamma_{lj}\gamma_{mj})\frac{\Psi_{ij}^L}{\left(s_{jk}^U\right)^2}$$

$$\text{where } w_{ilm}^U = \begin{cases} \dfrac{\theta_{il}(-\gamma_{ml})}{\left(s_{jl}^U\right)^2} & \text{if } l\in J_i^+ \\ \\ \dfrac{\theta_{il}(-\gamma_{ml})}{\left(s_{jl}^L\right)^2} & \text{if } l\in J_i^- \end{cases}$$

From these relations an interval for $\frac{\partial^2 f_2}{\partial n_{lk}\partial n_{mk}}$ can be derived:

$$\frac{\partial^2 f_2^L}{\partial n_{lk}\partial n_{mk}} = \frac{\partial g_{lk}^L(n)}{\partial n_{mk}}+\frac{\partial g_{mk}^L(n)}{\partial n_{lk}}+\sum_{i\in C}u_{ilm}^L$$

$$\text{where } u_{ilm}^L = \begin{cases} n_{ik}^U\dfrac{\partial^2 g_{ik}^L(n)}{\partial n_{lk}\partial n_{mk}} \text{ if } \dfrac{\partial^2 g_{ik}^L(n)}{\partial n_{lk}\partial n_{mk}}<0 \\ \\ n_{ik}^L\dfrac{\partial^2 g_{ik}^L(n)}{\partial n_{lk}\partial n_{mk}} \text{ if } \dfrac{\partial^2 g_{ik}^L(n)}{\partial n_{lk}\partial n_{mk}}\geq 0 \end{cases}$$

$$\frac{\partial^2 f_2^U}{\partial n_{lk}\partial n_{mk}} = \frac{\partial g_{lk}^U(n)}{\partial n_{mk}}+\frac{\partial g_{mk}^U(n)}{\partial n_{lk}}+\sum_{i\in C}u_{ilm}^U$$

$$\text{where } u_{ilm}^U = \begin{cases} n_{ik}^L\dfrac{\partial^2 g_{ik}^U(n)}{\partial n_{lk}\partial n_{mk}} \text{ if } \dfrac{\partial^2 g_{ik}^U(n)}{\partial n_{lk}\partial n_{mk}}<0 \\ \\ n_{ik}^U\dfrac{\partial^2 g_{ik}^U(n)}{\partial n_{lk}\partial n_{mk}} \text{ if } \dfrac{\partial^2 g_{ik}^U(n)}{\partial n_{lk}\partial n_{mk}}\geq 0 \end{cases}$$

The second partial derivatives for f_1 are:

$$\frac{\partial f_1}{\partial n_{lk}\partial n_{mk}} = \delta_{lm}\frac{1}{n_{lk}}-\frac{1}{\sum_{i\in C}n_{ik}}$$

$$\text{where } \delta_{lm} = \begin{cases} 1 & \text{if } l=m \\ 0 & \text{if } l\neq m \end{cases}$$

Noting that when $l=m$:

$$\frac{\partial f_1}{\partial n_{lk}\partial n_{mk}}=\frac{1}{n_{lk}}-\frac{1}{\sum_{i\in C}n_{ik}}=\frac{\sum_{i\neq l}n_{ik}}{n_{lk}\sum_{i\in C}n_{ik}}$$

it is clear that n_{lk} occurs in the right hand side's denominator, but is absent from its numerator, hence the following bounds can be derived:

$$\frac{\partial f_1^L}{\partial n_{lk}\partial n_{mk}} = \begin{cases} \dfrac{1}{n_{lk}^U}-\dfrac{1}{n_{lk}^U\sum_{i\neq l}n_{ik}^L} & \text{when } l=m \\ \\ -\dfrac{1}{\sum_{i\in C}n_{ik}^L} & \text{when } l\neq m \end{cases}$$

$$\frac{\partial f_1^U}{\partial n_{lk}\partial n_{mk}} = \begin{cases} \frac{1}{n_{lk}^L} - \frac{1}{n_{lk}^L \sum_{i\neq l} n_{ik}^U} & \text{when } l = m \\ -\frac{1}{\sum_{i\in C} n_{ik}^U} & \text{when } l \neq m \end{cases}$$

The bounds on the second derivatives of f can be expressed as:

$$\frac{\partial f^U}{\partial n_{lk}\partial n_{mk}} = \frac{\partial f_1^U}{\partial n_{lk}\partial n_{mk}} + \frac{\partial f_2^U}{\partial n_{lk}\partial n_{mk}}$$

$$\frac{\partial f^L}{\partial n_{lk}\partial n_{mk}} = \frac{\partial f_1^L}{\partial n_{lk}\partial n_{mk}} + \frac{\partial f_2^L}{\partial n_{lk}\partial n_{mk}}.$$

For a two phase non-reacting system the A matrix is of the form:

$$\begin{bmatrix} I_{|C|} & I_{|C|} \end{bmatrix}$$

where $I_{|C|}$ is the $|C| \times |C|$ identity matrix,

and $|C|$ is the cardinality of the set of species C.

A matrix Z whose columns form a basis for the null-space of A can take the form:

$$Z = \begin{bmatrix} I_{|C|} \\ -I_{|C|} \end{bmatrix}.$$

The Hessian matrix for a two phase system has the structure:

$$H = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}$$

$$\text{where } H_1 = \begin{bmatrix} \frac{\partial f}{\partial n_{11}\partial n_{11}} & \cdots & \frac{\partial f}{\partial n_{11}\partial n_{|C|1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial n_{|C|1}\partial n_{11}} & \cdots & \frac{\partial f^U}{\partial n_{|C|1}\partial n_{|C|1}} \end{bmatrix},$$

$$H_2 = \begin{bmatrix} \frac{\partial f}{\partial n_{12}\partial n_{12}} & \cdots & \frac{\partial f}{\partial n_{12}\partial n_{|C|2}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial n_{|C|2}\partial n_{12}} & \cdots & \frac{\partial f^U}{\partial n_{|C|2}\partial n_{|C|2}} \end{bmatrix}.$$

The projected Hessian can be written as:

$$\begin{aligned} Z^T H Z &= \begin{bmatrix} I_{|C|} - I_{|C|} \end{bmatrix} \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix} \begin{bmatrix} I_{|C|} \\ -I_{|C|} \end{bmatrix} \\ &= H_1 + H_2, \end{aligned}$$

hence the upper and lower bounds on the projected Hessian are:

$$M^U = \begin{bmatrix} \frac{\partial f^U}{\partial n_{11}\partial n_{11}} + \frac{\partial f^U}{\partial n_{12}\partial n_{12}} & \cdots & \frac{\partial f^U}{\partial n_{11}\partial n_{|C|1}} + \frac{\partial f^U}{\partial n_{12}\partial n_{|C|2}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f^U}{\partial n_{|C|1}\partial n_{11}} + \frac{\partial f^U}{\partial n_{|C|2}\partial n_{12}} & \cdots & \frac{\partial f^U}{\partial n_{|C|1}\partial n_{|C|1}} + \frac{\partial f^U}{\partial n_{|C|2}\partial n_{|C|2}} \end{bmatrix}$$

$$M^L = \begin{bmatrix} \frac{\partial f^L}{\partial n_{11}\partial n_{11}} + \frac{\partial f^L}{\partial n_{12}\partial n_{12}} & \cdots & \frac{\partial f^L}{\partial n_{11}\partial n_{|C|1}} + \frac{\partial f^L}{\partial n_{12}\partial n_{|C|2}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f^L}{\partial n_{|C|1}\partial n_{11}} + \frac{\partial f^L}{\partial n_{|C|2}\partial n_{12}} & \cdots & \frac{\partial f^L}{\partial n_{|C|1}\partial n_{|C|1}} + \frac{\partial f^L}{\partial n_{|C|2}\partial n_{|C|2}} \end{bmatrix}.$$

7.8.4 Role of the Convexity Check in the Global Optimization Algorithm

The convexity check is used to establish if the Gibbs energy minimization problem is convex when constrained to a subregion of the y-domain. This test can be carried out during the relaxed dual phase once the current y-subregion has been partitioned, before the relaxed dual problem is solved. If the test yields a negative result the algorithm proceeds with solving the relaxed dual subproblem as usual. If the test shows the current region to be convex then the following Gibbs energy minimization problem is solved:

$$\begin{aligned} \min_n G(n) & \qquad (7.39) \\ \text{subject to } An &= b \\ n^L &\leq n \leq n^U \end{aligned}$$

where n^U and n^L are the bounds imposed by the qualifying constraints for the relaxed dual subproblem. A modified global optimization algorithm replaces the relaxed dual phase step 3 with the following step:

New step 3: Convexity Check and Relaxed Dual Phase

Partition the region defined by $\mathcal{R}_{k_c}$ by placing hyperplanes through n^{k_c};

Store the bounds defining each subregion B_i as the interval vector $\mathcal{B}_{B_i}$.

for all $B_i \in CB$

Carry out the convexity test with the bounds on n specified by $\mathcal{B}_{B_i}$

if the region is shown to be convex;

Solve the domain constrained Gibbs energy minization problem 7.39;

Compare the optimal objective value, P^C ,with the present least upper bound P^U, if $P^C < P^U$ then let $P^U = P^C$.

else if the region is not shown to be convex;

Calculate bounds on the x-variables $\hat{\Psi}^k$ and $\hat{\Psi}^K$ using equations 7.36 and solve the relaxed dual subproblem to give ξ^* and n^*;

If $\xi^* \geq P^U$, then fathom the subregion,

If $\xi^* < P^U$, then set $k_s = k_s + 1$, $p(k_s) = k_c, I_{k_s} = K, \xi^{k_s} = \xi^*, \bar{n}^{k_s} = n^*$ and $\mathcal{R}_{k_s} = \mathcal{B}_{B_i}$

end if

end

All other aspects of the modified global optimization algorithm are the same as the original one.

7.8.5 Numerical Tests

To test the effectiveness of this strategy the test problems described in section 7.7 were solved using the new scheme. In all cases the solutions were the same as those reached by the original global optimization algorithm. Figure 7.15 shows a comparison between CPU times required by the original and the modified global optimization algorithms. The problem identification numbers correspond to the following problems:

1. Test problem 1: water + n-butyl acetate
2. Test problem 2: toluene + water + aniline
3. Test problem 3, feed 1: n-propanol + n-butanol + water
4. Test problem 4: ethanol + ethyl acetate + water
5. Test problem 5: n-butanol + water + n-butyl acetate

The problems were solved to within an ϵ-optimality convergence tolerance of 10^{-6}. Test problem 3, feed 2 proved too difficult for either method to converge to within an allocated maximum of 5000 major iterations.

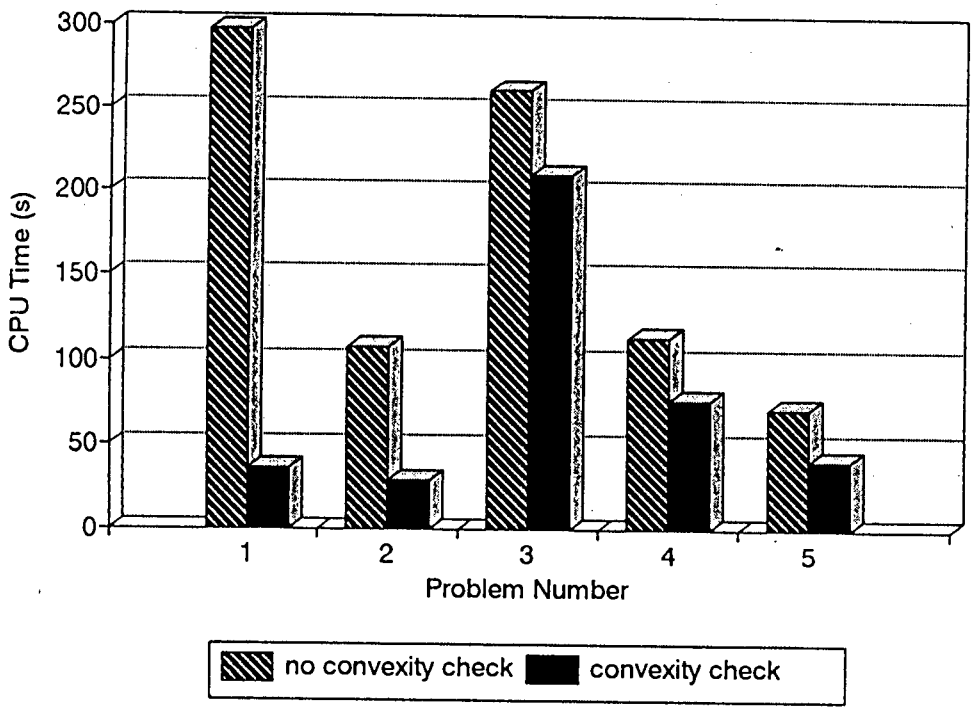

Figure 7.15: Comparison of CPU times for the global optimization algorithm variants

The effectiveness of the convexity test in reducing CPU time is subject to a trade-off between the accuracy with which the relaxed dual problems are capable of estimating the global minimum and the amount of convexity that may be exploited. A global optimization algorithm's ability to exclude regions in the y-domain from further consideration can be assessed from the number of unfathomed leaf nodes that exist at various stages of the optimization process. The fathoming which takes place through the relaxed dual problem mechanism can be compared with the convexity test fathoming by comparing the number of unfathomed leaf nodes generated by the original algorithm with the number generated by the modified algorithm. Figures 7.16 to 7.22 show the number of unfathomed leaf nodes as a function of the iteration number. In all cases besides feed 2 of problem 3 the convexity test displayed great success in the fathoming of nodes.

The typical trend is that the convexity test algorithm matches the unmodified algorithm in the early stages of a run. At a critical stage the y-subregions become small enough for the convexity test to start fathoming and it is here that the course of the respective algorithms diverge. The success of the convexity test in reducing the CPU time of a run, is dependent on the value of the convergence tolerance test parameter at the stage at which the convexity test starts yielding results. The steady rise in the number of unfathomed leaf nodes for feed 2 of problem 3, displayed in figure 7.19 indicates that relatively few regions are being fathomed by either algorithm. In this problem the difference in $\hat{G}$ between the trivial solution and the global optimum is of the order of 10^{-6} , which is not conducive to a high fathoming rate in the unmodified algorithm. The convexity test is not successful for this problem either as the mole fraction composition of the global solution is similar to that of the trivial solution, and at the trivial solution there is no convexity to exploit. This is because the Gibbs energy function is constant along the locus of the trivial solutions, hence the curvature of the function in the direction of the locus is zero. This phenomenon is clearly shown in figure 7.9 where along the trivial solution locus valley there is no curvature.

As problem 1 involves 2 chemical species and the rest involve 3, it appears that the dimension of the problem is another important factor which influences the effectiveness of the convexity test. As the dimension of the problem increases, the number of steps required to show convexity increases as well. Due to the approximate nature of the test, the accuracy of the interval matrix $\mathcal{M}^{(k)}$ as an approximation of the range of values accessible to $\tilde{H}^{(k)}(x)$ decreases as k increases. Consequently, the test becomes more conservative for problems of greater dimension.

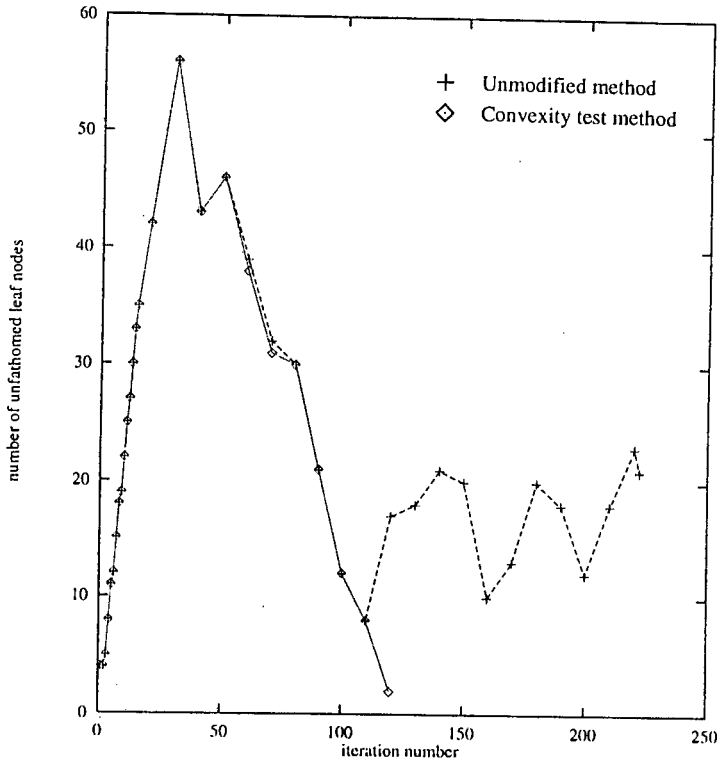

Figure 7.16: Comparison of leaf node numbers generated by global optimization algorithm variants: problem 1

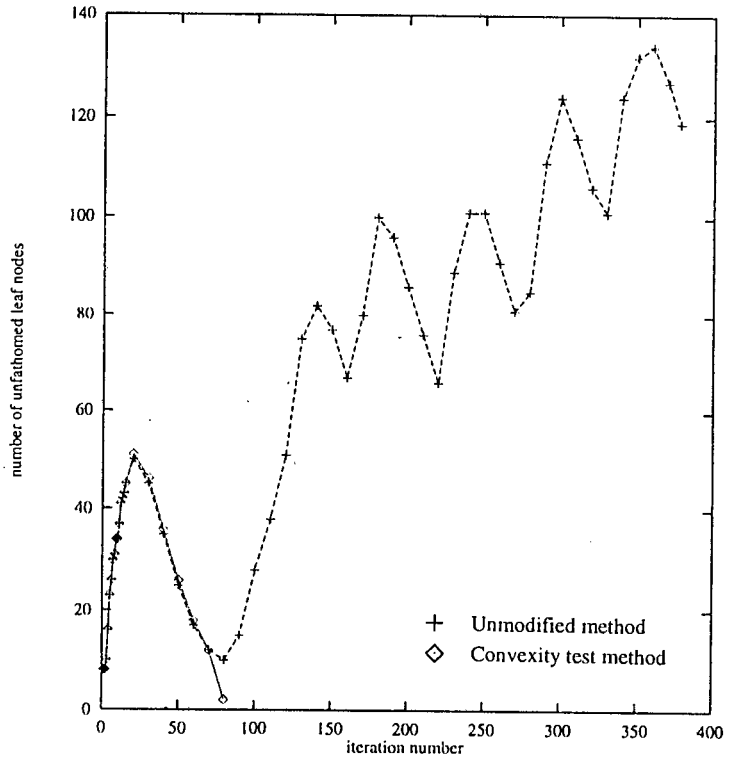

Figure 7.17: Comparison of leaf node numbers generated by global optimization algorithm variants: problem 2

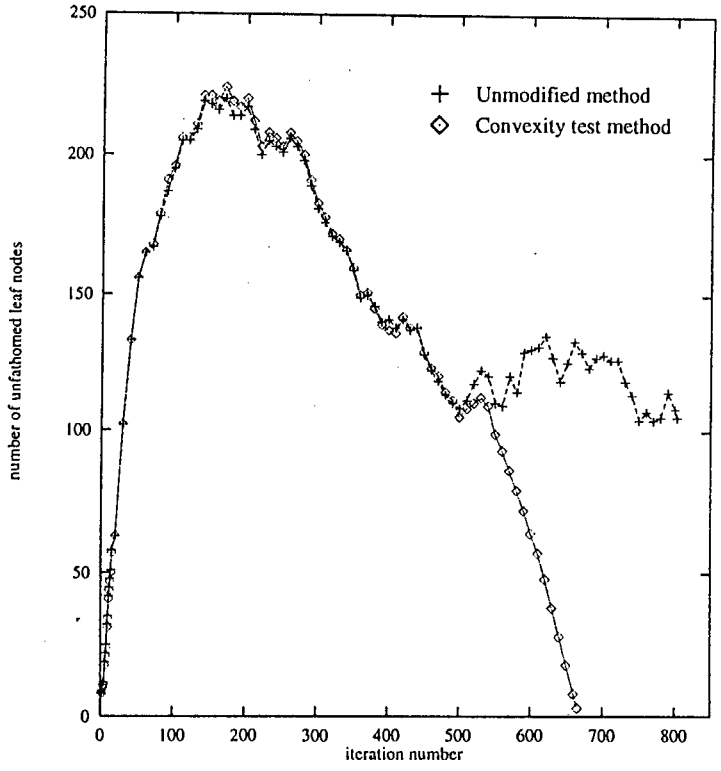

Figure 7.18: Comparison of leaf node numbers generated by global optimization algorithm variants: problem 3, feed 1

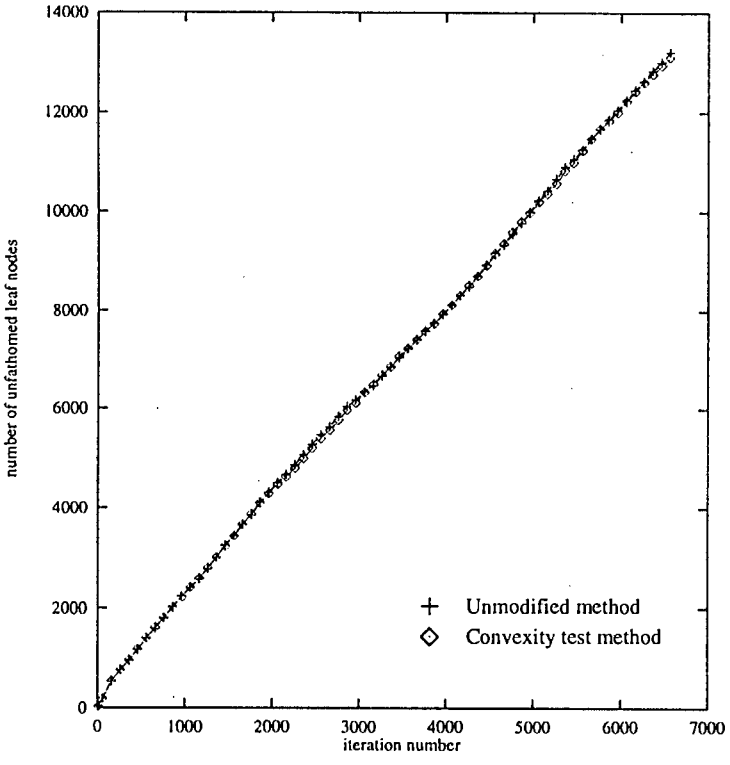

Figure 7.19: Comparison of leaf node numbers generated by global optimization algorithm variants: problem 3, feed 2

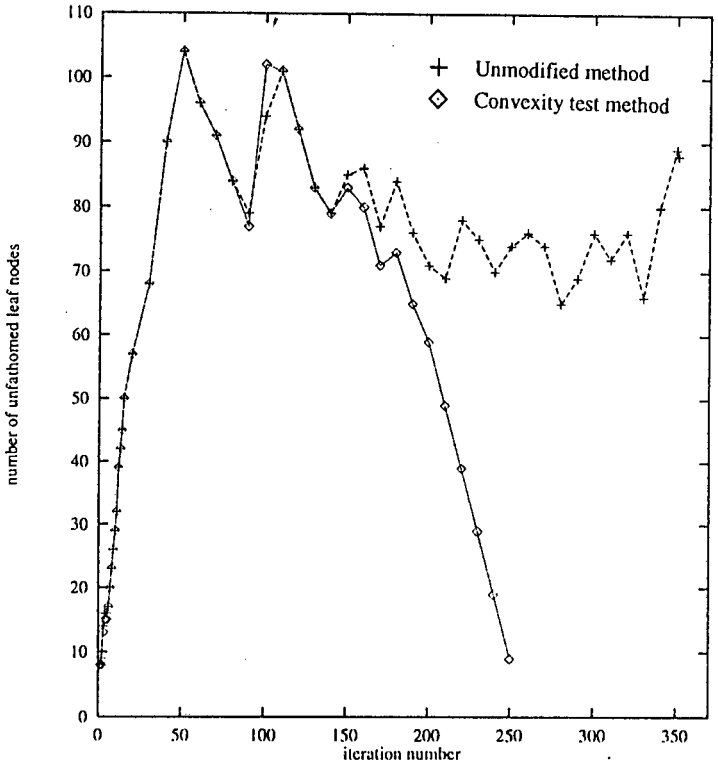

Figure 7.20: Comparison of leaf node numbers generated by global optimization algorithm variants: problem 4

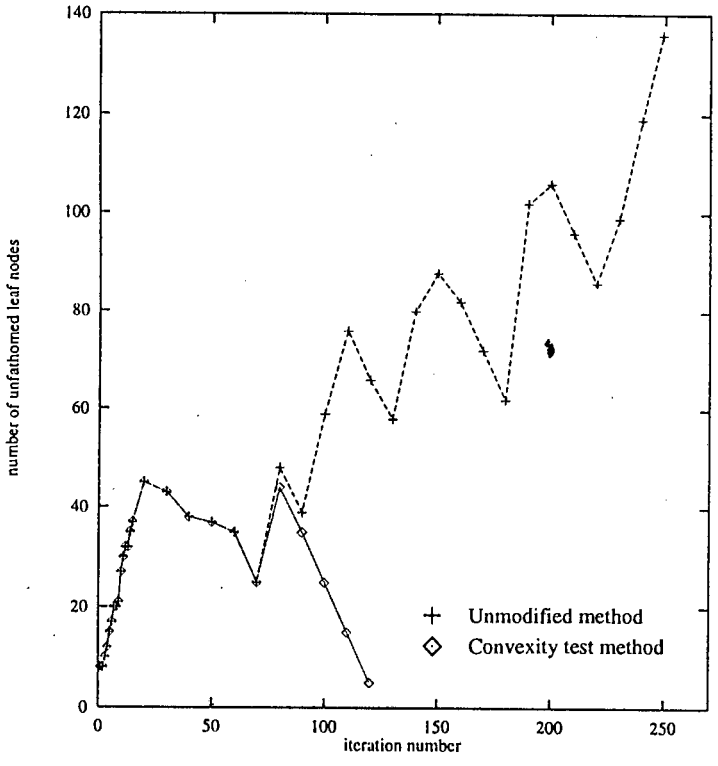

Figure 7.21: Comparison of leaf node numbers generated by global optimization algorithm variants: problem 5

Chapter 8

Heuristic Global Optimization Strategies

Heuristic methods for determining the true equilibrium point were developed before rigorous global optimization methods were applied to the equilibrium problem. Unable to *guarantee* the global optimality of a solution, these methods, nevertheless, have beneficial features which are absent from the rigorous approaches: they are simple to understand and simple to code, they are fast relative to the rigorous global optimization approaches, and they require a modest amount of storage space. The utility of such methods depends, however, on the frequency with which they correctly determine the true equilibrium point. Swank and Mullins [76], in a review of these methods, found Michelsen's tangent plane method to be completely reliable for the set of test problems used in the study. Other methods reviewed included Gautam and Seider's phase splitting method [23], which proved to be less reliable than Michelsen's approach [48]. The numerical methods used in this study were ,however, tuned for efficiency not reliability, and the results therefore may not represent the best attainable performance of the alternative methods in terms of reliability. In this chapter, another look is taken at heuristic methods for phase equilibrium computations, this time with the emphasis being on reliability rather the speed.

8.1 Stability Criteria as a Means of Identifying a True Equilibrium Point

Stability is a thermodynamic concept concerning the physical possibility of phase's existence. There are two definitions for stability for isothermal, isobaric phases. A phase is:

1. intrinsically unstable if an infinitesimal perturbation in composition can result in the lowering of the Gibbs energy;
2. metastable if it is not intrinsically unstable, yet is capable of existing in a lower Gibbs energy state.

No phase in an intrinsically unstable state can remain as a single phase, and will inevitably split into two phases; under carefully controlled conditions, however, it is possible to maintain a metastable phase. An intrinsically unstable phase can be identified via the eigenvalues of the projected Hessian matrix; one or more negative eigenvalues indicates intrinsic instability. Metastability is far more difficult to detect as it cannot be discovered by methods based purely on the local topology of the Gibbs energy surface. For practical purposes only the true stable equilibrium point is of importance; metastable and intrinsically unstable phases will therefore be referred to as unstable.

The tangent plane test for phase instability was originally suggested by Gibbs and formalized by Baker et al. [1]. Michelsen [48] motivates this test through the consideration of an isobaric, isothermal single phase system with a mole fraction composition $(z_1, z_2, ..., z_N)$. The Gibbs energy of this phase is

$$G^0 = n^0_{tot} \sum_{i=1}^{N} z_i \mu^0_i$$

where μ^0_i is the chemical potential of chemical i in the phase and n^0_{tot} is the total number of moles present the phase. Now consider the effect of phase splitting on the system where the molar extent of phase 2 is ϵ and that of phase 1 is $n^0_{tot} - \epsilon$, where ϵ is infinitesimal. Let the mole fractions in the second phase be $(y_1, y_2, ..., y_N)$. The change in the Gibbs energy corresponding to the phase split is then

$$\Delta G = G^1 + G^2 - G^0$$

A first order Taylor series expansion of G^1 is

$$G^1 = G^0 - \epsilon \sum_{i=1}^{N} y_i \left(\frac{\partial G}{\partial n_i} \right)_{n^0} = G^0 - \epsilon \sum_{i=1}^{N} y_i \mu_i^0$$

and the Gibbs energy for the second phase is

$$G^2 = \epsilon \sum_{i=1}^{N} y_i \left(\frac{\partial G}{\partial n_i} \right)_{n^2} = \epsilon \sum_{i=1}^{N} y_i \mu_i(y)$$

The Gibbs energy change is therefore

$$\Delta G = \epsilon \sum_{i=1}^{N} y_i \left(\mu_i(y) - \mu_i^0 \right)$$

The single phase system is stable if the Gibbs energy cannot be reduced by a phase split. A necessary condition for phase stability is therefore

$$f(y) = \sum_{i=1}^{N} y_i \left(\mu_i(y) - \mu_i^0 \right) \geq 0, \qquad \text{for all } y \geq 0 \text{ subject to } \sum_{i=1}^{N} y_i = 1$$

Geometrically $f(y)$ is the vertical distance from the hyperplane, tangent to the Gibbs energy surface at composition z, to the Gibbs energy surface at composition y. Figures 8.1 and 8.2 illustrate the tangent plane concept for a binary system. On the x-axis is the mole fraction y of one of the species. The composition z in figure 8.1 is the feed composition, the solid line is the hyperplane which is tangent to the Gibbs energy curve at z, and the dashed line is parallel to the tangent plane and is coincident with the Gibbs energy curve at point $(y^*, G(y^*))$. Arrows labelled $f(y_1)$ and $f(y^*)$ represent the vertical distance between the tangent plane and the Gibbs energy function at y_1 and y^* respectively. The minimizer of $f(y)$ is y^*. As the Gibbs energy function lies below the tangent plane at y^* the system is unstable. Figure 8.2 represents the case where the single phase system of composition z depicted in figure 8.1 has split into two phases of composition z_1 and z_2 respectively. The isoactivity criterion for equilibrium between two phases results in the plane tangent to the Gibbs energy function at z_1 being coincident with the plane tangent to the Gibbs energy function at z_2. As the Gibbs energy function lies nowhere below the tangent plane, $f(y)$ is positive for all compositions, hence the system is stable.

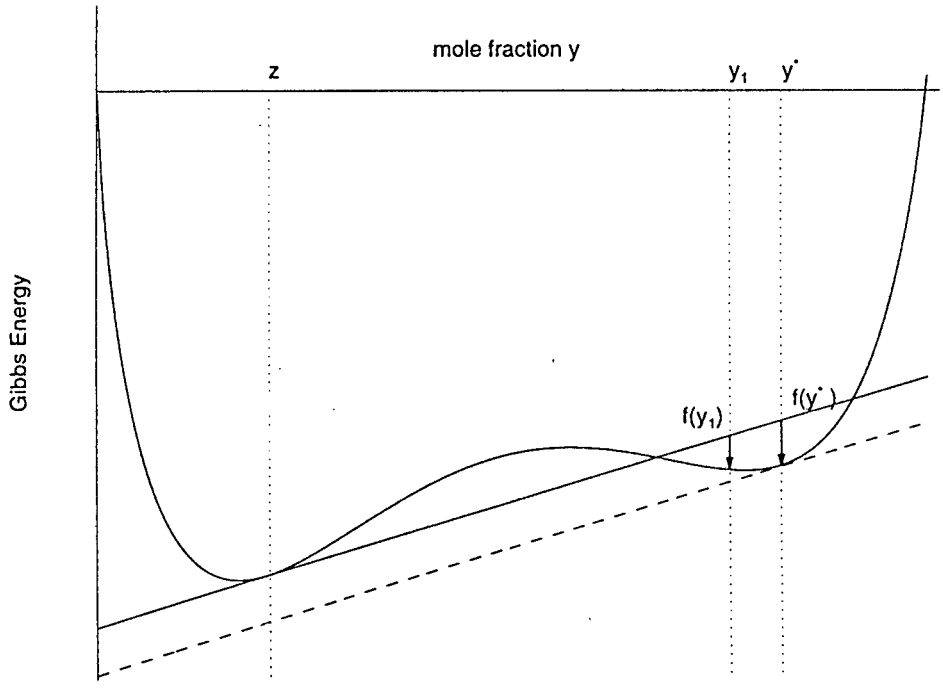

Figure 8.1: Tangent hyperplane and Gibbs energy function for an unstable system

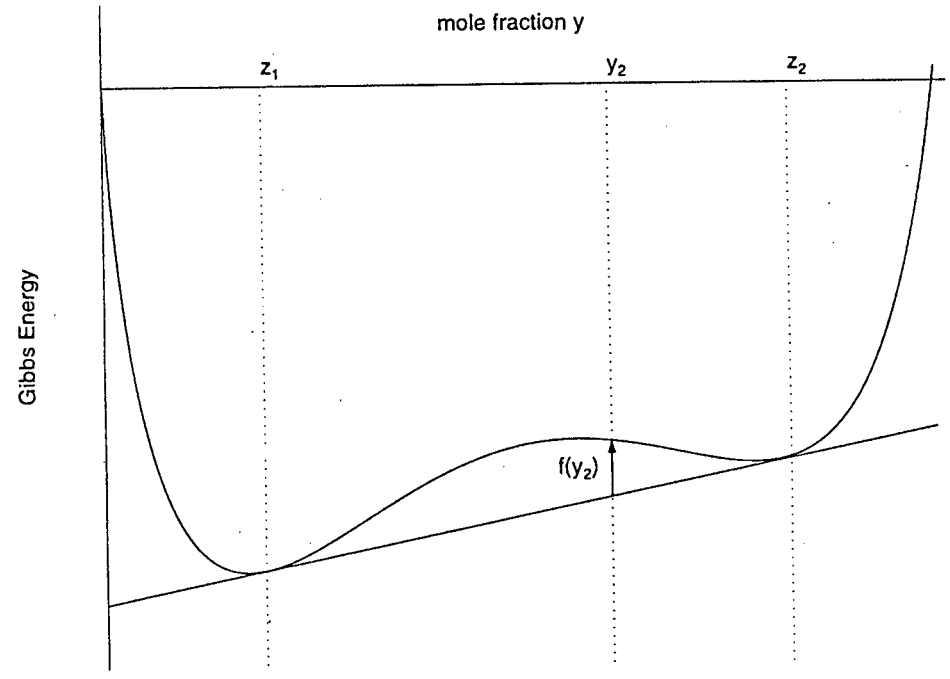

Figure 8.2: Tangent hyperplane and Gibbs energy function for a stable system

Smith et al. [69] generalized the tangent plane concept to allow for reacting systems and for systems where some phases are not accessable to all species. Here we will consider only the extension of the concept to reacting systems ignoring the possibility of species not existing in all phases. Smith et al. based the criterion on the following formulation of the chemical equilibrium problem:

$$\min_{x_{ik}\tilde{n}_k} G = \sum_{k\in P}\tilde{n}_k\sum_{i\in C}x_{ik}\mu_{ik} \tag{8.1}$$

subject to:

$$\tilde{n}_k \geq 0 \qquad k\in P \tag{8.2}$$

$$x_{ik} \geq 0 \qquad i\in C, k\in P \tag{8.3}$$

$$\sum_{k\in P}\tilde{n}_k\sum_{i\in C}a_{ji}x_{ik} - b_j = 0 \qquad j\in E \tag{8.4}$$

$$\sum_{i\in C}x_{ik} - 1 = 0 \qquad k\in P \tag{8.5}$$

where $\tilde{n}_k$ is the total number of moles in phase k. At a local minimum the following necessary optimality conditions are satisfied:

$$\begin{aligned}
&\nabla_{\tilde{n}x}\left(\sum_{k\in P}\tilde{n}_k\sum_{i\in C}x_{ik}\mu_{ik}\right)\\
&\quad -\sum_{k\in P}\theta_k\nabla_{\tilde{n}x}\left(\tilde{n}_k\right)\\
&\quad -\sum_{k\in P}\sum_{i\in C}\varpi_{ik}\nabla_{\tilde{n}x}\left(x_{ik}\right)\\
&-\sum_{j\in M}\lambda_j\nabla_{\tilde{n}x}\left(\sum_{k\in P}\tilde{n}_k\sum_{i\in C}a_{ji}x_{ik} - b_j\right)\\
&\quad -\sum_{k\in P}\rho_k\nabla_{\tilde{n}x}\left(\sum_{i\in C}x_{ik} - 1\right) = 0\\
&\theta_k\tilde{n}_k = 0 \qquad k\in P\\
&\varpi_{ik}x_{ik} = 0 \qquad i\in C, k\in P\\
&\theta_k \geq 0 \qquad k\in P\\
&\varpi_{ik} \geq 0 \qquad i\in C, k\in P
\end{aligned}$$

where $\theta_k, \varpi_k, \lambda_j$ and ρ_k are Lagrange multipliers associated with constraints 8.2, 8.3, 8.4 and 8.5 respectively. Smith et al. show that a Kuhn-Tucker point is a global minimum only if the following criterion obtains:

$$\sum_{i\in C}x_i\left(\mu_i\left(x\right) - \sum_{j\in E}\lambda_j^* a_{ji}\right) \geq 0 \quad \text{for all } x \text{ satisfying 8.3 and 8.5,} \tag{8.6}$$

where λ^* denotes the Lagrange multiplier λ at the Kuhn-Tucker point. Where no reaction occurs this criterion simplifies to the Gibbs tangent plane criterion.

When a phase has been found to be unstable, any composition x which does not satisfy 8.6 can be used as an initial estimate for the new phase in the equilibrium computation. Provided the new phase is initialized with a sufficiently small total molar quantity, the introduction of this phase will result in a decrease in the systems Gibbs energy.

8.1.1 Experimental Implementation of the Tangent Plane Criterion

To assess the ability of tangent plane based strategies to guide an equilibrium computation to the correct solution, a method was implemented based directly on a minimization formulation of the problem. This implementation is summarized below, while the FORTRAN code is to be found in appendix D.

Step1: **do for** $i = 1$ to N

Initialize the search variables

$$y_j = 0.99 \qquad j = i$$
$$y_j = \frac{0.01}{N-1} \qquad \forall\ j \neq i$$

solve the minimization problem:

$$\min_y f(y) = \sum_{i=1}^{N} y_i \left(\mu_i(y) - \mu_i^0(z)\right)$$
$$\text{subject to} \quad \sum_{i=1}^{N} y_i = 1$$
$$0 \leq y_i \leq 1 \qquad i = 1, 2, ..., N$$

and store the minimizer y^*.

end do

Step 2: **if** a negative tangent plane distance was found , $f(y^*) < 0$:

attempt to reduce the Gibbs energy of the system by minimizing the Gibbs energy function using a non-global nonlinear programming approach, initializing the incipient phase so that it has the same composition as the tangent distance minimizer,

return to **step 1;**

else stop.

Note that this method of determining the stability of a system is not rigorous because the objective function $f(y)$ is usually nonconvex and the global optimum is sought via a standard

nonlinear programming technique started from a number of points to improve the probability of the global minimum being found. The optimization software package MINOS 5.4 was used to solve all the minimization problems encountered in this scheme.

8.2 Phase Splitting Methods

A more direct approach to discovering the global minimum of the Gibbs energy surface is by the "phase splitting" approach where a *source* phase is split into two *trial* phases of carefully selected composition with the hope that the minimization method, starting from these compositions, will produce the required result. Such methods resemble the methods based on stability criteria; in both cases a good initial guess is required if the minimization method is to succeed. The determination of the trial phases' compositions is therefore the cornerstone of phase splitting methods.

8.2.1 Gautam and Seider's Method

Gautam and Seider's method [22] is inspired by an earlier method used by Boston and Fournier [5] in which the trial phases are initialized via an approximate solution method based on activities at infinite dilution. Instead of using activities at infinite dilution, Gautam and Seider used the activities computed at the composition of the source phase, constructing their initialization procedure on the following heuristics:

1. phase splitting occurs as a result of interaction between a pair of chemical species;
2. the magnitude of the activity associated with a species is indicative of its propensity to cause the phase to split.

The approximate solution method proposed by Gautam and Seider to affect the splitting of a source phase s into trial phases $t1$ and $t2$ is summarized below:

Step 1: find the chemical species α with the highest activity in source phase s.

Step 2: find the chemical species β with the highest activity in a binary phase with α where the ratio of the compositions of α and β is the same as that in the source phase.

Step 3: allocate the chemicals α and β to two new trial phases $t1$ and $t2$ using the isoactivity equations:

$$\begin{aligned} \gamma_\alpha^{t1} x_\alpha^{t1} &= \gamma_\alpha^{t2} x_\alpha^{t2} \\ \gamma_\beta^{t1} x_\beta^{t1} &= \gamma_\beta^{t2} x_\beta^{t2} \end{aligned}$$

where γ_α^{t1} is the activity coefficient of chemical species α in phase $t1$ and x_α^{t1} is the mole fraction of this species:

$$x_\alpha^{t1} = \frac{n_\alpha^{t1}}{n_\alpha^{t1} + n_\beta^{t1}}$$

Step 4: **do while** there are undistributed species

To the trial phases, allocate the chemical with the next highest binary activity with α according to the isoactivity criterion, keeping constant the molar quantities of the species already allocated.

end do

Step 5: Minimize the Gibbs energy starting at the trial composition.

if a new minimum is found return to step 1 and attempt to split each of the source phases;

else stop.

8.2.2 Experimental Implementation of Gautam and Seider's Method

The FORTRAN implementation used to assess the reliability of the Gautam and Seider method does not follow the original algorithm exactly, instead it follows an adaptation designed to improve its success rate. A weakness in the original algorithm is the possibility of a trivial solution being found in step 3, either as a result of convergence to the incorrect stationary point or because of the absence of a nontrivial solution. Limiting the species β to being in the subset of species that can form an unstable binary phase with species α is a modification suggested by Walraven and van Rompay to correct this problem. Additional measures they suggested are incorporated into the following step which replaces step 3.

New step 3: Find a binary mixture containing α and β with a composition where the Gibbs free energy as a function of mole fraction is concave, by minimizing the second derivative of the Gibbs energy function:

$$\frac{d^2 G(x_\alpha, x_\beta)}{dx_\alpha^2}$$

where $x_\beta = 1 - x_\alpha$.

Determine the equilibrium composition of this binary system by minimizing the Gibbs free energy. Allocate the species α and β to the trial phases by mass balance in such a way that the mole fractions of the trial phases are the same as those in the binary system. This is done by solving the system of linear equations:

$$\begin{bmatrix} x_\alpha^{t1} & x_\alpha^{t2} \\ x_\beta^{t1} & x_\beta^{t2} \end{bmatrix} \begin{bmatrix} n^{t1} \\ n^{t2} \end{bmatrix} = \begin{bmatrix} n_\alpha \\ n_\beta \end{bmatrix}.$$

The nonlinear equations defining the isoactivity conditions are solved by using MINOS 5.4 to solve a minimization problem in which the objective function is the square error on the isoactivity criterion: $\left(\gamma_i^{t1} x_i^{t1} - \gamma_i^{t2} x_i^{t2}\right)^2$. This method may not be the fastest means of solving the isoactivity equations yet it is a reliable and faithful representation of Gautam and Seider's approach.

8.2.3 Walraven and van Rompay's Method

Walraven and van Rompay [80] found that the phase composition estimates given by the Gautam and Seider algorithm can be far from the exact solution. Walraven and van Rompay improve on the Gautam and Seider algorithm by using the following heuristics:

1. the compositions given by the Gautam and Seider algorithm tend to lie close to the straight line passing through the source composition and the equilibrium compositions.
2. a construction of the line tangential to the Gibbs energy function yields good approximations of the equilibrium composition.

Walraven and van Rompay use these heuristics in the algorithm stated below. The algorithm statement is followed by an example which illustrates its motivation.

Step 1 Apply Gautam and Seider's algorithm to compute trial phase compositions x^{t1} and x^{t2}.

Step 2 Allocate α then all other species i via the isoactivity criterion in order of decreasing binary activity with α:

$$\gamma_i^{it1} x_i^{it1} = \gamma_i^{it2} x_i^{it2}.$$

The improved trial phase compositions are denoted $x^{it1}$and x^{it2}. This step is essentially a repeat of the isoactivity allocation process carried out in step 1. Note, however,

that the isoactivity equations in step 1 and in step 2 are solved species by species, not simultaneously therefore the step 2 compositions are a refinement of the trial phase compositions computed in step 1.

Step 3 Check the following conditions:

$$\frac{x_\alpha^{t1} - x_\alpha^s}{x_\alpha^{it1} - x_\alpha^s} \leq 1 \tag{8.7}$$

$$\frac{x_\alpha^{t2} - x_\alpha^s}{x_\alpha^{it2} - x_\alpha^s} \leq 1 \tag{8.8}$$

where x^s denotes the mole fraction composition of the source phase which is being split. If one or both of these conditions is not met, attempt to meet them while maintaining feasibility by displacing $x^{it1}$and x^{it2} along the line they define. This step attempts to keep $x^{it1}$and x^{it2} outside of the immiscible region.

Step 4.1 Find point $x^{\tan 1}$ located between x^s and x^{it1} which together with x^s defines a line passing through $(x^{\tan 1}, G(x^{\tan 1}))$ and $(x^s, G(x^s))$ which is tangent to the Gibbs energy curve at $(x^{\tan 1}, G(x^{\tan 1}))$.

If $x^{\tan 1} = x^s$, continue to **step 4.2** otherwise go to **step 4.1.1.**

Step 4.1.1 Find point $x^{\tan 2}$ located between $x^{\tan 1}$ and x^{it2} which together with $x^{\tan 1}$ defines a line passing through $(x^{\tan 2}, G(x^{\tan 2}))$ and $(x^{\tan 1}, G(x^{\tan 1}))$ which is tangent to the Gibbs energy curve at $x^{\tan 2}, G(x^{\tan 2})$. If x_s is on the line between $x^{\tan 1}$ and $x^{\tan 2}$ proceed to **step 4.1.2** otherwise the source phase is stable.

Step 4.1.2 Find point $x^{\tan 1}$ located between $x^{\tan 2}$ and x^{it1} which together with $x^{\tan 2}$ defines a line passing through $(x^{\tan 1}, G(x^{\tan 1}))$ and $(x^{\tan 2}, G(x^{\tan 2}))$ which is tangent to the Gibbs energy curve at $(x^{\tan 1}, G(x^{\tan 1}))$.

Step 4.2 Find point $x^{\tan 2}$ located between x^s and x^{it2} which together with x^sdefines a line passing through $(x^{\tan 2}, G(x^{\tan 2}))$ and $(x^s, G(x^s))$ which is tangent to the Gibbs energy curve at $(x^{\tan 2}, G(x^{\tan 2}))$.

If $x^{\tan 2} = x^s$ then report the source phase to be stable and stop, otherwise go to **step 4.2.1.**

Step 4.2.1 Find point $x^{\tan 1}$ located between $x^{\tan 2}$ and x^{it1} which together with $x^{\tan 2}$ defines a line passing through $(x^{\tan 1}, G(x^{\tan 1}))$ and $(x^{\tan 2}, G(x^{\tan 2}))$ which is tangent to the Gibbs energy curve at $(x^{\tan 1}, G(x^{\tan 1}))$. If x_s is on the line between $x^{\tan 1}$ and $x^{\tan 2}$ the source phase is unstable, procede to **step 5** otherwise the source phase is stable.

Step 5 Minimize the Gibbs energy of the system using $x^{\tan 1}$ and $x^{\tan 2}$ as initial compositions.

As much of this method is based on heuristic principles rather than rigorous analysis, its rationale is best explained via an example.

Example 11 Walraven and van Rompay's Method *This case study from Walraven and van Rompay's paper, involves the ternary system: n-propanol, n-butanol and water at 355 K. The feed, 0.12 moles n-propanol, 0.08 moles n-butanol and 0.80 moles water is labelled "source" on the ternary diagram 8.3, and clearly lies within the two phase region bounded by the binodal curve. Values from pertinent steps are shown on the diagram, giving an*

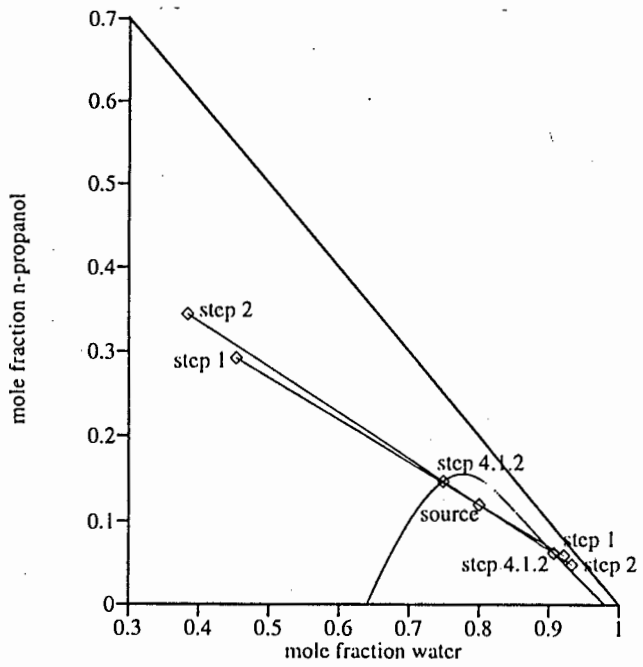

Figure 8.3: Typical progress of the Walraven and van Rompay method

overview of the Walraven and van Rompay solution strategy. **Step 1**, *which is essentially the Gautam and Seider initialization method, yields points* x^{t1} *and* x^{t2}, *which are far from the equilibrium compositions. Points* x^{it1} *and* x^{it2}, *calculated in* **step 2**, *lie farther from the equilibrium point on a line almost coincident with the equilibrium tie line. Points* $x^{\tan 1}$ *and* $x^{\tan 2}$ *generated by* **step 4.1.2** *are the initial guesses for the free energy minimization routine and are coincident with the equilibrium composition. The general strategy employed in the algorithm is therefore, to first find a pair of compositions that lie outside the unstable region on a line almost coincident with the equilibrium tie line, then to search for a more accurate equilibrium estimate along the line of compositions joining this pair. Figures 8.4 and 8.5 show how this is done. Let t parametrize the line connecting the pair of compositions found in* **step 2**. *If the points from* **step 2** *lie on a line nearly parallel to the equilibrium tie line, the Gibbs energy as a function of t will look similar to the nonconvex function in figure 8.4.*

This diagram shows step 4.1, wherein a composition $x^{\tan 1}$ is found which, together with x^s, defines a line which is tangent to $G(t)$ at $x^{\tan 1}$. As $x^{\tan 1}$ is different from x^s, Walraven

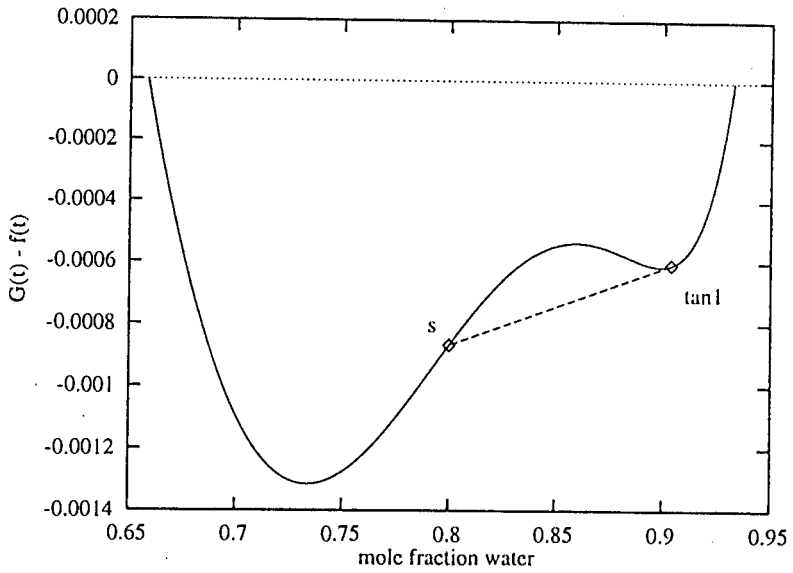

Figure 8.4: Walraven and van Rompay method: step 4.1

and van Rompay's algorithm dictates that the next step is **step 4.1.1**. *In this step, the composition, $x^{\tan 2}$ is found, which together with $x^{\tan 1}$ defines a line which is tangent to $G(t)$ at $x^{\tan 2}$. Figure 8.5 shows the result of* **step 4.1.2** *in which a new value for $x^{\tan 1}$ has been found, which together with $x^{\tan 2}$, defines a line tangent to $G(t)$ at $x^{\tan 1}$. As the line joining the pair of points from step 2 is coincident with the equilibrium tie line, the pair of points $x^{\tan 1}$ and $x^{\tan 2}$ is clearly an excellent approximation of the equilibrium state.*

8.2.4 Experimental Implementation of Walraven and van Rompay's Method

Walraven and van Rompay's method was implemented in FORTRAN according to the algorithm described in the previous section. The optimization package MINOS 5.4 was used to solve all minimization problems involved in the algorithm. The code is part of the Gautam and Seider method program, and is to be found in appendix C.

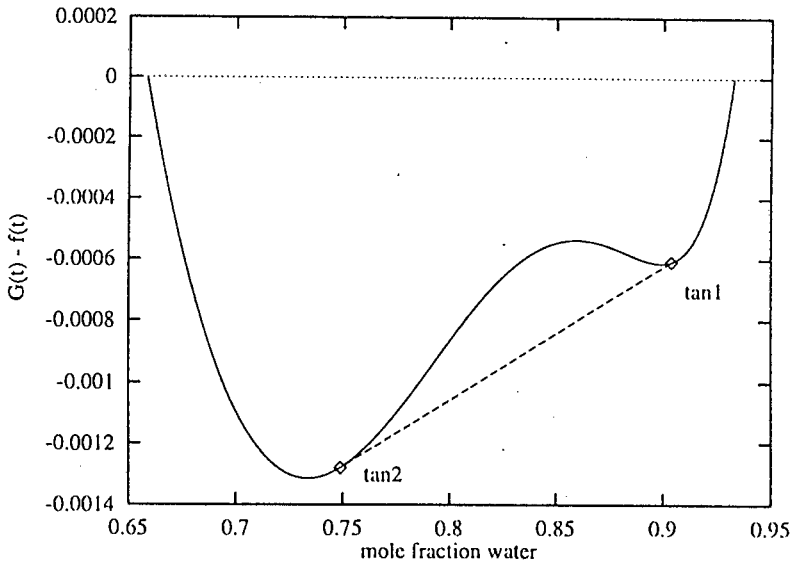

Figure 8.5: Walraven and van Rompay method: step 4.1.2

8.3 Evaluation of the Experimental Implementations

The reliability of the above methods was tested on a set of liquid-liquid equilibrium problems with the Gibbs energy function based on the NRTL activity coefficient model. Each of the phase splitting methods was initialized at the trivial single phase solution. For each set of NRTL data, a set of problems was solved. The problems corresponded to different source phase compositions, distributed evenly over the feasible composition domain. The results obtained for each method were then plotted on a ternary phase diagram to assess the validity of the solutions obtained and to identify feed composition regions that are problematic to the method employed. The systems used in this evaluation are:

Test system 1: n-propanol + n-butanol + water

This system, originally studied by Block and Hegner[4] in the context of three phase distillation columns, was used by Walraven and van Rompay[80] to demonstrate their phase splitting algorithm, and again by McDonald and Floudas[44] to demonstrate their global optimization procedure. The datafile for this problem is "prol-buol-wat1.dat" in appendix E. Both the Gautam - Seider and Walraven-van Rompay methods make use of unstable binary subsystems of the ternary system. In this example the only unstable binary pair is n-butanol + water. Results appear in ternary diagram 8.8.

Test system 2: toluene + water + aniline

Studied by Bender and Block [3], this system has also been solved by Castillo and Grossmann [12] using a variable metric nonlinear programming technique, Paules and Floudas [57], using a global optimum search technique, and McDonald and Floudas [44] using global optimization. Data for this example are from McDonald and Floudas [44] and appear in "tolu-wate-anil.dat". There are two unstable binary pairs for this system aniline + water, and toluene + water. This systems results appear in ternary diagram 8.9.

Test system 3: ethanol + ethyl acetate + water

Walraven and van Rompay [80] used this as a second example to demonstrate their phase splitting method. The data for this problem are from McDonald and Floudas [44] and appear in "etOH-etAc-wate.dat". The single unstable binary subset in this system is ethyl acetate + water. Ternary diagram 8.10 shows the results for this system.

Test system 4: water + ethanol + 2,2,4 - trimethyl pentane (TMP)

This example comes from Pesche and Sandler [59] who fitted the NRTL equation to experimentally obtained data. The only unstable binary pair in this system is water + 2,2,4-trimethyl pentane. Data appear in "wat-etOH-TMP.dat".

Results appear in figure 8.11.

Test system 5: water + tert-butyl-methyl ether (MTBE) + 2,2,4-trimethyl pentane

This example from the experimental work of Pesche and Sandler [59] has two unstable binary subsystems, water + tert-butyl-methyl ether, and water + 2,2,4-trimethyl pentane. Data appear in "wat-MTBE-TMP.dat" and the results are shown in figure 8.12.

Test system 6: water + tert-amyl alcohol (TAOH) + 2,2,4-trimethyl pentane

Another example from Pesche and Sandler [59], the two unstable binary subsystems are, tert-amyl alcohol + water, and 2,2,4-trimethyl pentane + water. Data appear in "wat-TAOH-TMP.dat" and the results are shown in figure 8.13.

Test system 7: water + tert-amyl-methyl ether (TAME) + 2,2,4-trimethyl pentane

This system is from Pesche and Sandler [59]. The two unstable binary pairs are tert-amyl-methyl ether + water and 2,2,4-trimethyl pentane + water. Data are shown in "wat-TAME-TMP.dat". Figure 8.14 shows the results for this system.

The feed compositions that were used for all systems besides system 1 are represented by the points shown in figure 8.6. The mole fraction compositions are distributed on a grid where vertically and horizontally adjacent compositions vary by 0.025 in species 1 and 2 respectively. Species 1 is always the first species listed in the description of the test systems while species 2 is the second. A higher grid resolution where adjacent compositions vary by 0.01 was used to test system 1. The test points for this system are displayed in figure 8.7.The results that are shown in figures 8.8 to 8.14 represent the feed compositions lying in the two phase region for which the respective methods incorrectly computed the equilibrium state as a single phase. In all figures, the symbols $\diamond$, $+$, and $\triangle$ respectively represent the feed compositions for which the Walraven and van Rompay, Gautam and Seider, and tangent plane method incorrectly computed single phase solutions. The tangent plane method did not incorrectly compute any single phase solutions for problems 2,4,5,6 and 7, hence the absence of the $\triangle$ symbol from figures 8.9, 8.11, 8.12, 8.13, and 8.14.

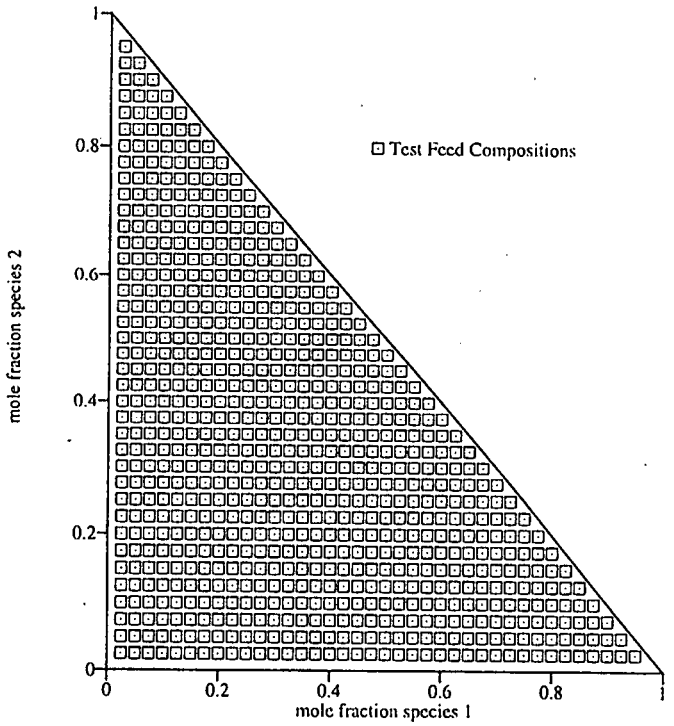

Figure 8.6: Feed compositions for reliability tests

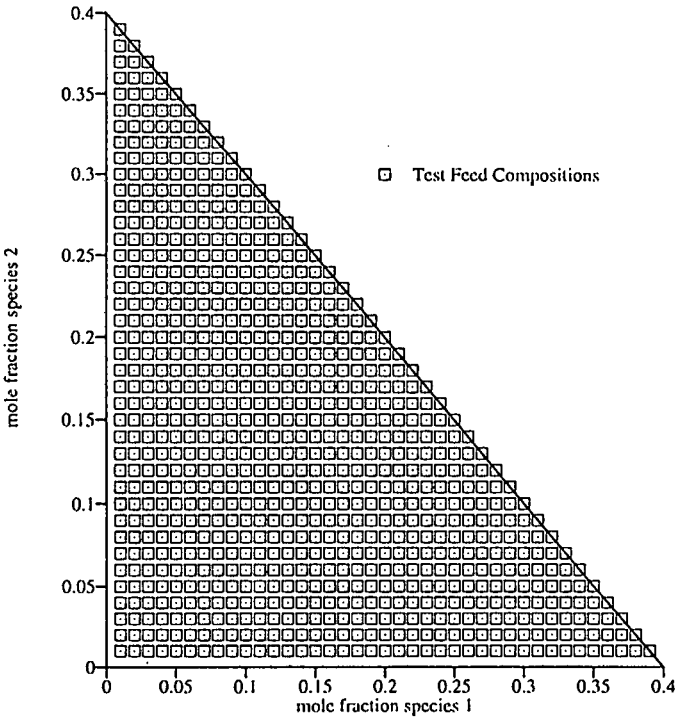

Figure 8.7: Feed compositions for reliability tests: system 1

8.3.1 Reliability Test Ternary Phase Diagrams

In figure 8.8, the binodal curve, indicates the boundary that separates the feed compositions that exist in two phases at equilibrium from those that exist in a single phase. The feed composition at which the methods failed is very close to the plait point, a region in which equilibrium computations are notoriously difficult.

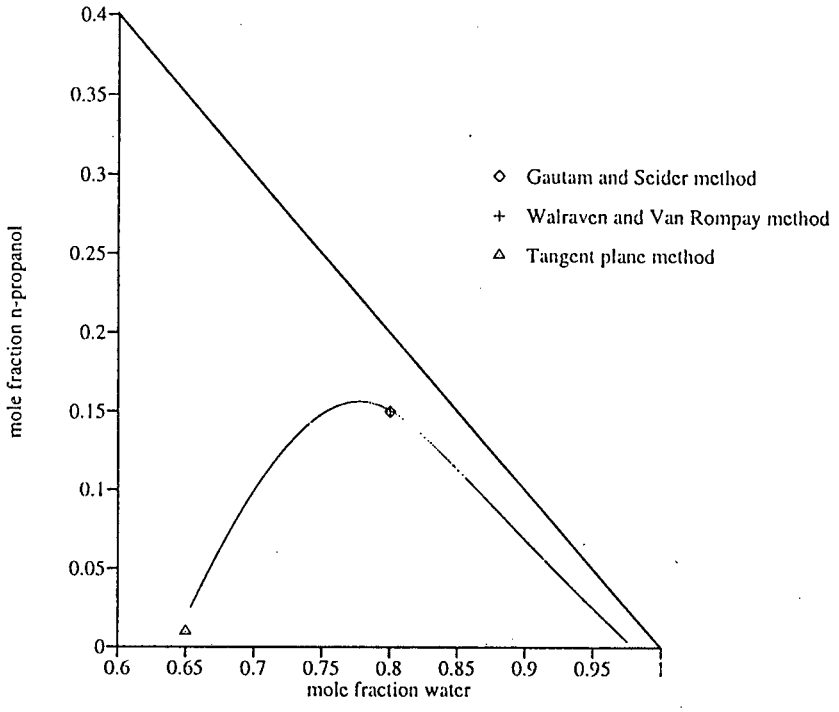

Figure 8.8: False single phase equilibria: n-propanol + n-butanol + water

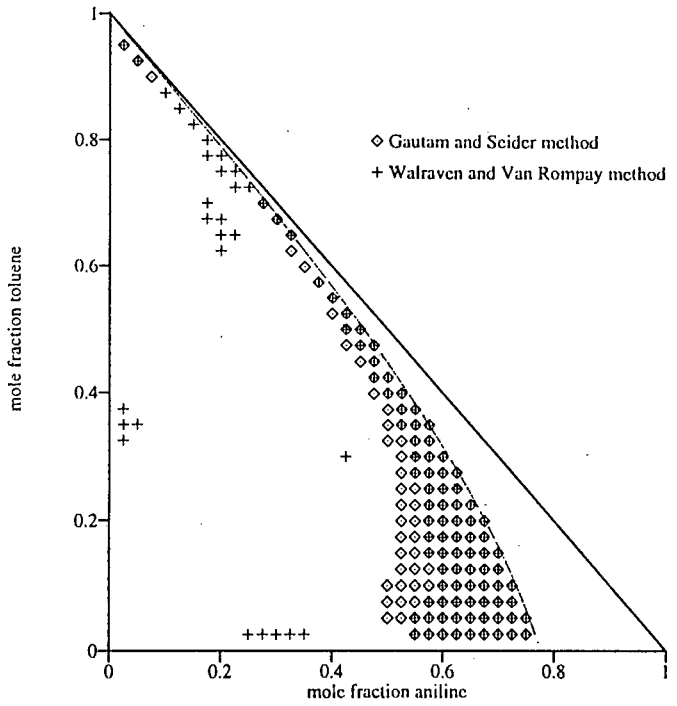

Figure 8.9: False single phase equilibria: toluene + water + aniline

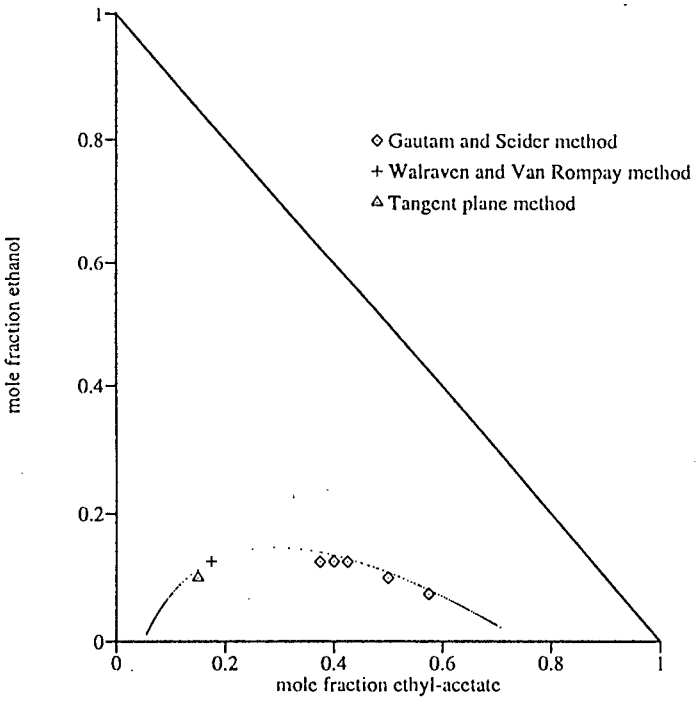

Figure 8.10: False single phase equilibria: ethanol + ethylacetate + water

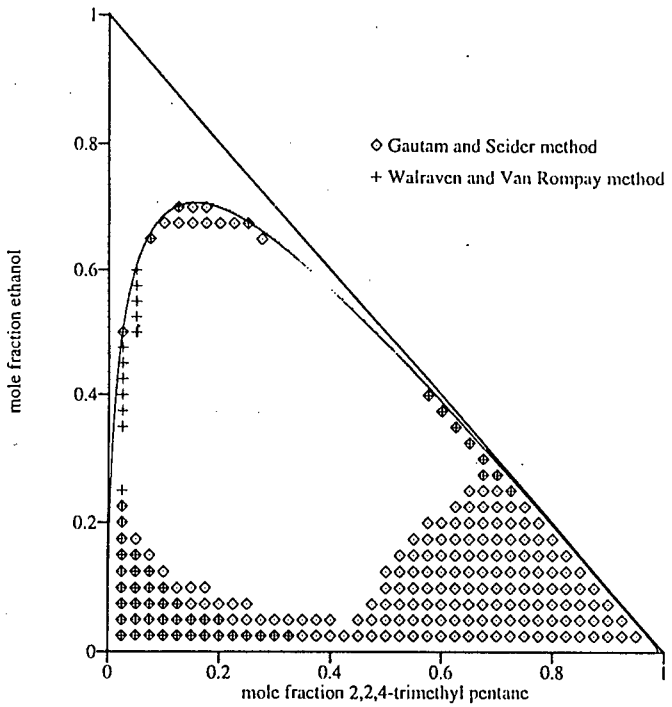

Figure 8.11: False single phase equilibria: water + ethanol + 2,2,4-trimethyl pentane

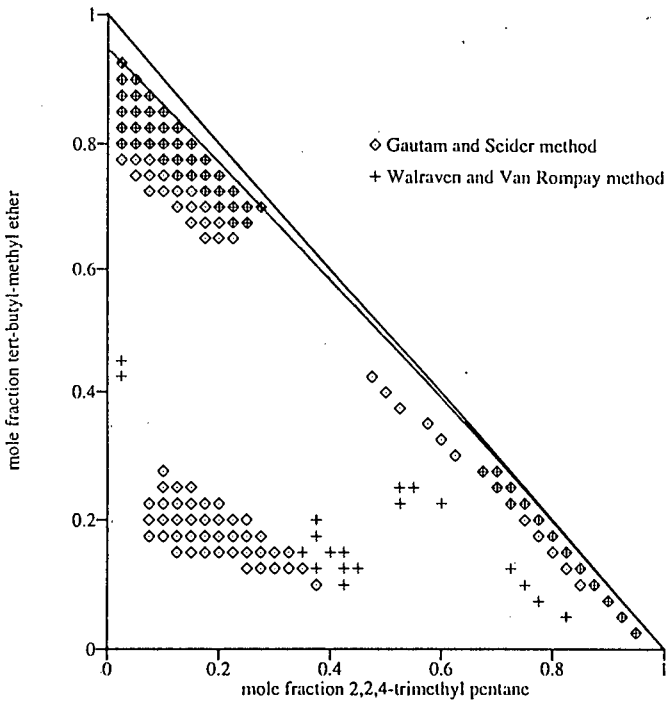

Figure 8.12: False single phase equilibria: water + tert-butyl-methyl ether + 2,2,4-trimethyl pentane

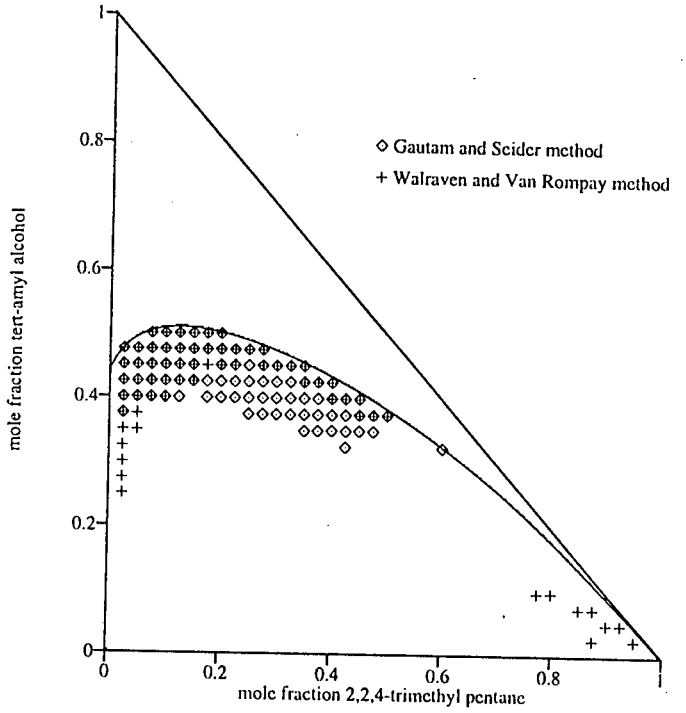

Figure 8.13: False single phase equilibria: water + tert-amyl alcohol + 2,2,4-trimethyl pentane

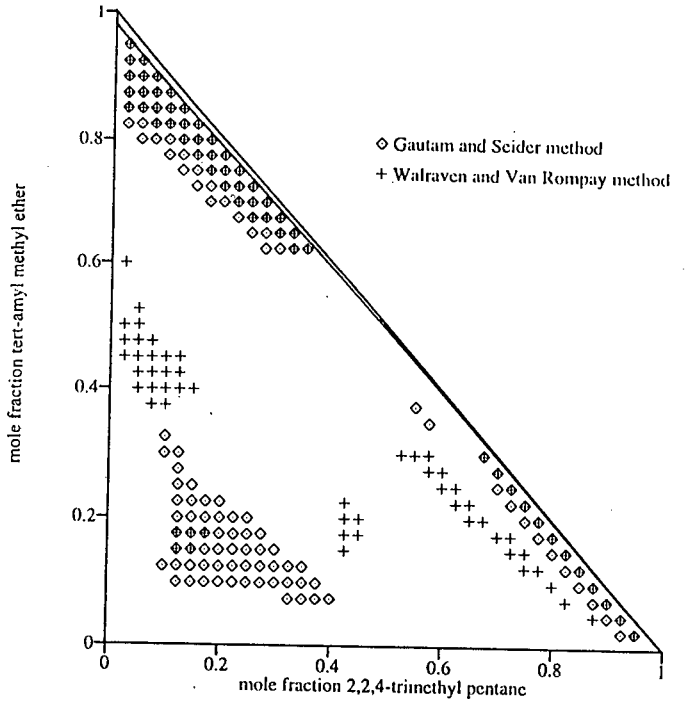

Figure 8.14: False single phase equilibria: water + tert-amyl-methyl ether + 2,2,4-trimethyl pentane

System	No. Feeds in Two Phase Region	Reliability		
		Gautam-Seider	Walraven-van Rompay	Tangent Plane
n-Propanol + n-Butanol + Water	43	0.98	0.98	0.98
Toluene + Water + Aniline	659	0.81	0.82	1.00
Ethanol + Ethylacetate + Water	93	0.95	0.99	0.99
Water + Ethanol + TMP	659	0.68	0.91	1.00
Water + MTBE + TMP	731	0.83	0.90	1.00
Water + TAOH + TMP	516	0.89	0.87	1.00
Water + TAME + TMP	741	0.81	0.86	1.00

Table 8.1: Reliability of Phase Splitting Algorithms

8.3.2 Discussion

The set of test problems indicate the general trends:

1. Phase splitting methods are most likely to fail when the feed is near the equilibrium composition of one of the equilibrium system's phases;
2. The performance of the Walraven and van Rompay method is an improvement on that of the Gautam and Seider method, but does sometimes result in failures where the Gautam and Seider method does not. This is due to the stability check in the Walraven and van Rompay method which may incorrectly indicate stability of the feed composition.
3. The Gautam-Seider and Walraven-van Rompay algorithms are more successful on systems where there is only one immiscible binary pair. This result appears to be a consequence of the heuristic which assumes phase splitting to occur primarily through the interaction of a pair of species.

An indication of the performance of a method is the ratio of two phase solutions found to the number of source phases in the unstable region on the ternary diagram solution:

$$\text{reliability} = \frac{\text{number of 2-phase solutions computed}}{\text{total number of feed compositions in 2-phase region}}.$$

Table 1 shows these statistics for all methods and all systems.

These results show the tangent plane method to be extremely reliable. Where the method does fail the results are likely to be of little practical consequence for the following reasons:

1. when the feed composition is near to the equilibrium composition of a phase the molar extent of the second phase is likely to be relatively small;

2. as the NRTL equation does not precisely match experimental observation; the equilibrium compositions determined by this equation differ from the physical equilibrium compositions. Incorrect results with regard to the phase splitting behaviour of feed compositions very near to the equilibrium curve are therefore as likely to result from the fit of the NRTL equation as from an error in the computation.

8.4 Comparison of CPU Times: Heuristic Versus Rigorous Approaches

In this section the CPU times required to solve a selection of ternary phase equilibrium problems are compared with the times required for the rigorous approaches to guarantee a global optimum. As the heuristic methods have not been optimized for speed these results do not represent their attainable efficiency. Therefore, the purpose of this section is not to make a definitive comparison of the methods, but to make some general observations on the order of magnitudes of the heuristic and rigorous methods' CPU times. Two systems were used for the CPU time studies, system 1 and system 7. Figure 8.15 shows the binodal curve for system 1 and the feed compositions that were used in the CPU time tests. Figure 8.16 shows the equilibrium composition curve and the CPU time test feed compositions for system 7. The values of the test compositions are shown in tables 8.2 and 8.3.

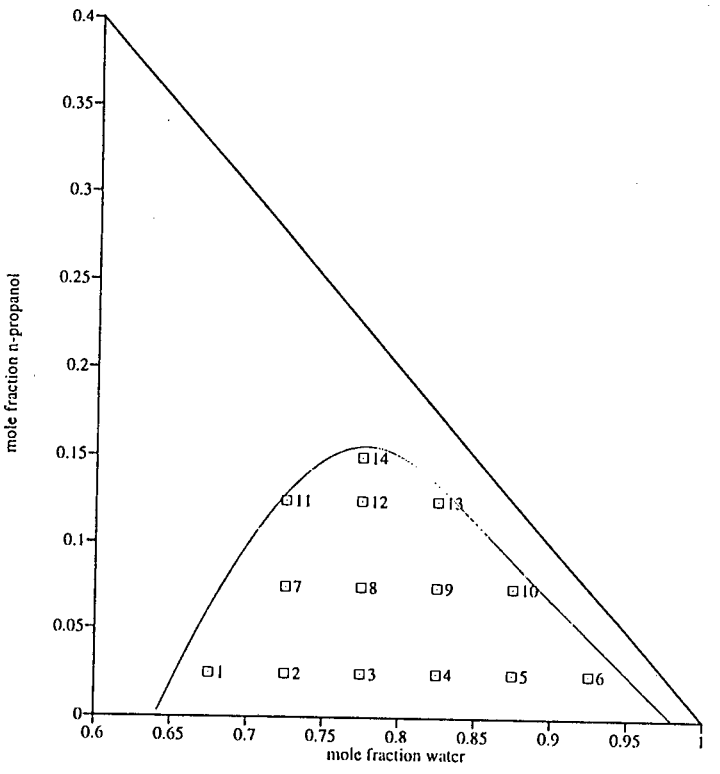

Figure 8.15: CPU time test feed compositions: system 1

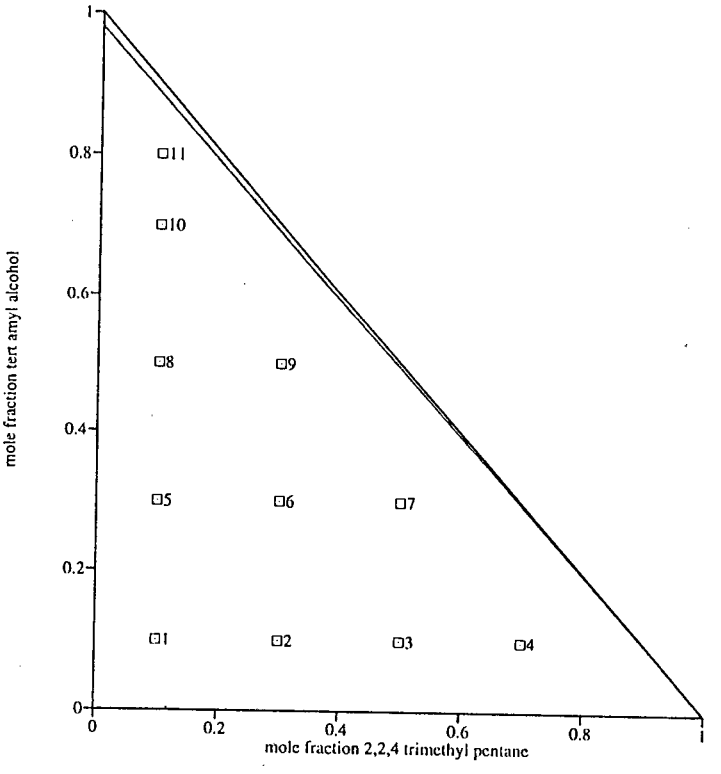

Figure 8.16: CPU time test feed compositions: system 7

feed identity	moles n-propanol	moles n-butanol	moles water
1	0.025	0.300	0.675
2	0.025	0.250	0.725
3	0.025	0.200	0.775
4	0.025	0.150	0.825
5	0.025	0.100	0.875
6	0.025	0.050	0.925
7	0.075	0.200	0.725
8	0.075	0.150	0.775
9	0.075	0.100	0.825
10	0.075	0.050	0.875
11	0.125	0.150	0.725
12	0.125	0.100	0.775
13	0.125	0.050	0.825
14	0.150	0.075	0.775

Table 8.2: CPU Test: System 1 Feed Compositions

The ϵ-optimality convergence tolerance used for the rigorous methods was set to 10^{-6}. The tests were done on a SUN SPARCstation 20.

The CPU times for the heuristic and rigorous methods on system 1 are shown in figure 8.17 and 8.18 respectively. CPU times are not reported for the rigorous methods for feeds 10 to 14 as they were unable to meet the convergence tolerance within an allocated 5000 major iterations. The following trends are clear:

1. Whereas the performance of the rigorous methods is highly dependent on the proximity of the feed composition to the binodal curve, the CPU times of the heuristic methods are not strongly dependent on this factor.

2. Of the heuristic methods, the tangent plane method solves all test problems in the shortest CPU time, followed by Gautam and Seider's method. This is an inversion of the results reported in the literature: Walraven and van Rompay reported superior performance for their method over the Gautam and Seider approach [80], and Swank and Mullins found their implementation of the Gautam and Seider method to be generally faster than their implementation of the tangent plane method [76]. Implementation details therefore play an important part in the efficiency of the heuristic methods, and since this study makes no attempt to optimize their computational speed, no conclusions can be drawn with regard to the relative efficiency of the heuristic methods.

feed identity	moles water	moles TAME	moles TMP
1	0.8	0.1	0.1
2	0.6	0.1	0.3
3	0.4	0.1	0.5
4	0.2	0.1	0.7
5	0.6	0.3	0.1
6	0.4	0.3	0.3
7	0.2	0.3	0.5
8	0.4	0.5	0.1
9	0.2	0.5	0.3
10	0.2	0.7	0.1
11	0.1	0.8	0.1

Table 8.3: CPU Test: System 7 Feed Compositions

3. Of the rigorous methods, the method which makes use of the convexity test is consistently faster. For feed compositions near the binodal curve the benefit of the convexity test is smaller than towards the centre of the two-phase region.

4. The CPU times taken for the rigorous solution methods to converge are hundreds of times larger than the heuristic methods' CPU times.

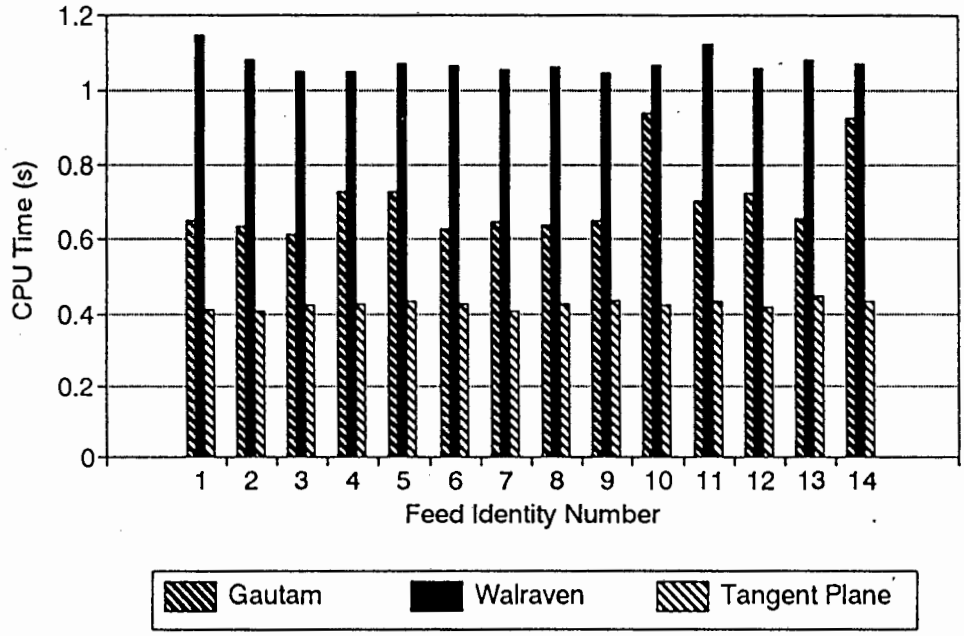

Figure 8.17: Heuristic methods' CPU times: system 1

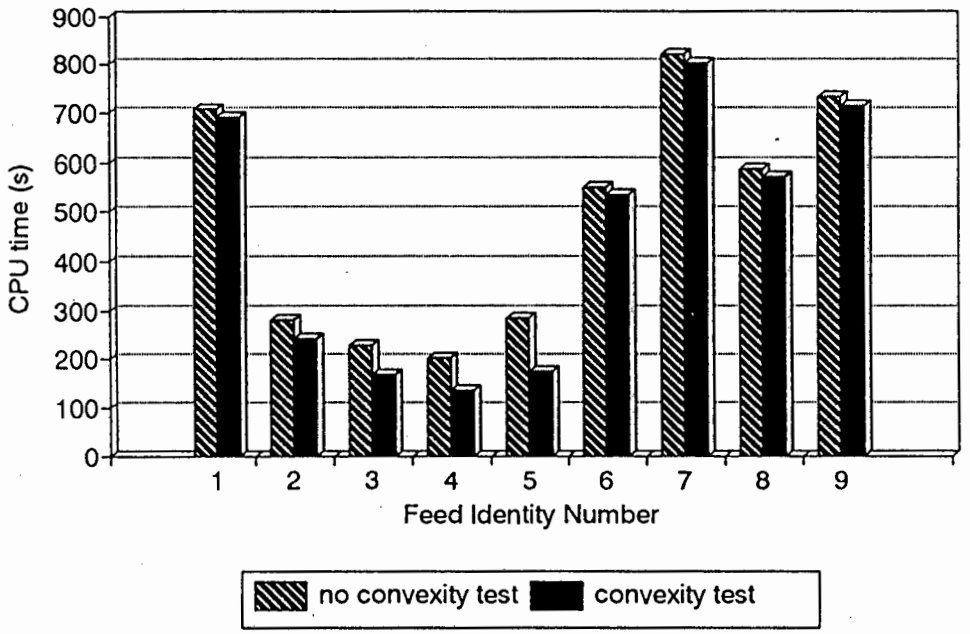

Figure 8.18: Rigorous methods' CPU times: system 1

Figures 8.19 and 8.20 show the CPU times for the heuristic and rigorous methods applied to system 7. Not reported, are heuristic method CPU times for compositions where an incorrect solution was obtained. The results pertaining to the heuristic methods are similar to those reported for system 1. For system 7 the rigorous methods converged far more rapidly than for system 1, and their CPU times compare more favorably with those of the heuristic approaches.

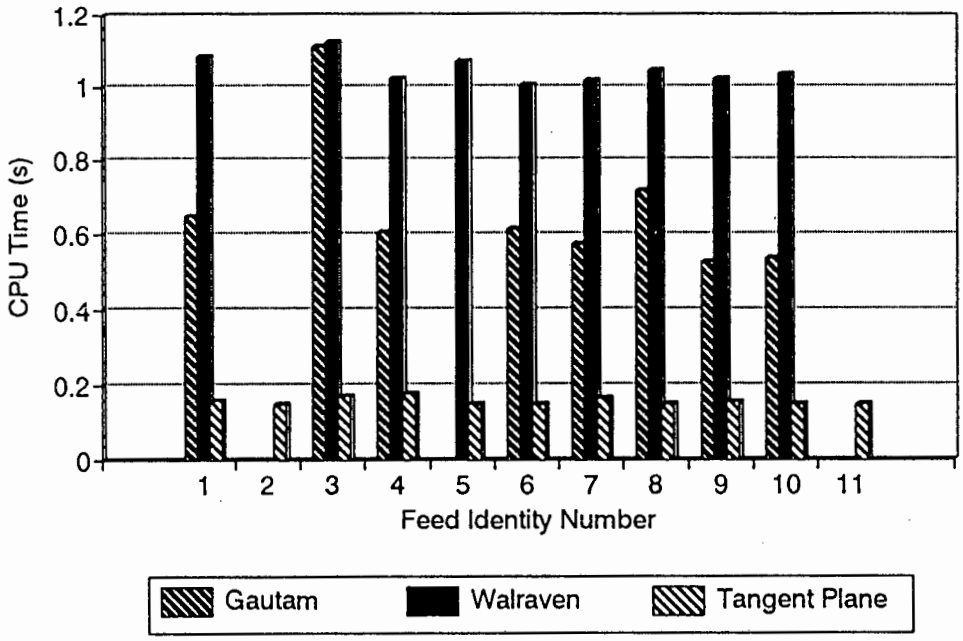

Figure 8.19: Heuristic methods' CPU times: system 7

The results from these tests clearly indicate that the rigorous approaches are far more computationally intensive than the heuristic approaches. However, it is important to note that the objectives of the rigorous and heuristic approaches differ enormously: the heuristic methods aim to compute the equilibrium composition while the rigorous global optimization methods aim to compute *and verify* the equilibrium composition.

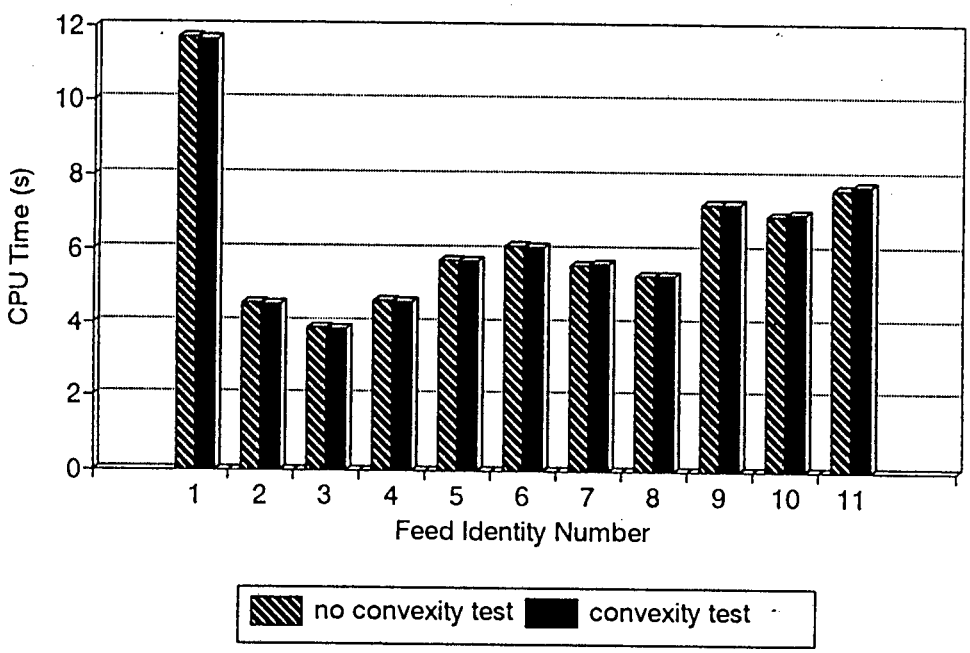

Figure 8.20: Rigorous methods' CPU times: system 7

Chapter 9

Conclusion

The problem of minimizing the Gibbs energy of a system can be formulated as a nonlinear program subject to linear equality and simple bound inequality constraints. Linearly constrained nonlinear programming has been studied extensively, in fact many of the important insights into the Gibbs energy minimization problem have originated from this more general field. A number of features which may influence the behaviour of an algorithm are inherent in the structure of the Gibbs energy minimization problem. Through the recognition and, where possible, the exploitation of these features general nonlinear programming algorithms may be tailored to suit the more specific problem. The features that characterize the Gibbs energy minimization problem are:

1. the Gibbs energy function is a function of its gradients;
2. the objective function is undefined when the non-negativity constraints are violated;
3. the gradients of the objective function become unbounded as species vanish from the system;
4. the Hessian matrix has a special structure due to the Gibbs-Duhem equation;
5. the projected Hessian of the objective function may become extremely ill-conditioned;
6. some thermodynamic models generate convex Gibbs energy functions, while others generate non-convex functions;
7. the number of species and phases present in the equilibrium system are not known *a priori*;

8. the Gibbs energy function may closely resemble an affine function for systems operating at very high temperatures, due to the dominance of enthalpic effects over entropic effects at these temperatures;

9. by today's nonlinear programming standards typical Gibbs energy minimization problems involve a small number of variables;

10. the Gibbs energy function is partially separable for multiphase systems.

Of these properties the ones that appear to have the most significance are 5 , 6 and 7.

The ill-conditioning of the problem severely diminishes the convergence rates of algorithms such as the steepest descent method which makes use of the gradient, but not the Hessian. In Newton's method where the Hessian is used and the second derivatives are calculated analytically, ill-conditioning is far less problematic. Newton's method can be applied to Gibbs energy minimization within the framework of reduced space, Lagrangian or barrier function methods.

A priori lack of knowledge of the number of species and phases can be dealt with by posing the equilibrium problem as a superstructure which encompasses the maximum possible number of species which can exist in each phase and the maximum number of possible phases which can exist in the system. This problem can be solved through the use of inequality constraints and an active set strategy in reduced space or Lagrangian methods. Alternatively, barrier functions may be used to incorporate the inequality constraints into the objective function. A third possibility, proposed by Paules and Floudas [57], is to pose the Gibbs energy minimization problem as a mixed integer nonlinear program wherein discrete (0,1) search variables are used to represent the presence or absence of a species or phase. This method was implemented using the Generalized Benders Decomposition [25]. Since the inclusion of discrete variables makes the problem as a whole considerably more difficult to solve, this method was found to be inappropriate for the Gibbs energy minimization problem. When the Gibbs energy function is not convex, the above approaches may fail to determine the correct number of phases owing to the presence of multiple local minima on the Gibbs energy surface.

The issue of convexity is by far the most important factor influencing the solution of the Gibbs energy minimization problem. When the Gibbs energy function is convex, the equilibrium problem is relatively easy to solve: in these problems a local minimum is also the global minimum, and convexity can be directly exploited in the design of efficient algorithms. Efficient interior point methods for convex minimization problems have been applied to a transformation of the geometric programming problem which is equivalent to the ideal gas

Gibbs energy minimization problem.

However, when the Gibbs energy function is nonconvex a local minimum is not necessarily the global minimum, hence global optimization strategies need to be employed to solve such problems. The focus of much of this thesis has been on methods of solving nonconvex Gibbs energy minimization problems where the Gibbs energy is modelled by the NRTL equation. Methods of globally minimizing the Gibbs energy can be classified into two categories: rigorous and heuristic.

The rigorous decomposition based method of McDonald and Floudas [44] was implemented and assessed. The method is theoretically capable of guaranteeing the global Gibbs energy *minimum* to within a predefined accuracy ϵ. In practice the method is robust and reliable, but expensive in CPU time, and there is no *a priori* way of setting the convergence tolerance ϵ to ensure that the global *minimizers* of the Gibbs energy function will be found. A modification of the global optimization algorithm was introduced in this thesis. It makes use of regional convexity on the Gibbs energy surface to improve the method's convergence rate. This method is based on interval analysis ideas and makes use of a positive definiteness test for the projected Hessian. The effectiveness of the modification in reducing CPU time is dependent on the specific problem and the convergence tolerance.

Unlike the rigorous methods, the heuristic algorithms cannot guarantee the global optimality of the solutions they find. Nevertheless, they are simpler to implement and run in a fraction of the time required by the rigorous methods. Their utility in solving equilibrium problems is dependent on their ability to find the global optimum. Results of tests on the solution of ternary phase equilibrium problems indicate that the tangent plane method [1, 47] is extremely reliable in practice, while the Gautam and Seider [22],and Walraven and van Rompay [80] methods are less dependable. The heuristic methods have the most difficulty finding the global minimum when the feed composition is close to the binodal curve. A comparison of the CPU times of the heuristic methods versus those of the rigorous methods shows that the CPU times for the rigorous methods are ten to a thousand times greater than the CPU times for the heuristic approaches.

The reliability of the tangent plane method in practice, and the order of magnitude difference between the rigorous and heuristic methods' CPU times indicate that the tangent plane approach is most appropriate for the solution of practical phase equilibrium problems. However, the empirical results on the reliability of the heuristic methods provide no guarantee on the reliability of these methods when applied to other systems. Therefore the global optimization of nonconvex Gibbs energy functions remains a challenging research subject.

References

[1] L. E. Baker, A. C. Pierce, and K. D. Luks. Gibbs energy analysis of phase equilibrium. *Society of Petroleum Engineers Journal*, 22:731–742, 1982.

[2] R. E. Balzhiser, M. R. Samuels, and J. D. Eliassen. *Chemical Engineering Thermodynamics.* Prentice Hall, 1972.

[3] E. Bender and U. Block. Thermodynamische berechnung der flussig-flussig-extraktion. *Verfahrenstechnik*, 9(3):106, 1975.

[4] U. Block and B. Hegner. Development and application of a simulation model for three phase distillation. *American Institute of Chemical Engineers Journal*, 22(582-589):3, 1976.

[5] J. F. Boston and H. I. Britt. A radically different formulation and solution of the single-stage flash problem. *Computers and Chemical Engineering*, 2:109–122, 1978.

[6] F. P. Boynton. Chemical equilibrium in multicomponent polyphase systems. *The Journal of Chemical Physics*, 32:1880, 1960.

[7] S. R. Brinkley. Erratum: 'note on the conditions of equilibrium for systems of many constituents'. *The Journal of Chemical Physics*, 14(11):686, 1946.

[8] S. R. Brinkley. Note on the conditions of equilibrium for systems of many constituents. *The Journal of Chemical Physics*, 14(9):563–564, 1946.

[9] S. R. Brinkley. Calculation of the equilibrium composition of systems of many constituents. *The Journal of Chemical Physics*, 15(2):107–110, 1947.

[10] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A Users Guide.* Palo Alto, C.A., 1988.

[11] M. Castier, P. Rasmussen, and A. Fredenslund. Calculation of simultaneous phase and chemical equilibria in nonideal systems. *Chemical Engineering Science*, 14(2):237, 1989.

[12] J. Castillo and I. E. Grossmann. Computation of chemical and phase equilibria. *Computers and Chemical Engineering*, 5:99–108, 1981.

[13] R. W. Cottle. Manifestations of the Schur complement. *Linear Algebra and its Applications*, 8:189–211, 1974.

[14] D. R. Cruise. Notes on the rapid computation of chemical equilibria. *The Journal of Physical Chemistry*, 68(12):3797–3802, 1964.

[15] D. den Hertog, F. Jarre, C. Roos, and T. Terlaky. A unifying investigation of interior point methods for convex programming. Technical Report 92-89, Faculty of Mathematics and Informatics, TU Delft, 1992.

[16] D. den Hertog, C. Roos, and T. Terlaky. On the classical logarithmic barrier function method for a class of convex programming problems. Technical Report 90-28, Faculty of Mathematics and Informatics, TU Delft, 1990.

[17] K. G. Denbigh. *The Principles of Chemical Equilibrium: with Applications in Chemistry and Chemical Engineering.* Cambridge University Press, 3rd edition, 1971.

[18] I. Duff. The solution of augmented systems. Technical Report RAL-93-084, Rutherford Appleton Laboratory, Chilton, England, 1993.

[19] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* Wiley, 1968.

[20] C. A. Floudas and V. Visweswaran. Primal-relaxed dual global optimization approach. *Journal of Optimization Theory and Applications*, 78(2):187–225, 1993.

[21] G. B. Garcia and W. I. Zangwill. *Pathways to Solutions, Fixed Points, and Equilibria.* Prentice-Hall, 1981.

[22] R. Gautam and W. D. Seider. Computation of phase and chemical equilibrium: Part 1. local and constrained minima in Gibbs free energy. *American Institute of Chemical Engineers Journal*, 25(6):991–999, 1979.

[23] R. Gautam and W. D. Seider. Computation of phase and chemical equilibrium: Part 2. phase splitting. *American Institute of Chemical Engineers Journal*, 25(6):999–1006, 1979.

[24] A. M. Geoffrion. Duality in nonlinear programming: A simplified applications-oriented approach. *SIAM Review*, 13(1):1–36, 1971.

[25] A. M. Geoffrion. Generalized Benders Decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

[26] P. Gill and W. Murray. Newton-type methods for unconstrained and linearly constrained optimization. *Mathematical Programming*, 7:311–350, 1974.

[27] P. E. Gill and W. Murray. *Newton-type Methods for Linearly Constrained Optimization*, chapter 2, pages 29–66. Academic Press, 1974.

[28] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright. On projected newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Mathematical programming*, 36:183–202, 1986.

[29] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press Inc., London, 1981.

[30] S. Gordon and B. J. McBride. NASA special publication-273. Technical report, NASA, 1971.

[31] W. W. Hager. *Applied Numerical Linear Algebra.* Prentice-Hall, 1988.

[32] C. E. Harvie, J. P. Greenberg, and J. H. Weare. A chemical equilibrium algorithm for highly non-ideal multiphase systems: Free energy minimization. *Geochimica et Cosmochimica Acta*, 51:1045–1057, 1987.

[33] O. Heuze, H. Presles, and P. Bauer. Computation of chemical equilibria. *The Journal of Chemical Physics*, 83(9):4734–4737, 1985.

[34] V. N. Huff, S. Gordon, and V. E. Morrell. Report 1037. Technical report, NACA, 1951.

[35] B. Jansen, C. Roos, and T. Terlaky. A short survey on ten years interior point methods. Technical Report 95-45, Faculty of Mathematics and Informatics, TU Delft, 1995.

[36] F. Jarre. Interior point methods for convex programming. *Applied Mathematics and Optimization*, 26:287–311, 1992.

[37] N. K. Karmarkar. A new polynomial–time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[38] K. O. Kortanek, X. Xu, and Y. Ye. An infeasible interior-point algorithm for solving primal and dual geometric programs. ftp://dollar.biz.uiowa.edu/pub/yyye/kxy.ps, 1995.

[39] G. Lantagne, B. Marcos, and B. Cayrol. Computation of complex equilibria by nonlinear optimization. *Computers and Chemical Engineering*, 12(6):589–599, 1988.

[40] L. Lenard. A computational study of active set strategies in nonlinear programming with linear constraints. *Mathematical Programming*, 16:81–97, 1979.

[41] A. Lucia and S. Macchietto. New approach to approximation of quantities involving physical properties derivatives in equation-orientated process design. *American Institute of Chemical Engineers Journal*, 29(5):705–712, 1983.

[42] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, 2nd edition, 1984.

[43] Y. H. Ma and C. W. Shipman. Computation of complex equilibrium. *American Institute of Chemical Engineers Journal*, 18(2):299–304, 1972.

[44] C. M. McDonald and C. A. Floudas. Decomposition based and branch and bound global optimization approaches for the phase equilibrium problem. *Journal of Global Optimization*, 5:205–251, 1994.

[45] C. M. McDonald and C. A. Floudas. Global optimization for the phase and chemical equilibrium problem: Application to the NRTL equation. *Computers and Chemical Engineering*, 19(11):1111–1141, 1995.

[46] C. M. McDonald and C. A. Floudas. Global optimization for the phase stability problem. *American Institute of Chemical Engineers Journal*, 41(7):1798–1814, 1995.

[47] M. L. Michelsen. The isothermal flash problem. part 1. stability. *Fluid Phase Equilibria*, 9:1–20, 1982.

[48] M. L. Michelsen. The isothermal flash problem. part 2. phase split calculation. *Fluid Phase Equilibria*, 9:21–40, 1982.

[49] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms: Part 2: Convex quadratic programming. *Mathematical Programming*, 44:43–66, 1989.

[50] B. A. Murtagh and M. A. Saunders. Large-scale linearly constrained optimization. *Mathematical Programming*, 14:41–72, 1978.

[51] L. M. Naphtali. Complex chemical equilibria by minimizing free energy. *The Journal of Chemical Physics*, 31:263–264, 1959.

[52] L. M. Naphtali. Calculate complex chemical equilibria. *Industrial and Engineering Chemistry*, 53(5):387–388, 1961.

[53] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*, volume 13 of *SIAM Studies in Applied Mathematics.* Society for Industrial and Applied Mathematics, 1994.

[54] A. Neumaier. Second-order sufficient optimality conditions for local and global nonlinear programming. *to appear in the Journal of Global Optimization*, 1991.

[55] R. C. Oliver, S. E. Stephanou, and R. W. Baier. Calculating free energy minimization. *Chemical Engineering*, pages 121–128, February 1962.

[56] U. Passy and D. J. Wilde. A geometric programming algorithm for solving chemical equilibrium problems. *SIAM Journal of Applied Mathematics*, 16(2), 1968.

[57] G. E. Paules and C. A. Floudas. APROS: Algorithmic development methodology for discrete-continuous optimization problems. *Operations Research*, 37(6):902–915, 1989.

[57a] G. E. Paules, and C. A. Floudas. A New Optimization Approach to Phase and Chemical Equilibrium Problems. Paper presented at the annual AIChE meeting, San Francisco, CA, November, 1989.

[58] A. L. Peressini, F. E. Sullivan, and J. J. Uhl. *The Mathematics of Nonlinear Programming*. Undergraduate Texts in Mathematics. Springer-Verlag, 1988.

[59] N. Peschke and S. I. Sandler. Liquid-liquid equilibria of fuel oxygenate + water + hydrocarbon mixtures. 1. *Journal of Chemical and Engineering Data*, 40:315–320, 1995.

[60] E. L. Peterson. *Geometric Programming*, chapter 2, pages 31–94. Plenum Press, New York and London, 1974.

[61] I. Prigogine and R. Defay. *Treatise on Thermodynamics Vol 1: Chemical Thermodynamics*. Longmans, 1954. Translated by D. H. Everette.

[62] B. N. Raju and C. S. Krishnaswami. Free energy minimization method for calculating thermodynamic equilibrium composition of chemical systems. *Indian Journal of Technology*, 4(4):99–100, 1966.

[63] M. M. Ruda and D. W. Thomson. Comparison of primal and dual solution methods for the chemical equilibrium problem using GRG and sensitivity analysis. *The Canadian Journal of Chemical Engineering*, 63:113–121, 1985.

[64] D. E. Salane. Symmetric minimum-norm updates for use in Gibbs free energy calculations. *Mathematical Programming*, 36:145–156, 1986.

[65] R. V. Sanderson and H. H. Y. Chien. Simultaneous chemical and phase equilibrium calculation. *Industrial and Engineering Chemistry Process Design and Development*, 12(1):81–85, 1973.

[66] S. I. Sandler. *Chemical and Engineering Thermodynamics*. John Wiley and Sons, 1989.

[67] R. W. H. Sargent and B. A. Murtagh. Projection methods for non-linear programming. *Mathematical Programming*, 4:245–268, 1973.

[68] P. V. L. N. Sarma, X. M. Martens, G. V. Reklaitis, and M. J. Rijckaert. *Geometric Programming*, chapter 12, pages 263–281. Plenum Press, New York and London, 1974.

[69] J. V. Smith, R. W. Missen, and W. R. Smith. General optimality criteria for multiphase multireaction chemical equilibrium. *American Institute of Chemical Engineers Journal*, 39(4):707–710, 1993.

[70] W. R. Smith. The computation of chemical equilibria in complex systems. *Industrial and Engineering Chemistry Fundamentals*, 19:1–10, 1980.

[71] W. R. Smith and R. W. Missen. On the two derivative method for optimisation. *The Canadian Journal of Chemical Engineering*, 45:346–348, 1967.

[72] W. R. Smith and R. W. Missen. Calculating complex chemical equilibria by an improved reaction adustment method. *The Canadian Journal of Chemical Engineering*, 46:269–272, 1968.

[73] W. R. Smith and R. W. Missen. *Chemical Reaction Equilibrium Analysis: Theory and Algorithms*. Wiley, 1982.

[74] E. E. Stone. Complex chemical equilibriums; application of Newton-Raphson method to solve nonlinear equations. *Journal of Chemistry Education*, 43(5):241–244, 1966.

[75] S. H. Storey and F. V. Zeggeren. Computation of chemical equilibrium composition 2. *The Canadian Journal of Chemical Engineering*, 48:591–593, October 1970.

[76] J. S. Swank and J. C. Mullins. Evaluation of methods for calculating liquid-liquid phase-splitting. *Fluid Phase Equilibria*, 30:101–110, 1986.

[77] J. A. Trangenstein. Customized minimization techniques for phase equilibrium computations in reservoir simulation. *Chemical Engineering Science*, 42(12):2847–2863, 1987.

[78] D. S. Villars. A method of successive approximations for computing combustion equilibria on a high speed digital computer. *The Journal of Physical Chemistry*, 63:521–525, 1959.

[79] J. Viswanathan and I. E. Grossmann. A combined penalty function and outer-approximation method for MINLP optimization. *Computers and Chemical Engineering*, 14(7):769–782, 1990.

[80] F. F. Y. Walraven and P. V. van Rompay. An improved phase splitting algorithm. *Computers and Chemical Engineering*, 12(8):777–782, 1988.

[81] W. B. White, S. M. Johnson, and G. B. Dantzig. Chemical equilibrium in complex mixtures. *The Journal of Chemical Physics*, 28(5):751–755, 1958.

[82] M. H. Wright. Interior methods for constrained optimization. *Acta Numerica*, pages 341–407, 1992. ftp://netlib.att.com/netlib/att/cs/doc/91/4-10.ps.Z.

[83] M. H. Wright. Some linear algebra issues in large scale optimization. http://netlib.att.com/netlib/att/cs/doc/93/4-01.ps.Z, 1993.

[84] Y. Ye. Interior point methods. ftp://dollar.biz.uiowa.edu/pub/yyye/intbook.ps, 1995.

[85] Y. Ye and E. Tse. An extension of Karmarkar's projective algorithm for convex quadratic programming. *Mathematical Programming*, 44:157–179, 1989.

[86] F. J. Zeleznik and S. Gordan. Calculation of complex chemical equilibria. *Industrial and Engineering Chemistry*, 60(6):27–57, 1968.

Appendix A

Mixed Integer Nonlinear Programming GAMS Program

```
$TITLE: Generalised Benders Decomposion for chemical equilibrium test
$OFFUPPER
$OFFSYMXREF  OFFSYMLIST
$INLINECOM {{ }}
$ONNESTCOM
OPTION LIMROW = 0
OPTION LIMCOL = 0
OPTION SYSOUT = OFF
OPTION SOLPRINT= ON
OPTION OPTCR = 0.0
       OPTCA = 0.0
       DECIMALS =  8

*-----------------------------------------------------------------------
* Define index sets
SET  I  Index set for atomic elements  /C,Fe,H,O/;
SET  S  Index set for the species /Fe,FeO,C,CO,CO2,H2,O2,H2O/;
SET  K  Index set for the phases  /solidFe,solidFeO,solidC,gas/;
SET  SK Index set to associate species with phases
       /Fe .solidFe
        FeO.solidFeO
        C  .solidC
        CO .gas
        CO2.gas
        H2 .gas
        O2 .gas
        H2O.gas/;

*-----------------------------------------------------------------------
*Define sets for control of the GBD algorithm
SETS
JM              Static iterations/J1*J100/
J(JM)           Dynamic iterations
JITER(JM)       Dynamic counter
JLOOPCTL(JM)    Loop control index;

ALIAS(JLOOP,JM);

*-----------------------------------------------------------------------
* Define and Input the Problem Data

TABLE A(I,S) Moles of Atoms of Elements I per mole of species S
            Fe   FeO   C    CO   CO2  H2   O2   H2O
       C              1     1     1
       Fe    1    1
       H                                2          2
       O           1         1     2         2     1   ;

TABLE W(S,K) Constants in the Chemical Potential Equations

              solidFe  solidFeO  solidC   gas
       Fe        0.00
       FeO                -8.53
       C                            0.00
       CO                                    -11.30
       CO2                                   -19.40
       H2                                      0.00
       O2                                      0.00
       H2O                                    -8.39   ;
```



```
PARAMETER FEED(I) Total Moles of Atoms of Elements in the system/
        C  2.75
        H  1.5
        Fe 1.00
        O  2.75/;


SCALAR  P Pressure in atm /1/;

PARAMETER MOLEFRAC(S,K) Mole fractions of species at equilibrium;

{{GBD parameters used to "fix" variables}}

PARAMETER Yparam(S,K)        current binary proposal
          YPparam(K)    ;
Yparam(S,K)$SK(S,K) = 1;
YPparam(K) = 1;
{{Yparam("CO","gas")=0;}}
YPparam("gas")=1;
PARAMETER Ysave(S,K,JM)      storage of binary variable solutions
          YPsave(K,JM)
          NTOTparam(K,JM)    storage of continuous variable solutions
          Nparam(S,K,JM)     storage of continuous variable solutions
          Lmult(S,K,JM)      Lagrange multipliers for NLP values
          Lmult2(S,K,JM)   {{new}}
          Lmult3(K,JM)
          ZoaUp              upper bound on MILP master
          ZoaLo              lower bound on MILP master
          Yopt(S,K)          storage of optimal solution binary combination
          YPopt(K)
          Jopt               optimal solution iteration
          Infes(JM)          storage of infeasible subproblem flags
          NLPrec(JM)         helps to interpret what is going on
          MIPrec(JM)         helps to interpret what is going on;
Zoaup = inf;

SCALARS  NOTFEAS /1/;
SCALAR   LOWMOL /1E-4/;
SCALAR   SCALEFAC/1/;
FEED(I)=SCALEFAC*FEED(I);
*-----------------------------------------------------------------------
* Define optimization variables

POSITIVE VARIABLES
    N(S,K)      moles of species s in phase k
    NINF(S,K)   values of moles from infeasible minimisations
    NTOT(K)     moles in phase k
    NTOTINF(K)  moles in phase k for infeasible problem ;

BINARY VARIABLES
    Y(S,K)      existance of species s in phase k
    YP(K)       existance of phase k ;

VARIABLES
    ZOAU        NLP objective value
    ZINF        objective value of infeasible minimisation
    ALPHA       minimised infeasibilities
    ZOAL        MILP objective value;

{{Bounds on variables}}
```

```
 N.LO(S,K)       = -1;
 NINF.LO(S,K)    = -1;
 N.UP(S,K)       = 100;
 NINF.UP(S,K)    = 100;
 N.L(S,K)$(SK(S,K)) = LOWMOL;

*---------------------------------------------------------------------------
*Declare equations
EQUATIONS
   NLPOBJ                NLP subproblem objective
   TMOLS(K)
   MASSBAL(I)            Element mass balance
   ZOSPF(S,K)            Controls species existance
   ZOSPF2(S,K)
                      {{Contains both binary and continuous variables}}
   ZOPHLO(K)             Lower bound on the moles existing a phase
   ZOPHLOI(K)            Lower bound on the moles existing a phase for infeas.
   NLPINF                Infeasible NLP objective
   ZOSPINF(S,K)          Equation in the minimisation of infeasibilities
   ZOSPINF2(S,K)
   MASSINF(I)            Element balance in minimisation of infeasibilities
   LAGRANGE(JM)          Lagrangian from feasible minimisations
   LAGINF(JM)            Lagrangian from infeasible minimisations
   ZOPHASE(S,K)          Pure binary constraint that controls phase existance
   CUTS(JM)              Binary cuts
   test;
*---------------------------------------------------------------------------
*Equations for feasible NLP subproblems

NLPOBJ..        ZOAU =E= SCALEFAC* SUM(K,
                         SUM(S$(SK(S,K)$Yparam(S,K)),
                         N(S,K)*(W(S,K)
                                 +LOG(P) +(LOG(N(S,K))-LOG(NTOT(K)))
                          )));

TMOLS(K)..      NTOT(K) =E= SUM(S$(SK(S,K)), N(S,K));

MASSBAL(I)..            FEED(I) =E= SUM(K,SUM(S$(SK(S,K)),A(I,S)*N(S,K)));

ZOSPF(S,K)$SK(S,K)..          N(S,K)-SCALEFAC*1000*Yparam(S,K) =L= 0 ;

ZOSPF2(S,K)$SK(S,K)..      -N(S,K)+LOWMOL*Yparam(S,K) =L= 0 ;

{{Note that the binary variables appear only in the constraints
            ZOSPF(S,K) and ZOSPF2(S,K)                }}

*---------------------------------------------------------------------------
{{Equations for NLP subproblems with minimisation of infeasibilities
Minimise a scalar that will be the maximum violation of any inequality
constraint containing the binary variables}}

NLPINF..        ZINF =E= ALPHA;

{{Note that the binary constraints appear only in the constraints EXISTP(I) }}

MASSINF(I)..  FEED(I) =E= SUM(K,SUM(S$(SK(S,K)),A(I,S)*NINF(S,K)));

ZOSPINF(S,K)$(SK(S,K))..      NINF(S,K)-SCALEFAC*1000*Yparam(S,K) =L= ALPHA  ;

ZOSPINF2(S,K)$(SK(S,K)).. -NINF(S,K)+LOWMOL*Yparam(S,K) =L= ALPHA ;
```



```
*---------------------------------------------------------------------------
{{Equations for the MILP master problems
Write the Lagrangian function indexed over J so that at the Jth iteration
J Lagrangians will be added.  The Lagrangian is written in terms of the
solution levels from the primal for each iteration that have been stored in
parameter Nparam() and the multipliers from constraints containing the the
binary variables stored in parameter Lmult(). }}


LAGRANGE(J)$(1-INFES(J))..
{{Note the use of Nparam to prevent logs of negatives}}
ZOAL =G= SCALEFAC*SUM(K,
              SUM(S$(SK(S,K)$Nparam(S,K,J)),
              Nparam(S,K,J)*(W(S,K)
               + (  LOG(P) +LOG(Nparam(S,K,J)) -LOG(NTOTparam(K,J))  )
                                )) +
         SUM(K, SUM(S$SK(S,K),
                   +Lmult(S,K,J)*( Nparam(S,K,J)-SCALEFAC*1000*Y(S,K))
      {{new*****}} +Lmult2(S,K,J)*( -Nparam(S,K,J)+LOWMOL*Y(S,K))
                                                                    ));

{{The infeasible Lagrangian has no objective function terms, only mixed
constraints containing the binaries}}

LAGINF(J)$(INFES(J))..
       SUM(K, SUM(S$SK(S,K),
                  +Lmult(S,K,J) *(  Nparam(S,K,J)-SCALEFAC*1000*Y(S,K) )
      {{new******}} +Lmult2(S,K,J)*( -Nparam(S,K,J)+LOWMOL*Y(S,K) )
                       ))=L= 0 ;

{{Pure binary constraints}}

ZOPHASE(S,K)$(SK(S,K)) ..                  Y(S,K) =L= YP(K)    ;
CUTS(J)..       SUM(S,SUM(K$(SK(S,K)),SIGN(Ysave(S,K,J)-0.5)*Y(S,K)))+
                SUM(K,SIGN(YPsave(K,J)-0.5)*YP(K))=L=
                SUM(S,SUM(K$(SK(S,K)),Ysave(S,K,J)))+
                SUM(K,YPsave(K,J))  -1;

{{Note that only the binary variables appear in the constraints EXISTP(I) }}

*---------------------------------------------------------------------------
{{Declare the NLP subproblems and the MILP master problem models}}
MODEL NLPSUB/NLPOBJ,TMOLS,MASSBAL,ZOSPF,ZOSPF2/;

MODEL INFNLP/NLPINF,MASSINF,ZOSPINF,ZOSPINF2/;

MODEL MASTER /LAGRANGE,LAGINF,ZOPHASE,CUTS,test/;

{{MASTER.WORK = 2000;}}

*---------------------------------------------------------------------------
{{Initialise algorithm parameters and sets}}
JITER(JM)=NO;
J(JM)=NO;
JLOOPCTL(JM)=YES;
INFES(JM)=0;

{{Initialise the parameters Ysave,YPsave to also contain the initial binary
combination guess}}
```

```
Ysave(S,K,JITER)=Yparam(S,K);
YPsave(K,JITER)=YPparam(K);
Y.L(S,K)=Yparam(S,K);
YP.L(K)=YPparam(K);
Yopt(S,K)=Yparam(S,K);
YPopt(K)=YPparam(K);
Jopt=2;

*---------------------------------------------------------------------------
*MAIN LOOP OF GENERALIZED BENDERS ALGORITHM

LOOP  (JLOOP$JLOOPCTL(JLOOP),
   {{initialize the variables}}
   N.L(S,K)$(SK(S,K))=0.001;
   NTOT.L(K)=0.001;

   {{ Update dynamic sets
   Shift current element of JITER(JM) }}

   JITER(JM) = YES$(ORD(JM) EQ ORD(JLOOP));

   {{Add element to J(JM)}}
   J(JLOOP) = YES;

   {{Update binary storage parameters}}
   Yparam(S,K)        = Y.L(S,K);
   Ysave(S,K,JITER) = Yparam(S,K);

   YPparam(K)        = YP.L(K);
   YPsave(K,JITER) = YPparam(K);
   DISPLAY "*************BINARY COMBINATION************",Y.L,YP.L;
   SOLVE NLPSUB USING NLP MINIMISING ZOAU;

   {{Flag if primal is infeasible for current selection of binaries}}
   NOTFEAS = 1$(NLPSUB.MODELSTAT EQ 5);

   {{Store feasibility flag}}
   INFES(JITER) = NOTFEAS;
   {{record the objective function value for fun}}
   NLPrec(JITER)$(1-NOTFEAS) = ZOAU.L;

   {{Check for better solution only from feasible primal subproblems}}
   IF (((1-NOTFEAS) AND (ZOAU.L LT ZOAUP)),
      JOPT = ORD(JLOOP);
      Yopt(S,K) = Yparam(S,K);
      YPopt(K) = YPparam(K);
      ZOAUP = ZOAU.L );

   {{Solve the minimisation of infeasibilities if necessary}}
   IF ( NOTFEAS, SOLVE INFNLP USING NLP MINIMIZING ZINF);

$ONTEXT
   Store the solution values of the continuous variables
   and marginal values and update the upper bound in the MILP
   master.  The assignments are made based on upon the feasibility
   of the NLP subproblem. If the NLP was feasible the solution
   levels of the NLP are assigned to the storage parameter, of not the
   infeasible minimisation are assigned to the parameters.
   Control is done with the scalar NOTFEAS.
```



```
$OFFTEXT

   Nparam(S,K,JITER) =    N.L(S,K)$(1-NOTFEAS) +    NINF.L(S,K)$NOTFEAS;
   NTOTparam(K,JITER)=    NTOT.L(K)$(1-NOTFEAS);
   Lmult(S,K,JITER)   =  - ZOSPF.M(S,K)$(1-NOTFEAS)   - ZOSPINF.M(S,K)$NOTFEAS;
   Lmult2(S,K,JITER) =  - ZOSPF2.M(S,K)$(1-NOTFEAS)  - ZOSPINF2.M(S,K)$NOTFEAS;
   {{Lmult3(K,JITER) =  - ZOPHLO.M(K)$(1-NOTFEAS)    - ZOPHLOI.M(K)$NOTFEAS;}}
   ZOAL.UP=ZOAUP;
   SOLVE MASTER USING MIP MINIMIZING ZOAL;
   {{Check the stopping criterion}}

   JLOOPCTL(JM)$(MASTER.MODELSTAT EQ 4 ) = NO;
   JLOOPCTL(JM)$(MASTER.MODELSTAT EQ 5 ) = NO;
   JLOOPCTL(JM)$(MASTER.MODELSTAT EQ 6 ) = NO;
   JLOOPCTL(JM)$(MASTER.MODELSTAT EQ 7 ) = NO;
   JLOOPCTL(JM)$(MASTER.MODELSTAT EQ 9 ) = NO;
   JLOOPCTL(JM)$(MASTER.MODELSTAT EQ 10) = NO;
   ZOAL.LO = ZOAL.L;
   MIPrec(JITER)=ZOAL.LO;
   DISPLAY MASTER.MODELSTAT;
   {{JLOOPCTL(JM)$((ZOAL.UP-ZOAL.LO)LE 1E-10)=NO;}}
   DISPLAY "!!!!!!!!!!!!!!!END OF LOOP!!!!!!!!!!!!!!!!!!!!";
   DISPLAY "ZOAL.LO",ZOAL.LO,"ZOAL.UP",ZOAL.UP,"Lmult",Lmult,
    "Lmult2",Lmult2,"Zospf",ZOSPF.M;);

DISPLAY "OPTIMAL SOLUTION FOUND", JOPT,ZOAUP,Yopt,YPopt,Nparam ;
DISPLAY NLPrec,MIPrec;
```

Appendix B

Decomposition Based Global Optimization FORTRAN Program

```
      PROGRAM GOP

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "COMM.INC"

      CHARACTER*7  NAMEDIROUT
      CHARACTER*10 NAMEDIRINP
      CHARACTER*27 NAMEDIRBASE
      CHARACTER*15 NAMEFILE
      CHARACTER*6  NAMEEXT(5),NAMEEXTINP
      CHARACTER*4  EXTNO
      CHARACTER*56 NAMEINP
      CHARACTER*55 NAMEOUT(3)

      REAL*8 XFEED(3)
      REAL   CPU,CPUNEW,CPUOLD

      NAMEDIRINP  = "/inputdata"
      NAMEDIRBASE = "/home/cliff/fortran/gop/rxn"
      NAMEDIROUT  = "/timing"
      NAMEFILE    = "/bute-wate_____"

      IW1=0           ! limited outout from GLOBAL
      ICONCHECK=1     ! use convexity check
      MAXMAJ=5000     ! maximum number of major iters in GLOBAL


      READ(*,*) ITOTCASE
      WRITE(*,*) "opening output file ",
     &NAMEDIRBASE//NAMEDIROUT//'/cputimes.dat'


      OPEN (UNIT=14,FILE=NAMEDIRBASE//NAMEDIROUT//'/bw.dat')

      DO ICASE=1,ITOTCASE
         READ(*,*)
         READ(*,*) EXTNO
         WRITE(*,*) "extno",EXTNO
         NAMEEXT(1)=".A"//EXTNO
         NAMEEXT(2)=".B"//EXTNO
         NAMEEXT(3)=".C"//EXTNO
         DO I=1,3
            NAMEOUT(I)=NAMEDIRBASE//NAMEDIROUT//NAMEFILE//NAMEEXT(I)
         ENDDO
         NAMEINP=NAMEDIRBASE//NAMEDIRINP//NAMEFILE//".dat"

         READ(*,*) XFEED(1), XFEED(2), XFEED(3)

         WRITE(*,*) NAMEOUT(1),XFEED(1),XFEED(2),XFEED(3)



         CPUOLD=CPUNEW
         ITFAIL=0


         CALL GLOBAL(ICONCHECK,NAMEOUT,NAMEINP,XFEED,MAXMAJ,
     &ITFAIL,CPUNEW)

         CPU=CPUNEW-CPUOLD
```



```
            WRITE(14,'(3(D12.3,2X),F8.4,2X,I2)')
     &         XFEED(1),XFEED(2),XFEED(3),CPU,ITFAIL

         ENDDO

         CLOSE(UNIT=14)

         END


         SUBROUTINE GLOBAL(ICONCHECK,NAMEOUT,NAMEINP,XFEED,MAXMAJ,
     &ITFAIL,ELAPSTIME)

c Decomposition based global optimization algorithm for an NRTL
c reacting or nonreacting systems.
c
c Based on:
c
c  "Decomposition Based and Branch and Bound Global Optimization
c   Approaches for the Phase Equilibrium Problem - McDonald and Floudas"
c
c This version has the option of using a convexity check
c to fathom regions
c
c if iconcheck = 1 the convexity check is used



      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "COMM.INC"
      INCLUDE "MINOSPAR.INC"

      CHARACTER*56 NAMEINP
      CHARACTER*55 NAMEOUT(3)
      REAL*8 XFEED(3)
      REAL   CPU,CPUSUB(2)
      INTEGER MAXMAJ

c declarations for local variables

      DOUBLE PRECISION ALPHA(MAXCI,MAXCI), ANTA(MAXCI),
     &                 ANTB(MAXCI),ANTC(MAXCI),GIBBS(MAXCI),
     &                 PSAT(CI), BNEW(NBASIC)

      REAL ELAPSTIME,ETIME,DTIME,XTIME(2)
      CALL READDATA (NAMEINP,ALPHA,ANTA,ANTB,ANTC,GIBBS,IRXN,IALPHA)

      DO I=1,CE
         NTOT(I)=XFEED(I)
      ENDDO

      OPEN(UNIT=1,FILE=NAMEOUT(1),STATUS="UNKNOWN")
      OPEN(UNIT=11,FILE=NAMEOUT(2),STATUS="UNKNOWN")
      OPEN(UNIT=12,FILE=NAMEOUT(3),STATUS="UNKNOWN")

      WRITE(11,98)
      WRITE(11,97) "   begin{tabular}{cccccc}"

97    FORMAT(A26)
```

```
98    FORMAT(" ITER PU",24X,"ML",19X,"PU-ML",16X,"TEST",12X,"RDUAL",
     &       1X,"NODES")


C Gas constant in Cal/K.mol.

      R   =1.9872


      DO 5 K=2,CK
         DO 5 I=1,CI
            DO 5 J=1,CE
               ATOMS(J,I,K)=ATOMS(J,I,1)
5     CONTINUE


c connected variables labelled
c   0 for a phase containing only non-connected variables
c   1 for a phase containing connected variables
c if there is a vapour phase, phase 1 is the vapour phase.
c set the nonbasic variables

      IFLAG=0
      M   =0
      DO 50 K=1,CK
         DO 50 I=1,CI
            L=0
            DO 60 WHILE((L.LT.CE).AND.(IFLAG.EQ.0))
               L=L+1
               IF (IBASICI(L).EQ.I.AND.IBASICK(L).EQ.K) THEN
                  IFLAG=1 ! The current species is basic.
               ENDIF
60          CONTINUE
            IF (IFLAG.EQ.0) THEN
               M=M+1
               INBASICI(M)=I
               INBASICK(M)=K
            ENDIF
            IFLAG=0
50    CONTINUE

      DO 40 J=1,CE
         DO 40 L=1,CE
            ABASIC(J,L)=ATOMS(J,IBASICI(L),IBASICK(L))
40    CONTINUE

c calculate the LU decomposition of the mass balance matrix for basic
c species.
c LU factorization needs to be done once only.

      CALL DGEFA(ABASIC,LDA,NBASIC,IPVT,INFO)

c NRTL model parameters

      IF (IALPHA.EQ.1) THEN
         DO 10 I = 1,CI
            DO 10 J = 1,CI
               GNRTL(I,J) =DEXP(-ALPHA(I,J)*TAU(I,J))
10       CONTINUE
      ENDIF
```



```
c calculation of vapour pressure using the Antoine equation.

      DO 260 I=1,CI
        PSAT(I)=10**(ANTA(I)-ANTB(I)/(TEMP+ANTC(I)))
        PSAT(I)=PSAT(I)/101.3D3
260   CONTINUE

c Setting the objective function constants appropriate to the
c type of problem to be solved.
c If the system is reacting

       IF (IRXN.EQ.1) THEN

c If a gas phase is included

          IF (IFLAGCV(1).EQ.0) THEN
             DO 310 I=1,CI
                GRT(I,1)=GIBBS(I)/R/TEMP+DLOG(PRESSURE)
                DO 320 K=2,CK
                   GRT(I,K)=GIBBS(I)/R/TEMP+ DLOG(PSAT(I))
320             CONTINUE
310          CONTINUE
          ELSE

c If a gas phase is not included

             DO 330 K=1,CK
                DO 330 I=1,CI
                   GRT(I,K)=GIBBS(I)/R/TEMP + DLOG(PSAT(I))
330          CONTINUE
          ENDIF

c If the system is not reacting

       ELSE

c If a gas phase is included

          IF (IFLAGCV(1).EQ.0) THEN
             DO 340 I=1,CI
                GRT(I,1)=0.00+DLOG(PRESSURE)
                DO 350 K=2,CK
                   GRT(I,K)=GIBBS(I)/R/TEMP + DLOG(PSAT(I))
350             CONTINUE
340          CONTINUE

c If a gas phase is not included

          ELSE
             DO 360 K=1,CK
                DO 360 I=1,CI
                   GRT(I,K)=0.00
360          CONTINUE
          ENDIF
       ENDIF

c initialisation of S values

      XCOUNT= -1
```

```
      DO 110 B=1,CB
         XCOUNT=XCOUNT+1
         XCOUNT2=XCOUNT
         DO 120 K=1,CK

c S values need only apply to connected variables.

            IF (IFLAGCV(K).EQ.1) THEN
               DO 125 I=1,CI
                  S(B,I,K)=INT(2*(MOD(XCOUNT2,2) )-1)
                  XCOUNT2=XCOUNT2/2-(MOD(XCOUNT2,2))/2
                  WRITE(*,*) "S",B,I,K,S(B,I,K)
125            CONTINUE
            ENDIF
120      CONTINUE
110   CONTINUE

c initialization of the algorithm
c note: NOTETOT generates a reference number for the nodes
c       the root node reference number is 0
c       NODESX (at the end of subroutine "DUAL"
c       is the number of unfathomed leaf nodes besides the one
c       selected for the next iteration.


      MAJITER   =0
      P(0)      =0
      NODETOT   =0
      NODESX    =0
      IRDUAL    =0
      NODEC     =0
      NODEL(0)  =1
      NODEIT(0) =0
      IFLAG     =0
      PU        = 1.D10
      ML        =-1.D10
      EPSILON2  = 1.D-14
      FEATOL    = 1.D-6


c Determine the upper and lower bounds on molar quantities

      DO 410 K=1,CK
         DO 410 I=1,CI
            DIV = 1D10
            DO 400 J=1,CE
               DIV1=NTOT(J)/ATOMS(J,I,K)
               IF (DIV1.LT.DIV) DIV=DIV1
400         CONTINUE
            UREG(0,I,K)=DIV
c           WRITE(*,*) "I,K,UREG",I,K,UREG(0,I,K)
410   CONTINUE

      DO 430 K=1,CK
         DO 430 I=1,CI
            LREG(0,I,K)=SMALL
430   CONTINUE

c limit the search space by restricting one of the
c basic components in one phase
```



```
      UREG(0,INBASICI(1),INBASICK(1))
&     =0.5*UREG(0,INBASICI(1),INBASICK(1))

c NMEM is defined for the nonbasic variables

      DO 420 L=1,CI*CK-CE
         I=INBASICI(L)
         K=INBASICK(L)
            NMEM(1,I,K)=0.5*UREG(0,I,K)
c            WRITE(*,*) "I,K,NMEM",I,K,NMEM(1,I,K)
420   CONTINUE

      DO 520 J=1,CE
         BNEW(J)=NTOT(J)
         DO 530 L=1,CI*CK-CE
            I=INBASICI(L)
            K=INBASICK(L)
            BNEW(J)=BNEW(J)-ATOMS(J,I,K)*NMEM(1,I,K)
530      CONTINUE
520   CONTINUE

      JOB=0
      CALL DGESL(ABASIC,LDA,NBASIC,IPVT,BNEW,JOB)

      DO 580 L=1,CE
         I=IBASICI(L)
         K=IBASICK(L)

c If the lower box bound on a variable is less than SMALL
c set that value to SMALL

          IF (BNEW(L).LE.SMALL) BNEW(L)=SMALL
            NMEM(1,I,K)=BNEW(L)
            WRITE(*,*) "I,K,NMEM",I,K,NMEM(1,I,K)
580      CONTINUE

       DO 2000 J=1,CE
          DO 2000 I=1,CI
             DO 2000 K=1,CK
                WRITE(*,*) "ATOMS",J,I,K,ATOMS(J,I,K)
2000   CONTINUE

       DO 2010 I=1,CI
          DO 2010 J=1,CI
             WRITE(*,*) "TAU",I,J,TAU(I,J)
2010   CONTINUE

       DO 2020 I=1,CI
          DO 2020 J=1,CI
             WRITE(*,*) "GNRTL",I,J,GNRTL(I,J)
2020   CONTINUE

       DO 2030 I=1,CI
          DO 2030 K=1,CK
             WRITE(*,*) "GRT",I,K,GRT(I,K)
2030   CONTINUE


       DO 2040 I=1,CE
```

```
            WRITE(*,*) "NTOT",I,NTOT(I)
2040    CONTINUE

c Start of main global optimization code

      DO 20 WHILE (IFLAG.EQ.0.AND.MAJITER.LE.MAXMAJ)

         MAJITER = MAJITER + 1

         ELAPSTIME=ETIME(XTIME)

         IF(MOD(MAJITER,10).EQ.0) THEN
            WRITE(*,*) "Time elapsed  = ",ELAPSTIME,"User = ",XTIME(1)
     &              ,"System =",XTIME(2)
         ENDIF



         CALL PRIMAL
         CALL SELECT
         CALL DUAL
         TEST = ((PU-ML)/DABS(PU))

         IF (TEST.LT.EPSILON.OR.NODESX.LT.0) THEN

            IFLAG=1

            WRITE(*,*) "Global Convergence Achieved"

            IF(NODESX.LT.0) THEN
               WRITE(*,*) "Entire Domain Fathomed"
            ELSE
               WRITE(*,*) "Convergence tolerance met"
            ENDIF

            DO 30 K=1,CK
               DO 30 I=1,CI
                  WRITE(1,*) "N(I,K)",I,K,NMEM(MINITER,I,K)
30             CONTINUE
               WRITE(1,*) 'MUMEM',MUMEM(MINITER)

         ENDIF
20    CONTINUE

      CLOSE(UNIT=12)
      CLOSE(UNIT=11)
      CLOSE(UNIT=1)

c if convergence has not been accomplished in max. interations

      IF(MAJITER.GE.MAXMAJ) ITFAIL=1

      RETURN
      END



C-------------------------------------------------------------------
C this subroutine evaluates the primal problem
```



```
      SUBROUTINE PRIMAL
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

C data type declarations for included parameters

      INCLUDE "COMM.INC"

      DOUBLE PRECISION NPHAS(CK),
     &                 TPSI(CI,CK),TLAM(CI,CK),
     &                 PCUR,XN(CI*CK),XNLO(CI,CK),XNHI(CI,CK)

      WRITE(1,*) "____________Primal____________"
      WRITE(1,*) "MAJITER",MAJITER

      DO 50 K=1,CK
         NPHAS(K)=0
         DO 50 I=1,CI
            NPHAS(K)=NPHAS(K)+NMEM(MAJITER,I,K)
50    CONTINUE

C PSIMEM and LAMMEM need be calculated for connected variables only

      DO 20 K=1,CK

C do for connected variables only

         IF (IFLAGCV(K).EQ.1) THEN
            DO 25 I=1,CI
               TPSI(I,K)=0
               TLAM(I,K)=0
               DO 40 J=1,CI
                  TPSI(I,K)=TPSI(I,K)+GNRTL(J,I)*NMEM(MAJITER,J,K)
40             CONTINUE
               PSIMEM(MAJITER,I,K)= NMEM(MAJITER,I,K)/TPSI(I,K)


c if NMEM is zero the multiplier is not defined so set it to 0

               IF (NMEM(MAJITER,I,K).LE.SMALL) THEN
                  LAMMEM(MAJITER,I,K)=0
               ELSE
                  DO 45 J=1,CI
                     TLAM(I,K)=TLAM(I,K)+
     &                  GNRTL(J,I)*TAU(J,I)*NMEM(MAJITER,J,K)
45                CONTINUE
                  LAMMEM(MAJITER,I,K)=-TLAM(I,K)/TPSI(I,K)
               ENDIF
25          CONTINUE
         ENDIF
20    CONTINUE

      PCUR=0
      DO 10 K=1,CK
         DO 10 I=1,CI
            TPCUR=0

C for the liquid phases only

            IF (IFLAGCV(K).EQ.1) THEN
              DO 30 J=1,CI
```

```
                TPCUR=TPCUR+GNRTL(I,J)*TAU(I,J)*PSIMEM(MAJITER,J,K)
30              CONTINUE
                PCUR=PCUR+NMEM(MAJITER,I,K)*TPCUR
              ENDIF

              PCUR=PCUR+NMEM(MAJITER,I,K)*
     &             ( GRT(I,K)+DLOG(NMEM(MAJITER,I,K))-DLOG(NPHAS(K)) )

10      CONTINUE

        IF (PCUR.LT.PU) THEN
           PU=PCUR
           MINITER=MAJITER

C local search to refine the upper bound.

           IX2N=0
           DO 60 K=1,CK
              DO 60 I=1,CI
                 IX2N=IX2N+1
                 XN(IX2N)=NMEM(MAJITER,I,K)
60         CONTINUE

           IBND=0
           CALL RXN(IBND,XNLO,XNHI,XN,OBJ)

           WRITE(12,*)
           WRITE(12,*) "ITER=",MAJITER
           WRITE(12,*) "  I  K   N(I,K)"
           IX2N=0
           DO 80 K=1,CK
              DO 80 I=1,CI
                 IX2N=IX2N+1
                 WRITE(12,99) I,K,XN(IX2N)
80         CONTINUE
           WRITE(12,98) OBJ
98         FORMAT("OBJECTIVE=",D15.9)
99         FORMAT(2I3,2X,D15.9)

           IF (OBJ.LT.PU) THEN
               PU=OBJ

c fathom nodes which have MU > PU

              DO 100 WHILE(MUMEM(NODEL(NODESX)).GT.PU
     &           .AND. NODESX.GE.1)
                 WRITE(*,*) MUMEM(NODEL(NODESX)),PU,NODESX
                 NODESX=NODESX-1
100           CONTINUE


              MINITER=0
              IX2N=0
              DO 70 K=1,CK
                 DO 70 I=1,CI
                    IX2N=IX2N+1
                    NMEM(MINITER,I,K)=XN(IX2N)
70            CONTINUE
              ENDIF
       ENDIF
```



```
        IF (IW1.EQ.1) WRITE(1,*)'          PU=',PU
        RETURN
        END

C_____________________________________________________________________
C Subroutine selects the parameters of the previous Lagrangians
C for the dual problem in the global optimization algorithm.

        SUBROUTINE SELECT
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INCLUDE "COMM.INC"

C number for root node

        IR=0
        IKT=NODEC
        IFLAG=0
        I=0

        DO 10 WHILE (IFLAG.EQ.0)
              I = I+1
              IF (IKT.GT.IR) THEN
                 PL(I)= NODEIT(IKT)
                 IKT = P(IKT)
              ELSE
                 IFLAG = 1
              ENDIF
10      CONTINUE
        CPL=I-1

999     FORMAT( "        PL(",I3,")=",I3                    )
998     FORMAT(5I4)
997     FORMAT(" I ,IKT ,  P, NIT, PL ")

        RETURN
        END
C_____________________________________________________________________
c Relaxed dual phase of the global optimization algorithm

        SUBROUTINE DUAL
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INCLUDE 'COMM.INC'
        INCLUDE 'MINOSPAR.INC'
        DOUBLE PRECISION XN(MAXNB),N(CI,CK),TL(CI,CK),TU(CI,CK),
     &                   LXBOX(CI,CK),UXBOX(CI,CK),MU,
     &                   BLO(NBASIC),BUP(NBASIC),
     &                   XNLO(CI,CK),XNHI(CI,CK)

        REAL ELAPSTIME,ETIME,DTIME,XTIME(2)
        INTEGER FLAGUL(CB)
        EXTERNAL DUALOPT

c set box bounds using S

995     FORMAT( "        LXBOX   ("I2,",",I2")=",E10.5 )
996     FORMAT( "        UXBOX   ("I2,",",I2")=",E10.5 )

        DO 100 B=1,CB

c flag used to inhibit redundant problems.
```

```
          FLAGUL(B)=0
          DO 110 L=1,CNB

c note that the use of S here must match the
c initialization of S in the main program.
c ie with regard to the assignment of I and K

                I=INBASICI(L)
                K=INBASICK(L)

c for connected variables only

                IF (IFLAGCV(K).EQ.1) THEN
                   IF (S(B,I,K).EQ.1) THEN
                      UBOX(B,I,K)=UREG(NODEC,I,K)
                      LBOX(B,I,K)=NMEM(MAJITER,I,K)
                   ELSEIF (S(B,I,K).EQ.-1) THEN
                      UBOX(B,I,K)=NMEM(MAJITER,I,K)
                      LBOX(B,I,K)=LREG(NODEC,I,K)
                   ENDIF

c remove redundant problems

                   IF (DABS(UBOX(B,I,K)-LBOX(B,I,K)).LE.EPSILON2)
     &             THEN
                      FLAGUL(B)=1
                   ENDIF
                ENDIF
110        CONTINUE

           IF (FLAGUL(B).EQ.0) THEN
              DO 20 J=1,CE

                 BLO(J)=NTOT(J)
                 BUP(J)=NTOT(J)

                 DO 30 L=1,CI*CK-CE

                    I=INBASICI(L)
                    K=INBASICK(L)

c the bounds on connected mole are set by the boxes

                    IF (IFLAGCV(K).EQ.1) THEN
                       BLO(J)=BLO(J)-ATOMS(J,I,K)*UBOX(B,I,K)
                       BUP(J)=BUP(J)-ATOMS(J,I,K)*LBOX(B,I,K)

c the bounds on other nonbasic mole variables are set by the constraints
c imposed by the problem as a whole.

                    ELSE
                       BLO(J)=BLO(J)-ATOMS(J,I,K)*UREG(0,I,K)
                       BUP(J)=BUP(J)-ATOMS(J,I,K)*LREG(0,I,K)
                    ENDIF

30               CONTINUE
20            CONTINUE
              JOB=0
```



```
              CALL DGESL(ABASIC,LDA,NBASIC,IPVT,BLO,JOB)
              CALL DGESL(ABASIC,LDA,NBASIC,IPVT,BUP,JOB)

c check if a region is feasible in terms of the basic variables

690        FORMAT(I3,I3,E15.5,E15.5)
691        FORMAT('I  ','K  ','LBOX           ','UBOX           ')

              DO 80 L=1,CE
                 I=IBASICI(L)
                 K=IBASICK(L)

c If the lower box bound on a variable is less than SMALL
c set that value to SMALL
c Note that BLO and BUP must be correctly aligned with the
c basic variables

                 IF (BLO(L).LE.SMALL) BLO(L)=SMALL

c for connected variables check if these bounds
c give consistent constraint qualifications

                 IF (IFLAGCV(K).EQ.1) THEN

                    IF (S(B,I,K).EQ.-1) THEN
                       TEST=NMEM(MAJITER,I,K)-BLO(L)
                       IF (TEST.LT.EPSILON2) THEN
                          FLAGUL(B)=1
                       ELSE
                          LBOX(B,I,K)=BLO(L)
                          UBOX(B,I,K)
     &                    =NMEM(MAJITER,I,K)
                       ENDIF
                    ELSE
                       TEST=BUP(L)-NMEM(MAJITER,I,K)
                       IF (TEST.LT.EPSILON2) THEN
                          FLAGUL(B)=1
                       ELSE
                          LBOX(B,I,K)=
     &                    NMEM(MAJITER,I,K)
                          UBOX(B,I,K)=BUP(L)
                       ENDIF
                    ENDIF

c for non-connected variables check if the subproblem is feasible.

                 ELSE

                    IF (BUP(L).LT.SMALL) FLAGUL(B)=1

                 ENDIF
80            CONTINUE

c If lower and upper bounds coincide do not perform optimization.

           IF (DABS(UBOX(B,I,K)-LBOX(B,I,K)).LE.EPSILON) FLAGUL(B)=1
           ENDIF
100     CONTINUE
```

```
c Loop over all box regions,
c setting up and solving relaxed dual problems.

      DO 190 B=1,CB
      IF (FLAGUL(B).EQ.0) THEN

c  convexity check: only for 2 phases

         IF (ICONCHECK.EQ.1) THEN

            DO K=1,CK
               DO I=1,CI
                  XNLO(I,K)=LBOX(B,I,K)
                  XNHI(I,K)=UBOX(B,I,K)
               ENDDO
            ENDDO


            CALL DLOHIT(ICONVEX,XNLO,XNHI)

            IF (ICONVEX.EQ.1) THEN

               IBND=1

               JI=0
               DO K=1,CK
                  DO I=1,CI
                     JI=JI+1
                     XN(JI)=0.5*(XNHI(I,K)+XNLO(I,K))
                  ENDDO
               ENDDO

               CALL RXN(IBND,XNLO,XNHI,XN,OBJ)

               IF (OBJ.LT.PU) THEN

                  PU=OBJ
                  IX2N=0
                  DO 70 K=1,CK
                     DO 70 I=1,CI
                        IX2N=IX2N+1
                        NMEM(0,I,K)=XN(IX2N)
70                CONTINUE

               ENDIF

            ENDIF
         ENDIF

c minimize relaxed dual subproblem if the convexity check fails or
c is not used.

         IF (ICONVEX.EQ.0.OR.ICONCHECK.EQ.0) THEN

c if the convexity check is negative proceed with the relaxed dual
c subproblem.

c Calculate the upper and lower bounds on the x type variables
c (i.e. PSI ) for the current box region.
```



```
c Note that XLIN applies only to the connected variables.

         DO 200 K=1,CK
         IF (IFLAGCV(K).EQ.1) THEN
         DO 205 I=1,CI
            TL(I,K)=0
            TU(I,K)=0
            DO 220 J=1,CI
               IF (J.NE.I) THEN
                  TL(I,K)=TL(I,K)+GNRTL(J,I)*UBOX(B,J,K)
                  TU(I,K)=TU(I,K)+GNRTL(J,I)*LBOX(B,J,K)
               ENDIF
220         CONTINUE

            LXBOX(I,K)=LBOX(B,I,K)/(LBOX(B,I,K)+TL(I,K))
            UXBOX(I,K)=UBOX(B,I,K)/(UBOX(B,I,K)+TU(I,K))

205      CONTINUE
         ENDIF
200      CONTINUE

c Calculate qualifying constraints to determine where
c Lagrangians should be evaluated in terms of the x values
c Loop over all regions of which the present region is a subdomain

         DO 230 IPL=1,CPL+1
            IF (IPL.LE.CPL) THEN
               KP=PL(IPL)
               FLAG=0

c flag for past Lagrangians

            ELSE
               KP=MAJITER
               FLAG=1

c flag for current Lagrangian

            ENDIF

c evaluate XLIN(MAXNOD,I,J,K)

            DO 240 I=1,CI
               DO 240 J=1,CI
                  DO 240 K=1,CK


                     T=GNRTL(J,I)*(TAU(J,I)+LAMMEM(MAJITER,I,K))
                     IF (T.GT.0) THEN
                        SS=+1
                     ELSE
                        SS=-1
                     ENDIF

c past Lagrangians
                  IF (FLAG.EQ.0) THEN

                     T=SS*(-NMEM(KP,j,K)+NMEM(MAJITER,j,K))

c check qualifying constraints
```

```
                        IF (T.GE.0) THEN
                           XLIN(KP,I,J,K)=LXBOX(I,K)
                        ELSE
                           XLIN(KP,I,J,K)=UXBOX(I,K)
                        ENDIF

c present Lagrangian
                     ELSEIF (FLAG.EQ.1) THEN

                        T=SS*S(B,J,K)

c check qualifying constraints

                        IF (T.GE.0) THEN
                          XLIN(KP,I,J,K)=LXBOX(I,K)
                        ELSE
                          XLIN(KP,I,J,K)=UXBOX(I,K)
                        ENDIF

                     ENDIF

240          CONTINUE
230       CONTINUE

          CALL DUALOPT(XN,OBJ)

c         count of relaxed dual problems solved
          IRDUAL=IRDUAL+1

c convert output from optimization routine into mole quantities.

          ICOUNT=0
          DO 250 K=1,CK
             DO 250 I=1,CI
                ICOUNT=ICOUNT+1
                IF (XN(ICOUNT).LT.SMALL) THEN
                   XN(ICOUNT)=SMALL
                ENDIF
                N(I,K)=XN(ICOUNT)
250       CONTINUE

         MU=OBJ
         IF (IW1.EQ.1) WRITE(1,*)'        OBJECTIVE =',OBJ
         IF (IW1.EQ.1) WRITE(1,*)'        INFORM    =',INFORM

c insert the solution from the relaxed dual problem into the
c list of data on nodes.

c Check for feasibility

          INFORM1=0
          DO 420 J=1,CE
             TEST=0
             DO 410 K=1,CK
                DO 410 I=1,CI
                   TEST=TEST+ATOMS(J,I,K)*N(I,K)
410          CONTINUE
             IF (DABS(TEST-NTOT(J)).GT.FEATOL) THEN
                INFORM1=1
```



```
                IF (IW1.EQ.1) THEN
                   WRITE(1,*) "****Mass Balance Violated****"
                   WRITE(1,*) 'TEST,NTOT(j)',TEST,NTOT(J)
                ENDIF
             ENDIF
420       CONTINUE

c Solutions greater than the upper bound are fathomed
c along with infeasible solutions.


          IF ((MU.LT.PU).AND.(INFORM1.EQ.0)
     &                          .AND.(INFORM .EQ.0)) THEN


             NODETOT=NODETOT+1
             NODESX =NODESX+1

             NODEIT(NODETOT)=MAJITER
             P(NODETOT)=NODEC
             MUMEM(NODETOT)=MU

             DO 300 I=1,CI
                DO 300 K=1,CK
                   UREG(NODETOT,I,K)=UBOX(B,I,K)
                   LREG(NODETOT,I,K)=LBOX(B,I,K)
                   NNODMEM(NODETOT,I,K)= N(I,K)
300          CONTINUE

c Insertion sort algorithm: places nodes in order of
c increasing corresponding MU value, the lowest value
c being at the top of the list, NODEL.

             I=0
             FLAG=0
             DO 310 WHILE (FLAG.EQ.0)
                I=I+1
                IF (I.LT.NODESX) THEN
                   J=NODEL(I)

c If MU is to be inserted first move other members
c of the list down before inserting.

                   IF (MU.LT.MUMEM(J)) THEN
                      DO 320 K=NODESX-1,I,-1
                         NODEL(K+1)=NODEL(K)
320                   CONTINUE
                      NODEL(I)=NODETOT

c flag indicates that the insertion has been done.

                      FLAG=1
                   ELSE
                      FLAG=0
                   ENDIF

c If MU is not inserted it is placed at the bottom of the list.

                ELSE
                   NODEL(I)=NODETOT
```

```
                  FLAG=1
                ENDIF

310           CONTINUE
            ENDIF
          ENDIF

          ENDIF ! endif for event of convexity check failure
190       CONTINUE

c select infimum of all mu values recorded

          NODEC=NODEL(1)

c remove the top element from the list

          DO 350 I=2,NODESX
             NODEL(I-1)=NODEL(I)
350       CONTINUE
          NODESX=NODESX-1

          ML=MUMEM(NODEC)

c******************** O U T P U T ************************************
          IF (IW1.EQ.1) THEN
             WRITE(1,998)
             WRITE(1,999) 0,NODEC,ML
             DO 400 I=1,5
               WRITE(1,999) I,NODEL(I),MUMEM(NODEL(I))
400          CONTINUE

             DO 405 K=1,CK
                DO 405 I=1,CI
                   WRITE(1,980) NODEC,I,K,LREG(NODEC,I,K)
                   WRITE(1,981) NODEC,I,K,UREG(NODEC,I,K)
405        CONTINUE
          ENDIF

980       FORMAT( "                  LREG(",I3,I3,I3,")= ",D30.5)
981       FORMAT( "                  UREG(",I3,I3,I3,")= ",D30.5)
999       FORMAT( "                ",I3 ,"       ",I3,"       ",D30.5)
998       FORMAT(  "   Ith Smallest Mu ","   Node  ","   Value of Mu    ")
c*********************************************************************

c set the value of NMEM for the next iteration.

          DO 330 I=1,CI
             DO 340 K=1,CK
                NMEM(MAJITER+1,I,K)=NNODMEM(NODEC,I,K)
340          CONTINUE
330       CONTINUE


          RETURN
          END

c_____________________________________________________________________
c solve the relaxed dual problem specified by RDUAL

          SUBROUTINE DUALOPT(XN,OBJ)
```



```
          IMPLICIT DOUBLE PRECISION (A-H,O-Z)

          INCLUDE "COMM.INC"
          INCLUDE "MINOSPAR.INC"
          COMMON  /XLAGDEFS/XLAGDEF,INITLAGDEF

c data type declarations for local variables

          DOUBLE PRECISION N(CI,CK)

c data type declarations for MINOS variables

          DOUBLE PRECISION  A(MAXNE),BL(MAXNB),BU(MAXNB),
         &                  XN(MAXNB),PI(MAXM),RC(MAXNB),Z(NWCORE),
         &                  XLAGDEF(100,CI,CI,CK)
          CHARACTER*8 NAMES(5)
          INTEGER*4   HA(MAXNE),HS(MAXNB)
          INTEGER     KA(MAXN+1),NAME1(NNAME),NAME2(NNAME),INITLAGDEF


          INITLAGDEF=0
          ISPECS=4
          IPRINT=0
          ISUMM =-6
          NOUT  =6
          BPLUS =1.D+10
          BMINUS=-BPLUS


          M=CPL+1+1+CE
          NN=CI*CK+1
          NB=NN+M
          NE=NN*M
          IOBJ=M
          NNCON=CPL+1
          NNOBJ=0
          NNJAC=CI*CK

*     ===================================================================
*     set the values in the constraint matrix
*
*              1             2   ....                      nnjac         nnjac+1
*
*        1     Lagrangians'  ***********   ***********   ***********      -1
*              Nonlinear
*        2     Jacobian      ***********   ***********   ***********      -1
*              .
*              .             ***********   ***********   ***********      -1
*              .
*     nncon    ***********   ***********   ***********   ***********      -1
*
*     nncon+1  ATOMS(1,1,1)  ATOMS(1,2,1)  ATOMS(1,1,2)  ATOMS(1,2,2)      0
*
*              ATOMS(2,1,1)                                                0
*
*     nncon+CE ATOMS(CE,CI,1)                           ATOMS(CE,CI,CK)   0
*
*        M     0.0000000000  0.000000000   0.000000000    0  Objective     1
*
*     ===================================================================
```

```
      NAMES(1)='RDual   '
      NAMES(2)='Mu      '
      NAMES(3)='Atoms   '
      NAMES(4)='Ranges  '
      NAMES(5)='Bounds  '

c clean up A

      DO 230 I=1,NE
         A(I)=0
230   CONTINUE


c mass balance constraints

      ICOUNT=0
      DO 100 K=1,CK
         DO 100 I=1,CI
            ICOUNT=ICOUNT+1
            DO 100 J=1,CE
               A((ICOUNT-1)*M+NNCON+J)=ATOMS(J,I,K)
100   CONTINUE

c objective function

      DO 30 I=1,NNJAC
         A(I*M)=0
30    CONTINUE

c mu in the Lagrangians

      DO 40 I=1,NNCON
         A(NNJAC*M+I)=-1
40    CONTINUE

c objective function

      A(NN*M)=1

      DO 50 I=1,NN+1
         KA(I)=(I-1)*M+1
50    CONTINUE

      ICOUNT=0
      DO 60 I=1,NN
         DO 60 J=1,M
            ICOUNT=ICOUNT+1
            HA(ICOUNT)=J
60    CONTINUE

c Bounds on mole variables

      ICOUNT=0
      DO 70 K=1,CK
         DO 70 I=1,CI
            ICOUNT=ICOUNT+1
            BL(ICOUNT)=LBOX(B,I,K)
            BU(ICOUNT)=UBOX(B,I,K)
70    CONTINUE
```



```
      DO 65 I=1,CI
         ICOUNT=ICOUNT+1
         BL(ICOUNT)=SMALL
         BU(ICOUNT)=10000
65    CONTINUE

c Bounds on mu

      BL(NN)=BMINUS
      BU(NN)=BPLUS

c Bounds on the slack variables for the lagrangians

      DO 80 I=NN+1,NN+NNCON
         BL(I)=0
         BU(I)=BPLUS
80    CONTINUE

c Bounds on the slacks for the mass balance constraints

      DO 110 I=1,CE
         BL(NN+NNCON+I)=-NTOT(I)
         BU(NN+NNCON+I)=-NTOT(I)
110   CONTINUE

c Bounds on the objective function

      BL(NN+M)=BMINUS
      BU(NN+M)=BPLUS

c initialization of variables

      frac=0.5
      ICOUNT=0
      DO 20 K=1,CK
         DO 20 I=1,CI
            ICOUNT=ICOUNT+1
            N(I,K)=frac*LBOX(B,I,K)+(1-frac)*UBOX(B,I,K)
            XN(ICOUNT)=N(I,K)
20    CONTINUE

      XN(NX)=10

      DO 25 I=1,M
         PI(I)=0
25    CONTINUE

      DO 150 I=1,NB
        HS(I)=0
150   CONTINUE

      DO 210 I=1,MAXNB
         RC(I)=0
210   CONTINUE

      INFORM=0
      MINCORE=0
      NS=0
      NINF=0
```

```
      SINF=0
      OBJ=0
      OBJADD=0

      DO 220 I=1,NWCORE
         Z(I)=0
220   CONTINUE

      CALL MISPEC(ISPECS,IPRINT,ISUMM,NWCORE,INFORM)

      CALL MINOSS('COLD',M,NN,NB,NE,NNAME,
     &            NNCON,NNOBJ,NNJAC,
     &            IOBJ,OBJADD,NAMES,
     &            A,HA,KA,BL,BU,NAME1,NAME2,
     &            HS,XN,PI,RC,
     &            INFORM,MINCORE,NS,NINF,SINF,OBJ,
     &            Z,NWCORE)

      OBJ=OBJ-OBJADD
      INFORM1=INFORM

      RETURN
      END

c_____________________________________________________________________
c subroutine used to set up MINOSS for the nonconvex problem.

      SUBROUTINE RXN(IBND,XNLO,XNHI,XN,OBJ)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      INCLUDE "COMM.INC"
      INCLUDE "MINOSPAR.INC"

c if IBND=1 then XNLO and XNHI are used to set the bounds
c on the mole variables.

c data type declarations for MINOS variables

      DOUBLE PRECISION  A(MAXNE),BL(MAXNB),BU(MAXNB),
     &                  XN(MAXNB),PI(MAXM),RC(MAXNB),Z(NWCORE),
     &                  XNLO(CI,CK),XNHI(CI,CK)
      DOUBLE PRECISION N(CI,CK)

      CHARACTER*8 NAMES(5)
      INTEGER*4   HA(MAXNE),HS(MAXNB)
      INTEGER     KA(MAXN+1),NAME1(NNAME),NAME2(NNAME)

      ISPECS=3
      IPRINT=0
      ISUMM =-6
      NOUT  =6
      BPLUS =1.D+10
      BMINUS=-BPLUS


      M=CE
      NN=CI*CK
      NB=NN+M
      NE=NN*M
      IOBJ = 0
```



```
      NNCON=0
      NNOBJ=CI*CK
      NNJAC=0

*************************************************************************
*     set the values in the constraint matrix
*
*              1             2   ....                        nn
*
*         1   ATOMS(1,1,1)  ATOMS(1,2,1)  ATOMS(1,1,2)  ATOMS(1,2,2)
*
*         .   ATOMS(2,1,1)
*
*        CE   ATOMS(CE,CI,1)                             ATOMS(CE,CI,CK)
*
*
*************************************************************************
c clean up A

      DO 230 I=1,NE
         A(I)=0
230   CONTINUE


c mass balance constraints

      ICOUNT=0
      DO 100 K=1,CK
         DO 100 I=1,CI
            DO 100 J=1,CE
               ICOUNT=ICOUNT+1
               A(ICOUNT)=ATOMS(J,I,K)
               HA(ICOUNT)=J
100   CONTINUE

      DO 50 I=1,NN+1
         KA(I)=(I-1)*M+1
50    CONTINUE

c Bounds on mole variables

      ICOUNT=0
      DO 70 K=1,CK
         DO 70 I=1,CI
            ICOUNT=ICOUNT+1
            IF (IRXN.EQ.0) THEN
               BL(ICOUNT)=SMALL
               BU(ICOUNT)=BPLUS
            ELSE
               BL(ICOUNT)=XNLO(I,K)
               BU(ICOUNT)=XNHI(I,K)
            ENDIF
70    CONTINUE


c Bounds on the slacks for the mass balance constraints
      DO 110 I=1,CE
         BL(NN+I)=-NTOT(I)
         BU(NN+I)=-NTOT(I)
110   CONTINUE
```

```
c initialization of variables

      FRAC=0.5
      ICOUNT=0
      DO 20 K=1,CK
         DO 20 I=1,CI
            ICOUNT=ICOUNT+1
            XN(ICOUNT)=NMEM(MAJITER,I,K)
20    CONTINUE

      DO 25 I=1,M
         PI(I)=0
25    CONTINUE

      DO 150 I=1,NB
        HS(I)=0
150   CONTINUE

      DO 210 I=1,MAXNB
         RC(I)=0
210   CONTINUE

      INFORM=0
      MINCORE=0
      NS=0
      NINF=0
      SINF=0
      OBJ=0
      OBJADD=0

      CALL MISPEC(ISPECS,IPRINT,ISUMM,NWCORE,INFORM)
      CALL MINOSS('COLD',M,NN,NB,NE,NNAME,
     &            NNCON,NNOBJ,NNJAC,
     &            IOBJ,OBJADD,NAMES,
     &            A,HA,KA,BL,BU,NAME1,NAME2,
     &            HS,XN,PI,RC,
     &            INFORM,MINCORE,NS,NINF,SINF,OBJ,
     &            Z,NWCORE)

      OBJ=OBJ-OBJADD
      INFORM1=INFORM

      RETURN
      END

c-----------------------------------------------------------------------

      SUBROUTINE FUNOBJ(MODE,NN,X,F,DELF,NSTATE,NPROB,Z,NWCORE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "COMM.INC"
      DOUBLE PRECISION X(NN),DELF(NN),Z(NWCORE)
      COMMON  /M1FILE/IREAD,IPRINT,ISUMM

      IF (NPROB.EQ.1) CALL OBJ1(NN,X,F,DELF,NSTATE,NPROB)
      RETURN
      END

c-----------------------------------------------------------------------
      SUBROUTINE FUNCON(MODE,M,NN,NJAC,X,G,DELG,
```



```
     &                  NSTATE,NPROB,Z,NWCORE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "COMM.INC"
      DOUBLE PRECISION G(M),DELG(M,NN),Z(NWCORE)
      COMMON  /M1FILE/IREAD,IPRINT,ISUMM
      IF (NPROB.EQ.2) CALL CON2(NN,M,X,G,DELG,NSTATE,NPROB)
      RETURN
      END
c-----------------------------------------------------------------------
c Objective and obj gradients for the nonconvex problem.

      SUBROUTINE OBJ1(NN,X,F,DELF,NSTATE,NPROB)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "COMM.INC"
      DOUBLE PRECISION X(NN),DELF(NN),NPHAS(CK),TPSI(CI,CK),
     &                 PSI(CI,CK),N(CI,CK)

      TINY=1.D-2*SMALL
      IX2N=0

      DO 10 K=1,CK
         DO 10 I=1,CI
           IX2N=IX2N+1
           N(I,K)=X(IX2N)
10      CONTINUE


      DO 20 K=1,CK
         NPHAS(K)=0
         DO 20 I=1,CI
            NPHAS(K)=NPHAS(K)+N(I,K)
            TPSI(I,K)=0
               DO 40 J=1,CI
               TPSI(I,K)=TPSI(I,K)+GNRTL(J,I)*N(J,K)
40          CONTINUE
         PSI(I,K)= N(I,K)/TPSI(I,K)
20    CONTINUE


      F=0
      ICOUNT=0
      DO 50 K=1,2
         DO 50 I=1,CI
            ICOUNT=ICOUNT+1

            DELF(ICOUNT)=( GRT(I,K)+DLOG(N(I,K))-DLOG(NPHAS(K)) )
            F= F+N(I,K)*DELF(ICOUNT)

            TPCUR=0

            IF (IFLAGCV(K).EQ.1) THEN
               DO 30 J=1,CI
                  TPCUR=TPCUR+GNRTL(I,J)*TAU(I,J)*PSI(J,K)
30             CONTINUE
               DELF(ICOUNT)=DELF(ICOUNT)+TPCUR
               F=F+N(I,K)*TPCUR

               K1=K
               DO 60 I1=1,CI
                  TEMP=0.D0
```

```
                  DO 70 J=1,CI
                     TEMP=TEMP-GNRTL(I1,J)*TAU(I1,J)*
     &               GNRTL(I,J)*N(J,K)/(TPSI(J,K)**2)
                     IF (J.EQ.I) THEN
                        TEMP = TEMP + GNRTL(I1,J)*TAU(I1,J)
     &                  /TPSI(J,K)
                     ENDIF
70                CONTINUE
                  DELF(ICOUNT)=DELF(ICOUNT)+N(I1,K)*TEMP
60             CONTINUE
            ENDIF

50    CONTINUE

      RETURN
      END
c_____________________________________________________________________
c Constraints for the relaxed dual problem

      SUBROUTINE CON2(NN,M,X,G,DELG,NSTATE,NPROB)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON  /M1FILE/IREAD,IPRINT,ISUMM

      INCLUDE "COMM.INC"
      INTEGER IX2N,KP,B
      DOUBLE PRECISION G(M)          ,DELG(M,NN)   ,
     &                 N(CI,CK)      ,X(NX)        ,NPHAS(CK) ,MU,
     &                 XLAG(MAXNOD)  ,DLAG(MAXNOD ,CI,CK)


      TINY=1.D-2*SMALL
      IX2N=0

      DO 10 K=1,CK
         NPHAS(K)=0
         DO 10 I=1,CI
           IX2N=IX2N+1
           N(I,K)=X(IX2N)
           NPHAS(K)=NPHAS(K)+N(I,K)
10    CONTINUE

c initialize G

      DO 100 I=1,M
         G(I)=0
100   CONTINUE

      DO 200 L=1,CPL+1
         IF (L.LE.CPL) THEN
            KP=PL(L)
         ELSE
            KP=MAJITER
         ENDIF
         XLAG(L)=0
         DO 210 K=1,CK
            DO 220 I=1,CI

C for the liquid phases only

               IF (IFLAGCV(K).EQ.1) THEN
```



```
                  DO 230 J=1,CI
                     XLAG(L)=XLAG(L)+XLIN(KP,I,J,K)*GNRTL(J,I)*
     &               (TAU(J,I)+LAMMEM(KP,I,K))*(N(J,K)-NMEM(KP,J,K))
230               CONTINUE
                  XLAG(L)=XLAG(L)+N(I,K)*(-LAMMEM(KP,I,K))
               ENDIF

               XLAG(L)=XLAG(L)+N(I,K)*(DLOG(N(I,K))+GRT(I,K))
220         CONTINUE
            XLAG(L)=XLAG(L)-NPHAS(K)*DLOG(NPHAS(K))
210      CONTINUE
         G(L)=(XLAG(L)-MU)

         DO 215 K=1,CK
            DO 225 J=1,CI
               DLAG(L,J,K)=0

c for liquid phases only

               IF (IFLAGCV(K).EQ.1) THEN
                  DO 235 I=1,CI
                     DLAG(L,J,K)=DLAG(L,J,K)+XLIN(KP,I,J,K)*
     &               GNRTL(J,I)*(TAU(J,I)+LAMMEM(KP,I,K))
235                  CONTINUE
                     DLAG(L,J,K)=DLAG(L,J,K)-LAMMEM(KP,J,K)
               ENDIF

               DLAG(L,J,K)=DLAG(L,J,K)+DLOG(N(J,K))+GRT(J,K)
               DLAG(L,J,K)=DLAG(L,J,K)-DLOG(NPHAS(K))
225         CONTINUE

215      CONTINUE

       ICOUNT=0
       DO 240 K=1,CK
          DO 240 I=1,CI
             ICOUNT=ICOUNT+1
             DELG(L,ICOUNT)=DLAG(L,I,K)
c             WRITE(*,*) L,ICOUNT,DELG(L,ICOUNT)

240    CONTINUE

200    CONTINUE

       RETURN
       END
c_____________________________________________________________________
      SUBROUTINE MATMOD
      RETURN
      END
c_____________________________________________________________________
      SUBROUTINE READDATA (NAMEINP,ALPHA,ANTA,ANTB,ANTC,GIBBS,
     &IRXN,IALPHA)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE 'COMM.INC'
      CHARACTER*56 NAMEINP
      DOUBLE PRECISION ALPHA(MAXCI,MAXCI),ANTA(MAXCI),ANTB(MAXCI),
     &                 ANTC(MAXCI),GIBBS(MAXCI)

      OPEN(UNIT=2,FILE=NAMEINP,STATUS='UNKNOWN')
```

```
      REWIND 2

      DO 2000 I=1,6
         READ(2,*)
         WRITE(*,*)
2000  CONTINUE

      READ (2,901)  IRXN
      WRITE(*,901)  IRXN
      READ(2,901)   !CI
      WRITE(*,901)  !CI
      READ(2,901)   !CK
      WRITE(*,901)  !CK

      DO 2010 I=1,CK
         READ(2,901) IFLAGCV(I)
         WRITE(*,901) IFLAGCV(I)
2010  CONTINUE
      DO 2020 I=1,MAXCK-CK
         READ(2,*)
         WRITE(*,*)
2020  CONTINUE

      READ(2,901)  !CE
      WRITE(*,901) !CE
      READ(2,902) TEMP
      WRITE(*,902) TEMP
      READ(2,902) PRESSURE
      WRITE(*,902) PRESSURE

      DO 2030 I=1,CE
         READ(2,902) NTOT(I)
         WRITE(*,*) "I,NTOT",I,NTOT(I)
2030  CONTINUE
      DO 2035 I=1,MAXCE-CE
         READ(2,*)
         WRITE(*,*)
2035  CONTINUE
      READ(2,*)
      WRITE(*,*)
      READ(2,901) IALPHA
      WRITE(*,*) "IALPHA",IALPHA
      DO 2040 I=1,4
         READ(2,*)
         WRITE(*,*)
2040  CONTINUE
      DO 2052 J=1,CE
         READ(2,906) (ATOMS(J,I,1),I=1,CI)
         WRITE(*,906)(ATOMS(J,I,1),I=1,CI)
2052  CONTINUE
      DO 2057 I=1,MAXCE-CE
         READ(2,*)
         WRITE(*,*)
2057  CONTINUE

      DO 2051 I=1,5
         READ(2,*)
         WRITE(*,*)
2051  CONTINUE
```



```
      DO 2060 I=1,CI
         READ(2,906) (TAU(I,J),J=1,CI)
         WRITE(*,*)(TAU(I,J),J=1,CI)
2060  CONTINUE

      DO 2070 I=1,MAXCI-CI
         READ(2,*)
         WRITE(*,*)
2070  CONTINUE

      DO 2080 I=1,5
         READ(2,*)
         WRITE(*,*)
2080  CONTINUE

      DO 2090 I=1,CI
        READ(2,906) (GNRTL(I,J),J=1,CI)
         WRITE(*,906)(GNRTL(I,J),J=1,CI)
2090  CONTINUE

      DO 2100 I=1,MAXCI-CI
         READ(2,*)
         WRITE(*,*)
2100  CONTINUE

      DO 2110 I=1,5
         READ(2,*)
         WRITE(*,*)
2110  CONTINUE

      DO 2120 I=1,CI
         READ(2,906) (ALPHA(I,J),J=1,CI)
         WRITE(*,906)(ALPHA(I,J),J=1,CI)
2120  CONTINUE

      DO 2130 I=1,MAXCI-CI
         READ(2,*)
         WRITE(*,*)
2130  CONTINUE

      DO 2140 I=1,5
         READ(2,*)
         WRITE(*,*)
2140  CONTINUE

      DO 2200 I=1,CI
         READ(2,905) ANTA(I),ANTB(I),ANTC(I)
         WRITE(*,905) ANTA(I),ANTB(I),ANTC(I)
2200  CONTINUE

      DO 2205 I=1,MAXCI-CI
         READ(2,*)
         WRITE(*,*)
2205  CONTINUE

      DO 2210 I=1,3
         READ(2,*)
2210  CONTINUE
```

```
      DO 2220 I=1,CI
         READ(2,907) GIBBS(I)
         WRITE(*,907) GIBBS(I)
2220  CONTINUE

      DO 2230 I=1,MAXCI-CI
         READ(2,*)
         WRITE(*,*)
2230  CONTINUE

      READ(2,*)
      WRITE(*,*)

      READ(2,908) EPSILON
      WRITE(*,908) EPSILON

      READ(2,908) SMALL
      WRITE(*,908) SMALL

      DO 2240 I=1,2
         READ(2,*)
2240  CONTINUE

      DO 2250 I=1,NBASIC
         READ(2,909) IBASICI(I),IBASICK(I)
2250  CONTINUE

      CLOSE (UNIT=2)

901   FORMAT(T30,I2)
902   FORMAT(T30,D13.7D2)
905   FORMAT(T15,3D14.6D2)
906   FORMAT(T15,6D9.2)
907   FORMAT(T15,D14.6D2)
908   FORMAT(T30,D8.2D2)
909   FORMAT(T20,I2,T30,I2)
      RETURN
      END
```

c

```
      SUBROUTINE DLOHIT(ICONVEX,NLO,NHI)

c subroutine to determine if box region is convex

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "COMM.INC"

      REAL*8 NLO(CI,CK),NHI(CI,CK),DDFLO(CI,CI),DDFHI(CI,CI)

      CALL DLOHI(NLO,NHI,DDFLO,DDFHI)

      DO I=1,CI
         IF (DDFLO(I,I).LT.0) THEN
            ICONVEX=0
            RETURN
         ENDIF
      ENDDO

      CALL INTSCHUR(ICONVEX,DDFLO,DDFHI)

      IF (ICONVEX.EQ.1) THEN
            WRITE(*,*) "region is convex"
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE INTSCHUR(ICONVEX,XLO,XHI)

c intschur uses interval methods with a schur complement test
c to determine if the principle minors of all Hessians within
c an interval are convex or not.
c Interval is convex if convex=1 on return

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "COMM.INC"
      REAL*8 XLO(CI,CI),XHI(CI,CI),T(4)

      ICONVEX=1

      IF (XLO(1,1).LT.0) THEN
         ICONVEX = 0
         RETURN
      ENDIF

      DO 10 K=2,CI
         DO 20 J=K,CI
            DO 30 I=K,CI

               T(1)=XLO(I,K-1)*XLO(K-1,J)
               T(2)=XLO(I,K-1)*XHI(K-1,J)
               T(3)=XHI(I,K-1)*XLO(K-1,J)
               T(4)=XHI(I,K-1)*XHI(K-1,J)
               TMAX=DMAX1(T(1),T(2),T(3),T(4))
               TMIN=DMIN1(T(1),T(2),T(3),T(4))

               IF (TMAX.LT.0) THEN
                 XLO(I,J)=XLO(I,J)-TMAX/XLO(K-1,K-1)
               ELSE
                 XLO(I,J)=XLO(I,J)-TMAX/XHI(K-1,K-1)
               ENDIF

               IF (TMIN.LT.0.D0) THEN
                  XHI(I,J)=XHI(I,J)-TMIN/XLO(K-1,K-1)
               ELSE
                  XHI(I,J)=XHI(I,J)-TMIN/XHI(K-1,K-1)
               ENDIF

               XLO(1,K)=XLO(K,K)

               IF (XLO(K,K).LT.0) THEN
                  ICONVEX = 0
                  RETURN
               ENDIF

30          CONTINUE
20       CONTINUE
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DLOHI(NLO,NHI,DDFLO,DDFHI)

c dlohi calculates upper and lower limits on the interval
c reduced Hessian
c of the NRTL function for a 2 phase nonreacting system
c output f, df, ddf


      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      INCLUDE "COMM.INC"
      REAL*8 THETA(CI,CI),XLO(CI),XHI(CI),
     &       NLO(CI,CK),NHI(CI,CK),SUMMLO(CI),SUMMHI(CI),DGLO(CI,CI),
     &       DGHI(CI,CI),DDGLO(CI,CI*CI),DDGHI(CI,CI*CI),DDFLO(CI,CI),
     &       DDFHI(CI,CI),NFEED(CI)



      DO II=1,CI
         DO J=1,CI
            THETA(II,J)=GNRTL(II,J)*TAU(II,J)
         ENDDO
      ENDDO


      F=0
      DO K=1,2


         DO II=1,CI
            XLO(II)=NLO(II,K)
            XHI(II)=NHI(II,K)
         ENDDO



         DO J=1,CI
            SUMMLO(J)=0.0D0
            SUMMHI(J)=0.0D0
            DO II=1,CI
C              GNRTL is always positive

               SUMMLO(J)=SUMMLO(J)+XLO(II)*GNRTL(II,J)
               SUMMHI(J)=SUMMHI(J)+XHI(II)*GNRTL(II,J)

            ENDDO
         ENDDO



         DO I=1,CI
            DO L=1,CI

               IF (THETA(I,L).LT.0) THEN
                  DGLO(I,L)=( THETA(I,L)/SUMMLO(L) )
                  DGHI(I,L)=( THETA(I,L)/SUMMHI(L) )
               ELSE
                  DGLO(I,L)=( THETA(I,L)/SUMMHI(L) )
                  DGHI(I,L)=( THETA(I,L)/SUMMLO(L) )
               ENDIF
```



```
               DO J=1,CI

                  TEMP=-THETA(I,J)*GNRTL(L,J)
                  IF (TEMP.LT.0) THEN

                     DGLO(I,L)=
     &                  DGLO(I,L)+
     &                  THETA(I,J)*XHI(J)*(-GNRTL(L,J))/
     &                  (  ( SUMMLO(J)-XLO(J)+XHI(J) )*SUMMLO(J)  )

                    DGHI(I,L)=
     &                  DGHI(I,L)+
     &                  THETA(I,J)*XLO(J)*(-GNRTL(L,J))/
     &                  (  ( SUMMHI(J)-XHI(J)+XLO(J) )*SUMMHI(J)  )

                  ELSE

                     DGHI(I,L)=
     &                  DGHI(I,L)+
     &                  THETA(I,J)*XHI(J)*(-GNRTL(L,J))/
     &                  (  ( SUMMLO(J)-XLO(J)+XHI(J) )*SUMMLO(J)  )


                     DGLO(I,L)=
     &                  DGLO(I,L)+
     &                  THETA(I,J)*XLO(J)*(-GNRTL(L,J))/
     &                  (  ( SUMMHI(J)-XHI(J)+XLO(J) )*SUMMHI(J) )

                  ENDIF
               ENDDO
            ENDDO
         ENDDO



         DO I=1,CI
            DO M=1,CI
               DO L=1,CI
                  IF (L.LT.M) THEN !take symmetry of hessian into account

                     DDGLO(I,CI*(M-1)+L)=DDGLO(I,CI*(L-1)+M)
                     DDGHI(I,CI*(M-1)+L)=DDGHI(I,CI*(L-1)+M)

                  ELSE

                     DDGLO(I,CI*(M-1)+L)=0.0D0
                     DDGHI(I,CI*(M-1)+L)=0.0D0

                     DO J=1,CI

                        TEMP=THETA(I,J)*2*GNRTL(L,J)*GNRTL(M,J)

                        IF (TEMP.LT.0) THEN


                           DDGLO(I,CI*(M-1)+L)=DDGLO(I,CI*(M-1)+L)+
     &                        TEMP*XHI(J)/
     &                        ((SUMMLO(J)-XLO(J)+XHI(J))*SUMMLO(J)**2)
```

```
CV1                     TEMP*XHI(J)/(SUMMLO(J)**3)

                        DDGHI(I,CI*(M-1)+L)=DDGHI(I,CI*(M-1)+L)+
     &                  TEMP*XLO(J)/
     &                  ((SUMMHI(J)-XHI(J)+XLO(J))*SUMMHI(J)**2)

CV1                     TEMP*XLO(J)/(SUMMHI(J)**3)
                     ELSE

                        DDGHI(I,CI*(M-1)+L)=DDGHI(I,CI*(M-1)+L)+
     &                  TEMP*XHI(J)/
     &                  ((SUMMLO(J)-XLO(J)+XHI(J))*SUMMLO(J)**2)

                        DDGLO(I,CI*(M-1)+L)=DDGLO(I,CI*(M-1)+L)+
     &                  TEMP*XLO(J)/
     &                  ((SUMMHI(J)-XHI(J)+XLO(J))*SUMMHI(J)**2)

                     ENDIF

                  ENDDO


                  TEMP=-THETA(I,L)*GNRTL(M,L)
                  IF (TEMP.LT.0) THEN

                     DDGLO(I,CI*(M-1)+L)=
     &               DDGLO(I,CI*(M-1)+L)+
     &               TEMP/(SUMMLO(L)**2)

                     DDGHI(I,CI*(M-1)+L)=
     &               DDGHI(I,CI*(M-1)+L)+
     &               TEMP/(SUMMHI(L)**2)

                  ELSE

                     DDGHI(I,CI*(M-1)+L)=
     &               DDGHI(I,CI*(M-1)+L)+
     &               TEMP/(SUMMLO(L)**2)

                     DDGLO(I,CI*(M-1)+L)=
     &               DDGLO(I,CI*(M-1)+L)+
     &               TEMP/(SUMMHI(L)**2)

                  ENDIF


                  TEMP=-THETA(I,M)*GNRTL(L,M)
                  IF (TEMP.LT.0) THEN

                     DDGLO(I,CI*(M-1)+L)=
     &               DDGLO(I,CI*(M-1)+L)+
     &               TEMP/(SUMMLO(M)**2)

                     DDGHI(I,CI*(M-1)+L)=
     &               DDGHI(I,CI*(M-1)+L)+
     &               TEMP/(SUMMHI(M)**2)

                  ELSE
```



```
                     DDGHI(I,CI*(M-1)+L)=
     &               DDGHI(I,CI*(M-1)+L)+
     &               TEMP/(SUMMLO(M)**2)

                     DDGLO(I,CI*(M-1)+L)=
     &               DDGLO(I,CI*(M-1)+L)+
     &               TEMP/(SUMMHI(M)**2)

                  ENDIF

               ENDIF
            ENDDO
         ENDDO
      ENDDO


      IF (K.EQ.1) THEN
         SIGNN=+1
      ELSEIF (K.EQ.2) THEN
         SIGNN=-1
      ENDIF

c calculate the reduced Hessian matrix

      IF (K.EQ.1) THEN
         DO L=1,CI
            DO M=1,CI
               DDFLO(L,M)=0.0D0
               DDFHI(L,M)=0.0D0
            ENDDO
         ENDDO
      ENDIF

      XTOTLO=0
      XTOTHI=0

      DO I=1,CI
          XTOTLO=XTOTLO+XLO(I)
          XTOTHI=XTOTHI+XHI(I)
      ENDDO

      DO L=1,CI
         DO M=1,CI

            IF (L.EQ.M) THEN
               DDFLO(L,M)=DDFLO(L,M)+1/XHI(L)-1/(XTOTLO-XLO(L)+XHI(L))
               DDFHI(L,M)=DDFHI(L,M)+1/XLO(L)-1/(XTOTHI-XHI(L)+XLO(L))
            ELSE
               DDFLO(L,M)=DDFLO(L,M)              -1/XTOTLO
               DDFHI(L,M)=DDFHI(L,M)              -1/XTOTHI
            ENDIF

            DDFLO(L,M)=DDFLO(L,M)+DGLO(L,M)+DGLO(M,L)
            DDFHI(L,M)=DDFHI(L,M)+DGHI(L,M)+DGHI(M,L)


            DO J=1,CI

               TEMP=DDGLO(J,CI*(M-1)+L)
```

```
                IF (TEMP.LT.0) THEN
                   DDFLO(L,M)=DDFLO(L,M)+XHI(J)*DDGLO(J,CI*(M-1)+L)
                ELSE
                   DDFLO(L,M)=DDFLO(L,M)+XLO(J)*DDGLO(J,CI*(M-1)+L)
                ENDIF

                TEMP=DDGHI(J,CI*(M-1)+L)
                IF (TEMP.LT.0) THEN
                   DDFHI(L,M)=DDFHI(L,M)+XLO(J)*DDGHI(J,CI*(M-1)+L)
                ELSE
                   DDFHI(L,M)=DDFHI(L,M)+XHI(J)*DDGHI(J,CI*(M-1)+L)
                ENDIF

             ENDDO


          ENDDO
       ENDDO

ENDDO

RETURN
END
```

```
* MINOS reads "fort.3" to set up nonconvex problem
Begin
   Problem number                    1
   Minimize
   Jacobian                      Dense


*  Nonlinear objective variables = (no.chemicals)*(no.phases)
   Nonlinear objective variables     6


   Nonlinear jacobian variables      0
   Major iterations              50000
   Minor iterations                100
   Penalty parameter                10
   Feasibility tolerance       1.0D-14
   Function precision          1.0D-14
   Optimality tolerance        1.0D-14
   Hessian dimension                33
   Derivative level                  3
   Verify level                      0
   Summary file                     -6
   Summary frequency                 0
   Scale option                      1
   Iterations                     5000
   Debug level                       0
   Print level (jflxb)           00000
   Print frequency                   0
   Summary level                    -1
   Summary frequency                 0

   Linesearch tolerance            0.1
   Subspace tolerance              0.5
End
```

```
* MINOS reads "fort.4" to set up relaxed dual subproblem
Begin Relaxed_Dual
   Problem number                      2
   Minimize
   Jacobian                        Dense
   Nonlinear objective variables       0

*  Nonlin. Jac = (no.chemicals)*(no.phases)
   Nonlinear jacobian variables        6


   Major iterations               100000
   Minor iterations                  100
   Penalty parameter                  10
   Feasibility tolerance         1.0D-14
   Function precision            1.0D-14
   Optimality tolerance          1.0D-11
   Hessian dimension                  33
   Derivative level                    3
   Verify level                        0
   Summary file                       -6
   Summary frequency                   0
   Scale option                        1
   Iterations                      50000
   Debug level                         0
   Print level (jflxb)             00000
   Print frequency                     0
   Summary level                      -1
   Summary frequency                   0
   Linesearch tolerance              0.1
   Subspace tolerance                0.5
End Relaxed_Dual
```

```
      PARAMETER(MAXM=CE+MAXNOD+1,MAXN=CI*CK+1,MAXNB=MAXM+MAXN,
     &          MAXNE=MAXM*MAXN,NWCORE=5000,NNAME=1)
```

```
c      File "COMM.INC"

c this file contains global parameters used throughout the algorithm

       INTEGER CB,CI,CK,CE,CNB,LDA,MB,MAXNOD,NBASIC,NX

c******* P R O B L E M   S P E C I F I C   P A R A M E T E R S *********
c
c  CI                   Number of chemical types in the system
c  CK                   Number of phases postulated
c  CE                   Number of atomic species
c  CB=2**(CI*CK)        Systems with no gas phase
c  CB=2**(CI*(CK-1))    Systems with a gas phase


       PARAMETER(CI=2,CK=2,CE=2,MB=CE,CB=2**(CI*CK),
     &         CNB=(CI*CK-CE),
     &         MAXNOD=20000,MAXCUT=1)

c************************************************************************
       PARAMETER (MAXCI = 6,MAXCK = 4,MAXCE = 6)

       PARAMETER(NX=CI*CK+1)
       PARAMETER(LDA=MB,NBASIC=MB)
       PARAMETER(SCALEFAC=1)

c data type declarations for common variables

       DOUBLE PRECISION UREG,    LREG,     PU,    ML,
     &                  EPSILON,MUMEM,
     &                  TAU,     GNRTL,    GRT,      TEMP,
     &                  PRESSURE      ,    NTOT,
     &                  NMEM,    NNODMEM,  PSIMEM,   LAMMEM,
     &                  LBOX,    UBOX,     XLIN ,    ATOMS ,
     &                  ABASIC,  SMALL,    EPSILON,  EPSILON2,
     &                  FEATOL
       INTEGER   NODETOT, MAJITER, IP, IS, ISC, S, IRDUAL,
     &           NODEIT,  P, NODEL, NODEC, CPL, PL,B

       INTEGER IBASICI ,IBASICK,
     &         INBASICI,INBASICK,IFLAGCV

      INTEGER  IPVT


       COMMON /HISTORY/UREG(0:MAXNOD,CI,CK),LREG(0:MAXNOD,CI,CK),
     &                 PU,ML,NODETOT,NODESX,IRDUAL,
     &                 MAJITER,IP(MAXNOD),IS(MAXNOD),ISC,
     &                 S(2**(CI*CK),CI,CK)

       COMMON /SYSDATA/ TAU(MAXCI,MAXCI),GNRTL(MAXCI,MAXCI),
     &                  GRT(CI,CK),TEMP,PRESSURE,NTOT(MAXCI),
     &                  ATOMS(MAXCE,MAXCI,MAXCK),
     &                  ABASIC(LDA,NBASIC),
     &                  IPVT(NBASIC),
     &                  IBASICI(CE),IBASICK(CE),
     &                  INBASICI(CNB),INBASICK(CNB),IFLAGCV(CK)

       COMMON /MEMORY / NMEM(0:MAXNOD,CI,CK),NNODMEM(MAXNOD,CI,CK),
```



```
     &                  PSIMEM(MAXNOD,CI,CK),
     &                  LAMMEM(MAXNOD,CI,CK),MUMEM(0:MAXNOD),
     &                  NODEIT(0:MAXNOD),P(0:MAXNOD),NODEL(0:MAXNOD),
     &                  NODEC,MINITER,INFORM1

       COMMON /DUALPAR/ LBOX(2**(CI*CK),CI,CK),UBOX(2**(CI*CK),CI,CK),
     &                  XLIN(MAXNOD,CI,CI,CK),CPL,PL(MAXNOD),B

       COMMON /SMALLVAL/SMALL,EPSILON,EPSILON2,FEATOL

       COMMON /OUTCNRTL/IW1
```

Appendix C

Gautam & Seider and Walraven & van Rompay Phase Splitting FORTRAN Program

```
      PROGRAM SPLITGRID

c this main program loops through a series of points in the composition domain

      INCLUDE "walgaut.com"
      CHARACTER*72 DATAFILE
      CHARACTER*72 OUTFILEA,OUTFILEB,OUTFILEC,OUTFILED
      REAL*8 NTOTRQQ(MAXCI),XQQ(MAXCI,MAXCK),XUQQ(3),XLQQ(3)
      REAL CPU,CPUNEW,CPUOLD,ETIME,XTIME(2)
      INTEGER CKXOPT

      DATAFILE="/home/cliff/fortran/split/datafile/prop-bute-wate.dat"
      OUTFILEA ="/home/cliff/fortran/split/outfiles/prop-bute-wate.ts"
      CALL DATINIT(DATAFILE)
      OPEN (UNIT=8,FILE=OUTFILEA,STATUS="unknown")

c lowest value for X(1)
      XLQQ(1)=0.0D0
c highest value for X(1)
      XUQQ(1)=1.00D0

c lowest value for X(3)
      XLQQ(3)=0.0D0
c higest value for X(3)
      XUQQ(3)=1.00D0

      DELTAQQ=0.025

      NTOTRQQ(3)=XLQQ(3)
      DO 10 WHILE(NTOTRQQ(3).LE.XUQQ(3))
         REWIND (UNIT=7)
         REWIND (UNIT=10)

         NTOTRQQ(1)=XLQQ(1)

         IQ1FEAS=1
         IQ2FEAS=1
         DO 20 WHILE(IQ2FEAS.EQ.1.AND.NTOTRQQ(1).LE.XUQQ(1))
            NTOTRQQ(2)=1.0D0-NTOTRQQ(1)-NTOTRQQ(3)


            IQ1FEAS=1
            DO 40 I=1,3

               IF(NTOTRQQ(I).LT.1.0D0) THEN
                  IF(NTOTRQQ(I).LT.1.0D-5) THEN
                     IQ1FEAS=0
                  ENDIF
               ELSE
                  IQ1FEAS=0
                  IQ2FEAS=0
               ENDIF

40          CONTINUE

            IF (IQ1FEAS.EQ.1) THEN

               DO 30 I=1,3
                  NTOT(I)=NTOTRQQ(I)
                  XQQ(I,1)=0.0D0
```



```
                  XQQ(I,2)=0.0D0
30             CONTINUE

               CALL PHASESPLIT(XQQ,CKXOPT)
               WRITE(8,1004)  NTOTRQQ(1),NTOTRQQ(2),NTOTRQQ(3),
     &         XQQ(1,1),XQQ(2,1),XQQ(3,1),XQQ(1,2),XQQ(2,2),XQQ(3,2),
     &         XQQ(1,3),XQQ(2,3),XQQ(3,3),CKXOPT


            ENDIF
            NTOTRQQ(1)=NTOTRQQ(1)+DELTAQQ
20       CONTINUE
         NTOTRQQ(3)=NTOTRQQ(3)+DELTAQQ
10    CONTINUE

      CLOSE(UNIT=8)

1000  FORMAT(9(D10.4,2X))
1002  FORMAT(9(D12.6,2X))
1003  FORMAT(6(D12.6,2X))
1004  FORMAT(12(D12.6,2X),3I)
      END


c_____________________________________________________________________
      SUBROUTINE PHASESPLIT(Y2,CKXOPT)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      INCLUDE "walgaut.com"
      INCLUDE "MINOSPAR.INC"
      INCLUDE "actcom.com"


      CHARACTER*72 DATAFILE
      CHARACTER*72 OUTFILE


      REAL*8  ACTIV(MAXCI),   DACTIV(MAXCI,MAXCI),  BINACTIV(MAXCI),
     &        N(MAXCI,MAXCK), X(MAXCI),              XMOLES(MAXCI),
     &        ICOAL(MAXCK),   Y(MAXCI),              Y2(MAXCI,MAXCK),
     &        PHAS(MAXCK),    GSPLIT(MAXCK),         NOPT(MAXCI,MAXCK),
     &        DELF(MAXCI*MAXCK)

      REAL ELAPSTIME,DIFFTIME,DTIME,ETIME,XTIME(2)

      INTEGER FLAGA,      FLAGB,    NSPECIES,
     &        ID(MAXCI),  NCALC,    IDCALC(MAXCI),
     &        MODEDERIV,  CKXOPT,   ICBPAIR(0:MAXCI),
     &        IBPAIR(0:MAXCI,MAXCI)

      LOGICAL WALRAV

      COMMON /OPTIONS/WALRAV

c note DATA should be updated to include ICBPAIR and IBPAIR information.

      OUTFILE ="/home/cliff/fortran/split/outfiles/pro-but-wat.debug"
      OPEN (UNIT=7,FILE=OUTFILE,STATUS="unknown")
```

```
c get the elapsed time

      ELAPSTIME = ETIME(XTIME)

      WRITE(7,1008)  ELAPSTIME
      WRITE(7,1009)  XTIME(1)
      WRITE(7,1010)  XTIME(2)

1008  FORMAT ("Time elapsed:",2X,E12.4)
1009  FORMAT ("User time   :",2X,E12.4)
1010  FORMAT ("System time :",2X,E12.4)

      CKX=1
      CKXOPT=1
      GOLD=1.D10
      IDPHAS(1)=2
      DO 400 I=1,CI
        N(I,1)=NTOT(I)
        NOPT(I,1)=NTOT(I)
        Y(I)=NTOT(I)
400   CONTINUE

c calculate the Gibbs energy at the trivial solution

      NPROB=4
      CKT=1
      CALL OBJ1(CI,Y,GOLD,DELF,NSTATE,NPROB)

c determine which species form an unstable binary solution with each other
c ICBPAIR(I) indicates the number of unstable binary pairs that can be formed
c            with species I.
c IBPAIR(I,J) indicated the Jth species which together with species I can form
c             an unstable binary pair.

c pro-but-wat and ethanol-ethyl acetate-water settings

      ICBPAIR(1)=0
      ICBPAIR(2)=1
      IBPAIR(2,1)=3
      ICBPAIR(3)=1
      IBPAIR(3,1)=2

      WALRAV=1


      IF (WALRAV.EQ.1) THEN
         WRITE(7,*) "Walraven and Van Rompay Method"
      ELSE
         WRITE(7,*) "Gautam and Seider Method"
      ENDIF

      ICONVFLAG=0
      ITER=0
      DO 210 WHILE (ICONVFLAG.EQ.0)
      ITER=ITER+1


c calculate the total moles in each phase

      DO 340 K=1,CKX
```



```
         PHAS(K) = 0
         DO 345 I = 1,CI
            PHAS(K) = PHAS(K)+N(I,K)
345      CONTINUE
340   CONTINUE

c calculate the mole fractions in each phase

      DO 300 K=1,CKX
         DO 300 I=1,CI
            Y2(I,K) = N(I,K)/PHAS(K)
            Y(I)=Y2(I,K)
300   CONTINUE

c check for phase coalescence

      DO 310 K=1,CKX
         IF (ICOAL(K).EQ.0) THEN
            DO 350 K2=K+1,CKX
               IF ( PHAS(K2) .LT. 1.D-8 ) THEN
                  ICOAL(K2)=1
               ELSE
                  I=0
                  ISPLITFLAG=0

                  DO 360 WHILE ((I.LT.CI).AND.(IFLAG.EQ.0))
                     I=I+1
                     IF ( ABS(Y2(I,K2)-Y2(I,K)) .GT. 1.D-4) THEN
                        ISPLITFLAG=1
                     ENDIF
360                  CONTINUE

                  IF (ISPLITFLAG.EQ.0) THEN
                     ICOAL(K2)=1
                     DO 370 I=1,CI
                        N(I,K)=N(I,K)+N(I,K2)
                        N(I,K2)=0.D0
                        IF (KTEST.EQ.K2) KTEST=K
370                  CONTINUE
                  ENDIF
               ENDIF
350         CONTINUE
         ENDIF
310   CONTINUE

      DO 380 K=1,CKX
         IF (ICOAL(K).EQ.1) THEN
            CKX=CKX-1
            IF (KTEST.GT.K) KTEST=KTEST-1
            DO 390 K1=K,CKX
               ICOAL(K)=ICOAL(K+1)
               DO 390 I=1,CI
                  N(I,K1)=N(I,K1+1)
                  PHAS(K1)=PHAS(K1+1)
                  P(K1)=P(K1+1)
390         CONTINUE
         ENDIF
380   CONTINUE
```

```
c write the mole fractions of the current set of phases to a data file

      WRITE(7,*)
      WRITE(7,1005)
      DO 410 K=1,CKX
         DO 410 I=1,CI
            WRITE(7,1004) I,K,N(I,K)/PHAS(K)
410   CONTINUE
1004  FORMAT(I4,2X,I4,2X,D12.3)
1005  FORMAT("chem",1X,"phase",2X,"mole fraction")

c loop through all existing phases as source phases

      ISPLIT=0
      DO 230 ISOURCE=1,CKX
c        WRITE(*,*) "ISOURCE",ISOURCE

c calculate the activities of all species in the source phase

         DO 20 I=1,CI
            XMOLES(I)=N(I,ISOURCE)
            NSOURCE(I)=N(I,ISOURCE)
            ID(I)     =I
            IDCALC(I)=I
20       CONTINUE

         NSPECIES  =CI
         NCALC     =CI
         MODEDERIV =0

         CALL ACTIVITY(    XMOLES,  NSPECIES, ID,
     &                     NCALC,   IDCALC,   MODEDERIV,
     &                     ACTIV,   DACTIV              )

c test which species' activity is the largest

         TOPACTIV=-1.D10
         DO 30 I=1,CI

c do not allow a species that cannot form an unstable binary pair to
c be assigned to TOPACTIV

            IF (ICBPAIR(I).NE.0) THEN
               IF (ACTIV(I).GT.TOPACTIV) THEN
                  BINL(0)=I
                  TOPACTIV=ACTIV(I)
               ENDIF
            ENDIF
30       CONTINUE


c calculate the binary activities with BINL(0)

         NSPECIES  =2
         NCALC     =1
         MODEDERIV =0

         ICOUNT=0
         DO 40 I=1,CI
            IF (I.NE.BINL(0)) THEN
```



```
               ICOUNT=ICOUNT+1
               ID(1)=BINL(0)
               ID(2)=I
               IDCALC(1)=I

               CALL ACTIVITY(    XMOLES,  NSPECIES, ID,
     &                           NCALC,   IDCALC,   MODEDERIV,
     &                           ACTIV,   DACTIV              )


               BINACTIV(I)=ACTIV(I)

c insert the species into the list ranking the binary activities

               I2=0
               INSERTFLAG=0
               DO 50 WHILE(INSERTFLAG.EQ.0)
                  I2=I2+1
                  IF (I2.LT.ICOUNT) THEN
                     J=BINL(I2)
                     IF (BINACTIV(I).GT.BINACTIV(J)) THEN
                        DO 60 J2=ICOUNT,I2,-1
                           BINL(J2+1)=BINL(J2)
60                      CONTINUE

c note J2 must be set to I2 otherwise J2 = I2-1

                        J2=I2
                        BINL(J2)=I
                        INSERTFLAG=1
                     ELSE
                        INSERTFLAG=0
                     ENDIF
                  ELSE
                     BINL(ICOUNT)=I
                     INSERTFLAG=1
                  ENDIF
50             CONTINUE

            ENDIF
40       CONTINUE

         I3=0
         AFLAG=0
         DO 65 WHILE(AFLAG.EQ.0)
            I3=I3+1
            IF (ICBPAIR(BINL(I3)).GE.1) THEN
               DO 66 I4=1,ICBPAIR(BINL(I3))
                  IF (IBPAIR(BINL(I3),I4).EQ.BINL(0)) THEN
                     AFLAG=1
                     I5=BINL(I3)
                  ENDIF
66             CONTINUE
            ENDIF
65       CONTINUE


c move BINL(I3) to BINL(1) and BINL(I3-1) to BINL(I3) and so on.
```

```
      DO 67 I2=I3-1,1,-1
         BINL(I2+1)=BINL(I2)
67    CONTINUE
      BINL(1)=I5

      WRITE(7,*)
      WRITE(7,998)
      DO 45 I=0,CI-1
         WRITE(7,999)I,BINL(I),BINACTIV(BINL(I))
45    CONTINUE
998   FORMAT('Rank   Species   Binary Activity')
999   FORMAT(I4,3X,I4,6X,D20.5)

c trial phase loop

      FLAGA=0
      FLAGB=0

c detect if there is a vapour phase other than the source phase

      IF(IVAP.NE.1) FLAGA=1  ! if a vapour phase is to be excluded from the
                             ! algorithm set FLAGA to 0.

      IF (FLAGA.NE.1) THEN
         DO 150 K=1,CKX
            IF (K.NE.ISOURCE) THEN
               IF (IDPHAS(K).EQ.1) FLAGA=1
            ENDIF
150      CONTINUE
      ENDIF


      DO 100 WHILE (FLAGB.EQ.0)

c skip the vapour trial phase if this leads to two vapour phases
c in the system.

         IF (FLAGA.EQ.0) THEN
            ISPLIT=ISPLIT+1
            P(ISPLIT)=ISOURCE   ! remember the parent of the trial phases
            IDPHAST(ISPLIT,1)=1 ! vapour phase
            IDPHAST(ISPLIT,2)=2 ! NRTL liquid phase
            CALL DISTRIB(N,GSPLIT)
            FLAGA=1             ! don't set another vapour trial phase
            IF (IDPHAS(ISOURCE).EQ.1) FLAGB=1
         ELSE
            ISPLIT=ISPLIT+1
            P(ISPLIT)=ISOURCE   ! remember the parent of the trial phases
            IDPHAST(ISPLIT,1)=2 ! NRTL liquid phase 1
            IDPHAST(ISPLIT,2)=2 ! NRTL liquid phase 2
            CALL DISTRIB(N,GSPLIT)
            FLAGB=1             ! don't set another pair of trial phases
         ENDIF
100   CONTINUE

230   CONTINUE
      ICSPLIT=ISPLIT

c get the elapsed time
```



```
      ELAPSTIME = ETIME(XTIME)

      WRITE(7,*)
      WRITE(7,*) "End of the stability check and initial guess "
      WRITE(7,1008)  ELAPSTIME
      WRITE(7,1009)  XTIME(1)
      WRITE(7,1010)  XTIME(2)


c attempt to reduce Gibbs energy with trial phases replacing source

      ICOUNT = 0
      GNEW=GOLD+1
      DO 200 WHILE(GNEW.GT.GOLD.AND.ICOUNT.LT.ICSPLIT)
         ICOUNT=ICOUNT+1
         ISPLIT=ISPLITL(ICOUNT)

c if the source phase was found to be unstable perform a Gibbs energy
c minimization using the starting points from the phase splitting algorithm

         IF(GSPLIT(ISPLIT).LT.1.D+10) THEN

            ICOUNT1=0
            DO 110 K=1,CKX+1
               DO 110 I=1,CI
                  ICOUNT1=ICOUNT1+1
                  IF (K.EQ.P(ISPLIT)) THEN
                     X(ICOUNT1) = NTRIAL(ISPLIT,I,1)
                     IDPHAS(K)=IDPHAST(ISPLIT,1)
                  ELSEIF(K.LE.CKX) THEN
                     X(ICOUNT1) = N(I,K)
                  ELSEIF(K.EQ.CKX+1) THEN
                     X(ICOUNT1) = NTRIAL(ISPLIT,I,2)
                     IDPHAS(K)=IDPHAST(ISPLIT,2)
                  ENDIF
110         CONTINUE


            CKT=CKX+1
            IDPHAS(CKT)=IDPHAST(ISPLIT,2)

            write(7,*) x

            CALL RXN(X,GNEW)

c if the source.phase was found to be stable

         ELSE

            GNEW=1.D+10+1

         ENDIF

200   CONTINUE


      WRITE(7,*) "GNEW",GNEW
      WRITE(7,*) "GOLD",GOLD
```

```
      IF (GNEW.LT.GOLD-1.D-14) THEN

         ICONVFLAG=0
         CKX=CKX+1
         GOLD=GNEW
         CKXOPT=CKX

         ICOUNT1=0
         DO 120 K=1,CKX
            DO 120 I=1,CI
               ICOUNT1=ICOUNT1+1
               N(I,K)=X(ICOUNT1)
               NOPT(I,K)=N(I,K)
120      CONTINUE

      ELSE

         ICONVFLAG=1
         WRITE(7,*)
         WRITE(7,1003)
         WRITE(7,*)
         WRITE(7,1006)

         DO 225 K=1,CKXOPT
            PHAS(K)=0.D0
            DO 225 I=1,CI
               PHAS(K)=PHAS(K)+NOPT(I,K)
225      CONTINUE

         DO 235 K=1,CKXOPT
            DO 235 I=1,CI
               Y2(I,K)=NOPT(I,K)/PHAS(K)
               WRITE(7,1007) K,I,NOPT(I,K),Y2(I,K)
235      CONTINUE

      ENDIF

c get the elapsed time

      ELAPSTIME = ETIME(XTIME)
      WRITE(7,*)
      WRITE(7,*) "Times at end of the equilibrium calculation"
      WRITE(7,1008)  ELAPSTIME
      WRITE(7,1009)  XTIME(1)
      WRITE(7,1010)  XTIME(2)

210   CONTINUE

1001  FORMAT("PHASE ",I3)
1002  FORMAT(I3,D12.3)
1003  FORMAT("EQUILIBRIUM SYSTEM DETERMINED AS:")
1006  FORMAT("phase",2X,"chem",2X,"moles",9X,
     & "mole fraction in phase")
1007  FORMAT(I5,2X,I4,2X,D12.4,2X,D12.4)

      END

c-----------------------------------------------------------------------
      SUBROUTINE DISTRIB(N,GSPLIT)
```



```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"
      INCLUDE "actcom.com"

      REAL*8   X(MAXCI*MAXCK),  DELF(MAXCI*MAXCK),GSPLIT(MAXCK),
     &         N(MAXCI,MAXCK),  XN(5),            XBIN(2),
     &         XTOT(2),         XIT1(MAXCI),      XIT2(MAXCI),
     &         XT1(MAXCI),      XT2(MAXCI),       TXTOT(2),
     &         XS(MAXCI),       XS2(MAXCI),       TTRIAL(MAXCI,2)

      LOGICAL STABLE,WALRAV

      INTEGER IDS(2)

      COMMON /UNSTAB/  IDS
      COMMON /OPTIONS/ WALRAV

913   FORMAT("  I",1X,"BINL(I)",3X,"NSOURCE(BINL(I)",3X,
     & "XSOURCE(BINL(I)",3X,"NTRIAL(BINL(I),1)",3X,
     & "XTRIAL1(BINL(I)",3X,"NTRIAL(BINL(I),2)",3X,
     & "XTRIAL2(BINL(I)" )
914   FORMAT(I3,1X,I7,6(3X,D15.5))

      WRITE(7,*)
      WRITE(7,*) "Nature of trial phases: vapour=1, liquid=2 "
      WRITE(7,*) "Trial phase 1:",IDPHAST(ISPLIT,1 )
      WRITE(7,*) "Trial phase 2:",IDPHAST(ISPLIT,2 )

c determine the initial values for the trial phases using the
c isoactivity criterion.

      IPASSFLAG=0
      IPASS    =0

      DO 5 WHILE(IPASSFLAG.EQ.0)

         IF (IPASS.EQ.0) THEN
            WRITE(7,*)
            WRITE(7,*) "1st Gautam and Seider phase initialization"
         ELSE
            WRITE(7,*)
            WRITE(7,*) "2nd Gautam and Seider phase initialization"
         ENDIF

         IF (IPASS.EQ.0) IB1SUBS=1
         IF (IPASS.EQ.1) IB1SUBS=0

         DO 10 WHILE(IB1SUBS.LE.CI-1)

c call the routine that equalizes activities in the trial phases

            IF  (IB1SUBS.EQ.1.AND.IPASS.EQ.0) THEN

               IDS(1)=BINL(0)
               IDS(2)=BINL(1)
               CALL UNSTABL(XBIN,OBJ,INFBIN)
               XN(1)=XBIN(1)
               XN(2)=XBIN(2)

            ELSE
```

```
      IF (WALRAV.EQ.1) THEN      ! the Walraven option

c check if |T1m-Sm|<|IT1m-Sm| and |T2m-Sm|<|IT2m-Sm|

        DO 120 I=1,CI
           XIT1(I)=NTRIAL(ISPLIT,I,1)/XTOT(1)
           XIT2(I)=NTRIAL(ISPLIT,I,2)/XTOT(2)
           XT1(I)=TTRIAL(I,1)/TXTOT(1)
           XT2(I)=TTRIAL(I,2)/TXTOT(2)
           XS(I)=NSOURCE(I)/(XTOTS)
120     CONTINUE


9011    FORMAT(I3,5(D10.5,2X))
        R1= ( XT1(BINL(0))-XS(BINL(0)) )/
     &      ( XIT1(BINL(0))-XS(BINL(0)) )
        R2= ( XT2(BINL(0))-XS(BINL(0)) )/
     &      ( XIT2(BINL(0))-XS(BINL(0)) )

        IF (R1.GT.1.OR.R2.GT.1) THEN
           IF (R1.GT.1) THEN

              WRITE(7,*) "Adjust trial phase 1 "
              MINI   = 0
              MAXI   = 0
              XITMIN = 1.0
              XITMAX = 0.0

c readjust the mole fractions

              DO 130 I=1,CI
                 XIT1(I)=(1-R1)*XS(I)+XIT1(I)*R1

c record the minimum mole fraction

                 IF (XIT1(I).LT.XITMIN) THEN
                    MINI=I
                    XITMIN=XIT1(I)
                 ENDIF


130           CONTINUE

c if the smallest mole fraction is nearly zero or negative the
c moles must be adjusted so they are all positive

              IF (XITMIN.LT.1.D-9) THEN
                 WRITE(7,*) " Minimum mole fraction is less than 0 "
                 AQ=(XIT1(MINI)-1.D-9)/(XIT1(MINI)-XS(MINI))

                 DO 140 I=1,CI
                    XIT1(I)=AQ*XS(I)+(1-AQ)*XIT1(I)
140              CONTINUE

              ENDIF

              DO 142 I=1,CI

c record the maximum mole fraction
```



```
                 IF (XIT1(I).GT.XITMAX) THEN
                    MAXI=I
                    XITMAX=XIT1(I)
                 ENDIF
142           CONTINUE

              IF (XITMAX.GT.(1-1.D-9)) THEN
                 WRITE(7,*) "Maximum mole fraction exceeds 1"
                 AQ=(XIT1(MAXI)-(1-1.D-9))/(XIT1(MAXI)-XS(MAXI))

                 DO 145 I=1,CI
                    XIT1(I)=AQ*XS(I)+(1-AQ)*XIT1(I)
145              CONTINUE

              ENDIF

           ENDIF

c 2nd trial phase readjustment

          IF (R2.GT.1) THEN
             WRITE(7,*)
             WRITE(7,*) "Adjust trial phase 2"

             MINI   = 0
             MAXI   = 0
             XITMIN = 1.0
             XITMAX = 0.0

c Readjust the mole fractions

             DO 230 I=1,CI
                XIT2(I)=(1-R1)*XS(I)+XIT2(I)*R1

c record the minimum mole fraction

                IF (XIT2(I).LT.XITMIN) THEN
                   MINI=I
                   XITMIN=XIT2(I)
                ENDIF

230          CONTINUE

c if the smallest mole fraction is nearly zero or negative the
c moles must be adjusted so they are all positive

             IF (XITMIN.LT.1.D-9) THEN
                WRITE(7,*) "Minimum mole fraction is less than 0"
                AQ=(XIT2(MINI)-1.D-9)/(XIT2(MINI)-XS(MINI))
                DO 240 I=1,CI
                   XIT2(I)=AQ*XS(I)+(1-AQ)*XIT2(I)

c record the maximum mole fraction

                IF (XIT2(I).GT.XITMAX) THEN
                   MAXI=I
                   XITMAX=XIT2(I)
                ENDIF

240             CONTINUE
```

```
              ENDIF

c if the maximum mole fraction is greater than 1-eps readjust

              IF (XITMAX.GT.(1-1.D-9)) THEN
                 WRITE(7,*) " Maximum mole fraction exceeds 1"
                 AQ=(XIT2(MAXI)-(1-1.D-9))/(XIT2(MAXI)-XS(MAXI))
                 DO 245 I=1,CI
                    XIT2(I)=AQ*XS(I)+(1-AQ)*XIT2(I)
245              CONTINUE
              ENDIF

           ENDIF

           WRITE(7,9021)
           DO 290 I=1,CI
              WRITE(7,9020) I,XIT1(I),XIT2(I)
290        CONTINUE

        ENDIF

9009    FORMAT(6(D10.4,2X))

9010    FORMAT("XT2(1) ",2X,"XIT2(1) ",2X,"XT2(2) ",2X,"XIT2(2) ",2X
     &         "XT2(3) ",2X,"XIT2(3) ")

c prepare the input values for the Walraven stablity check and
c refinement of the initial guesses for the equilibrium calculation

        WRITE(7,*)
        WRITE(7,*) "Walraven and Van Rompay's initial guess refinement"

        CALL STABLCHECK(XIT1,XIT2,XS,STABLE)

        XTOT(1)=XTOTS*(XS(1)-XIT2(1))/
     &                (XIT1(1)-XIT2(1))

        XTOT(2)=XTOTS-XTOT(1)


        DO 80 I=1,CI
           NTRIAL(ISPLIT,I,1)=XIT1(I)*XTOT(1)
           NTRIAL(ISPLIT,I,2)=NSOURCE(I)-NTRIAL(ISPLIT,I,1)
80      CONTINUE

        WRITE(7,*)
        WRITE(7,9007)
        DO 90 I=1,CI
           WRITE(7,9008) I,NTRIAL(ISPLIT,I,1),NTRIAL(ISPLIT,I,2),
     &                   XIT1(I),XIT2(I)
90      CONTINUE

9005   FORMAT("XIT1",8X,"XIT2",8X,"XS")
9006   FORMAT(3(D10.4,2X))
9007   FORMAT("I",3X,"NTRIAL(I,1)",5X,"NTRIAL(I,2)",5X,
     &              "XTRIAL(I,1)",5X,"XTRIAL(I,2)")
9008   FORMAT(I2,2X,4(D10.4,6X))
9020   FORMAT(I2,2X,2(D10.4,6X))
9021   FORMAT("I",3X,"XTRIAL(I,1)",5X,"XTRIAL(I,2)")
1004   FORMAT(I4,I4,D12.3)
```



```
        CLOSE(UNIT=1)

c if the source phase has been found to be stable further computation
c is carried out on the trial system.

        WRITE(*,*) "STABLE",STABLE

        ELSE ! if the Walraven and Van Rompay refinements are not to be
             ! carried out.

        STABLE=0

        ENDIF

        IF (STABLE.EQ.0) THEN

c the inactive phases are phases CKX+1 to MAXCK
c trial phase 1 replaces the source phase in the tempory variable X
c trial phase 2 is put into phase CKX+1

           ICOUNT=0
           DO 100 K=1,CKX+1
              DO 100 I=1,CI
                ICOUNT=ICOUNT+1
                IF (K.EQ.ISOURCE) THEN
                   X(ICOUNT) = NTRIAL(ISPLIT,I,1)
                ELSEIF(K.LE.CKX) THEN
                   X(ICOUNT) = N(I,K)
                ELSEIF(K.EQ.CKX+1) THEN
                   X(ICOUNT) = NTRIAL(ISPLIT,I,2)
                ENDIF

100        CONTINUE


c calculate the Gibbs energy of the trial system.

           CKT=CKX+1
           NPROB=0
           CALL OBJ1(CI*CKT,X,F,DELF,NSTATE,NPROB)
           GSPLIT(ISPLIT) = F

        ELSE

           GSPLIT(ISPLIT)=1.D+10

        ENDIF


        WRITE(*,1001)
        WRITE(*,1002) ISPLIT,GSPLIT(ISPLIT)
1001    FORMAT("ISPLIT GSPLIT")
1002    FORMAT(I4,3X,D10.3)
1003    FORMAT(3(D12.4,1X))


c insert the split into a list according to increasing Gibbs energy

        I2=0
        INSERTFLAG=0
```

```
      DO 30 WHILE(INSERTFLAG.EQ.0)
         I2=I2+1
         IF (I2.LT.ISPLIT) THEN
            J=ISPLITL(I2)
            IF (GSPLIT(ISPLIT).LT.GSPLIT(J)) THEN
               DO 40 J2=ISPLIT,I2,-1
                  ISPLITL(J2+1)=ISPLITL(J2)
40             CONTINUE
               J2=I2
               ISPLITL(J2)=ISPLIT
               INSERTFLAG=1
            ELSE
               INSERTFLAG=0
            ENDIF
         ELSE
            ISPLITL(ISPLIT)=ISPLIT
            WRITE (*,*) "ISPLITL(ISPLIT)",ISPLITL(ISPLIT)
            INSERTFLAG=1
         ENDIF
30    CONTINUE


      WRITE(*,998)
      DO 50 I=1,ISPLIT
         WRITE(*,999)I,ISPLITL(I),GSPLIT(ISPLITL(I))
50    CONTINUE

      WRITE(*,*) "ISPLIT,IDPHAST(ISPLIT,1)",ISPLIT,IDPHAST(ISPLIT,1)
      WRITE(*,*) "ISPLIT,IDPHAST(ISPLIT,2)",ISPLIT,IDPHAST(ISPLIT,2)


998   FORMAT('I    ISPLITL(I)   Gibbs energy')
999   FORMAT(I4,3X,I4,6X,D20.5)


      RETURN
      END

c_____________________________________________________________
c     Objective and obj gradients for the nonconvex problem.

      SUBROUTINE OBJ1(NN,X,F,DELF,NSTATE,NPROB)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"
      INCLUDE "actcom.com"
      DOUBLE PRECISION X(NN),DELF(NN),NPHAS(CK),TPSI(CI,CK),
     &                 PSI(CI,CK),N(CI,CK)
       TINY=1.D-2*SMALL
       IX2N=0

       DO 10 K=1,CKT
          DO 10 I=1,CI
            IX2N=IX2N+1
            N(I,K)=X(IX2N)
10    CONTINUE

      DO 20 K=1,CKT
         NPHAS(K)=0.D0
```



```
         IF (NPROB.EQ.0) THEN
            IF (K.EQ.ISOURCE) THEN
               IDPHASTEMP=IDPHAST(ISPLIT,1)
            ELSEIF (K.EQ.CKX+1) THEN
               IDPHASTEMP=IDPHAST(ISPLIT,2)
            ENDIF
         ELSEIF (NPROB.EQ.4) THEN
            IDPHASTEMP=2
         ELSE
            IDPHASTEMP=IDPHAS(K)
         ENDIF



         DO 20 I=1,CI
            NPHAS(K)=NPHAS(K)+N(I,K)
            TPSI(I,K)=0.D0
            IF ( IDPHASTEMP.EQ.2) THEN
               DO 40 J=1,CI
                  TPSI(I,K)=TPSI(I,K)+GNRTL(J,I)*N(J,K)
40             CONTINUE
               PSI(I,K)= N(I,K)/TPSI(I,K)
            ENDIF

20    CONTINUE


      F=0
      ICOUNT=0
      DO 50 K=1,CKT
         DO 50 I=1,CI
            ICOUNT=ICOUNT+1

            TPCUR=0.D0

c Note phase 1 is an ideal vapour phase and this section applies only to
c the liquid phase.

            IF (NPROB.EQ.0) THEN
               IF (K.EQ.ISOURCE) THEN
                  IDPHASTEMP=IDPHAST(ISPLIT,1)
               ELSEIF (K.EQ.CKX+1) THEN
                  IDPHASTEMP=IDPHAST(ISPLIT,2)
               ENDIF
            ELSE
               IDPHASTEMP=IDPHAS(K)
            ENDIF

            IF (IDPHASTEMP.EQ.2) THEN

                DO 30 J=1,CI
                   TPCUR=TPCUR+GNRTL(I,J)*TAU(I,J)*PSI(J,K)
30              CONTINUE
            ENDIF

            F=F+N(I,K)*(TPCUR+
     &          ( GRT(I,K)+DLOG(N(I,K))-DLOG(NPHAS(K)) ))

c  determine gradients only if this function is being called from the
c  NLP solver.
```

```
            IF (NPROB.NE.0) THEN
               DELF(ICOUNT)=(TPCUR+
     &              ( GRT(I,K)+DLOG(N(I,K))-DLOG(NPHAS(K)) ))

               IF (IDPHAS(K).EQ.2) THEN
                  DO 60 K1=1,CKT
                     DO 60 I1=1,CI
                        TEMP=0.D0
                        DO 70 J=1,CI
                           IF (K1.EQ.K) THEN
                              TEMP=TEMP-GNRTL(I1,J)*TAU(I1,J)*
     &                             GNRTL(I,J)*N(J,K1)/(TPSI(J,K1)**2)
                              IF (J.EQ.I) THEN
                                 TEMP = TEMP + GNRTL(I1,J)*TAU(I1,J)
     &                                /TPSI(J,K)
                              ENDIF
                           ENDIF
70                         CONTINUE
                           DELF(ICOUNT)=DELF(ICOUNT)+N(I1,K1)*TEMP
60                      CONTINUE
                     ENDIF
                  ENDIF



50    CONTINUE
      RETURN
      END
*-----------------------------------------------------------------------
c subroutine used to set up MINOSS for the nonconvex problem.
      SUBROUTINE RXN(XN,OBJ)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      INCLUDE "walgaut.com"
      INCLUDE "MINOSPAR.INC"
      INCLUDE "actcom.com"

c data type declarations for MINOS variables

      DOUBLE PRECISION  A(MAXNE),BL(MAXNB),BU(MAXNB),
     &                  XN(MAXNB),PI(MAXM),RC(MAXNB),Z(NWCORE)

      CHARACTER*8 NAMES(5)
      INTEGER*4   HA(MAXNE),HS(MAXNB)
      INTEGER     KA(MAXN+1),NAME1(NNAME),NAME2(NNAME)

      NPROB=1
      NOBJ=CI*CKT
      NJAC=0
      CALL SPECWRITER(NPROB,NOBJ,NJAC)
      ISPECS=3
      IPRINT=10
      ISUMM =6
      NOUT  =6
      BPLUS =1.D+10
      BMINUS=-BPLUS
      SMALL =1.D-08
```



```
      M=CE
      NN=CI*CKT
      NB=NN+M
      NE=NN*M
      IOBJ = 0
      NNCON=0
      NNOBJ=CI*CKT
      NNJAC=0

*-----------------------------------------------------------------------
*     set the values in the constraint matrix
*
*             1            2   ....                      nn
*
*        1   ATOMS(1,1,1)  ATOMS(1,2,1)  ATOMS(1,1,2)  ATOMS(1,2,2)
*
*        .   ATOMS(2,1,1)
*
*       CE   ATOMS(CE,CI,1)                            ATOMS(CE,CI,CKX)
*
*
*-----------------------------------------------------------------------
c clean up A

      DO 10 I=1,NE
         A(I)=0
10    CONTINUE


c mass balance constraints

      ICOUNT=0
      DO 20 K=1,CKT
         DO 20 I=1,CI
            DO 20 J=1,CE
               ICOUNT=ICOUNT+1
               A(ICOUNT)=ATOMS(J,I,K)
               HA(ICOUNT)=J
20    CONTINUE

      DO 30 I=1,NN+1
         KA(I)=(I-1)*M+1
30    CONTINUE

c Bounds on mole variables

      ICOUNT=0
      DO 40 K=1,CKT
         DO 40 I=1,CI
            ICOUNT=ICOUNT+1
            BL(ICOUNT)=SMALL
            BU(ICOUNT)=BPLUS
40    CONTINUE

c Bounds on the slacks for the mass balance constraints

      DO 50 I=1,CE
         BL(NN+I)=-NTOT(I)
         BU(NN+I)=-NTOT(I)
```

```
50    CONTINUE


      DO 60 I=1,M
         PI(I)=0
60    CONTINUE

      DO 70 I=1,NB
        HS(I)=0
70    CONTINUE

      DO 80 I=1,MAXNB
         RC(I)=0
80    CONTINUE


      INFORM=0
      MINCORE=0
      NS=0
      NINF=0
      SINF=0
      OBJ=0
      OBJADD=0

      CALL MISPEC(ISPECS,IPRINT,ISUMM,NWCORE,INFORM)
      CALL MINOSS('COLD',M,NN,NB,NE,NNAME,
     &               NNCON,NNOBJ,NNJAC,
     &               IOBJ,OBJADD,NAMES,
     &               A,HA,KA,BL,BU,NAME1,NAME2,
     &               HS,XN,PI,RC,
     &               INFORM,MINCORE,NS,NINF,SINF,OBJ,
     &               Z,NWCORE)

      OBJ=OBJ-OBJADD
      INFORM1=INFORM
      DO 100 I=1,NN
         WRITE(*,*) "X",I,XN(I)
100   CONTINUE


      RETURN
      END


c_____________________________________________________________________
c subroutine used to set up MINOSS for the isoactivity problem.

      SUBROUTINE ISOACTIV2(XN,OBJ)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      INCLUDE "walgaut.com"
      INCLUDE "actcom.com"

      PARAMETER( MAXM=3,           MAXN=2,       MAXNB=MAXM+MAXN,
     &           MAXNE=MAXM*MAXN,  NWCORE=1000,  NNAME=1)

c data type declarations for MINOS variables

      REAL*8     A(MAXNE),     BL(MAXNB),     BU(MAXNB),
     &           XN(MAXNB),    PI(MAXM),      RC(MAXNB),
```



```
     &           Z(NWCORE)

      CHARACTER*8 NAMES(5)

      INTEGER*4   HA(MAXNE),   HS(MAXNB)

      INTEGER     KA(MAXN+1),  NAME1(NNAME), NAME2(NNAME)

      NPROB=2
      IF (IB1SUBS.EQ.1.AND.IPASS.EQ.0) THEN
         NOBJ=2
         NJAC=0
         NNOBJ=2      ! Number of nonlinear objective variables
         NNJAC=0      ! Number of nonlinear Jacobian variables
         NNCON=0      ! Number of nonlinear constraints
         NN=2         ! Number of variables (excluding slacks)
         M=1          ! Linear constraints + Nonlinear constraints + Linear Obj
         IOBJ=0       ! Linear part of objective function
         NE=1         ! Number of nonzero entries in A
         A(1)=1
         HA(1)=1
         KA(1)=1
         KA(2)=2
      ELSE
         NOBJ=1
         NJAC=0
         NNOBJ=1
         NNJAC=0
         NNCON=0
         NN=1
         M=1
         IOBJ=0
         NE=1
         A(1)=1
         HA(1)=1
         KA(1)=1
         KA(2)=2
      ENDIF
      CALL SPECWRITER(NPROB,NOBJ,NJAC)
      ISPECS=3
      IPRINT=10
      ISUMM =6
      NOUT  =6
      BPLUS =1.D+10
      BMINUS=-BPLUS

      NB=NN+M        ! Number of bounds (in BL or BU)


c Bounds on mole variables

      ICOUNT=0

      DO 70 K=1,NN
         BL(K)=0.D0
70    CONTINUE

      IF (IB1SUBS.EQ.1.AND.IPASS.EQ.0) THEN
         BL(1)=1.D-8
         BU(1)=NSOURCE(BINL(0))
```

```
         BL(2)=1.D-8
         BU(2)=NSOURCE(BINL(1))
         BL(3)=BMINUS
         BU(3)=BPLUS
         XN(1)=0.1*NSOURCE(BINL(0))
         XN(2)=0.9*NSOURCE(BINL(1))
      ELSE
         BL(1)=1.D-8
         BU(1)=NSOURCE(BINL(IB1SUBS))
         BL(2)=BMINUS
         BU(2)=BPLUS
         IF (IPASS.EQ.0) THEN
            XN(1)=0.5*NSOURCE(BINL(IB1SUBS))
         ELSE
            XN(1)=NTRIAL(ISPLIT,BINL(IB1SUBS),1)
         ENDIF
      ENDIF

      DO 25 I=1,M
         PI(I)=0
25    CONTINUE

      DO 150 I=1,NB
        HS(I)=0
150   CONTINUE

      DO 210 I=1,5
         RC(I)=0
210   CONTINUE

      INFORM=0
      MINCORE=0
      NS=0
      OBJ=0
      OBJADD=0

      CALL MISPEC(ISPECS,IPRINT,ISUMM,NWCORE,INFORM)
      CALL MINOSS('COLD',M,NN,NB,NE,NNAME,
     &            NNCON,NNOBJ,NNJAC,
     &            IOBJ,OBJADD,NAMES,
     &            A,HA,KA,BL,BU,NAME1,NAME2,
     &            HS,XN,PI,RC,
     &            INFORM,MINCORE,NS,NINF,SINF,OBJ,
     &            Z,NWCORE)

      OBJ=OBJ-OBJADD
      INFORM1=INFORM

      WRITE(7,*)
      WRITE(7,*) 'Isoactivity objective=',OBJ
      RETURN
      END

c-----------------------------------------------------------------------

      SUBROUTINE FUNOBJ(MODE,NN,X,F,DELF,NSTATE,NPROB,Z,NWCORE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"
      DOUBLE PRECISION X(NN),DELF(NN),Z(NWCORE)
```



```
      COMMON  /M1FILE/IREAD,IPRINT,ISUMM

      IF (NPROB.EQ.1) CALL OBJ1(NN,X,F,DELF,NSTATE,NPROB)
      IF (NPROB.EQ.2) CALL OBJ2(NN,X,F,DELF,NSTATE,NPROB)

c this one is for Walraven's algorithm only
      IF (NPROB.EQ.3) CALL OBJ3(NN,X,F,DELF,NSTATE,NPROB)

      IF (NPROB.EQ.5) CALL OBJ5(NN,X,F,DELF,NSTATE,NPROB)
      IF (NPROB.EQ.6) CALL NEWOBJ1(NN,X,F,DELF,NSTATE,NPROB)

      RETURN
      END


c-----------------------------------------------------------------------

      SUBROUTINE FUNCON(  MODEX,M,NN,NJAC,X,G,DELG,NSTATE,
     &                    NPROB,Z,NWCORE          )

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"
      INCLUDE "actcom.com"

      REAL*8   X(NN),     G(M),     DELG(M,NN),
     &         Z(NWCORE)

      RETURN
      END
c-----------------------------------------------------------------------

      SUBROUTINE OBJ2(NN,X,F,DELF,NSTATE,NPROB)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"
      INCLUDE "actcom.com"

      REAL*8   FT(2),      DFT(2,2),     XT(2),
     &         X(NN),      F,            DELF(NN)

      DO 10 I=1,NN
         XT(I)=X(I)
10    CONTINUE

      MODE2=3
      CALL ACTOBJECT(NN,XT,FT,DFT,MODE2)

      IF (NN.EQ.1) F=FT(1)**2
      IF (NN.EQ.2) F=FT(1)**2+FT(2)**2
      IF (IDEBUG.EQ.1) THEN
         WRITE(*,*) "Isoactivity objective: ",F
      ENDIF

      RETURN
      END

c-----------------------------------------------------------------------
      SUBROUTINE SPECWRITER(NPROB,NOBJ,NJAC)
      ISPEC=3
      OPEN (UNIT=ISPEC,FILE="fort.3",STATUS="unknown")
      REWIND ISPEC
```

```
      WRITE(ISPEC,*)    "   Begin Nonconvex                          "
      WRITE(ISPEC,*)    "   Problem number                    ",NPROB
      WRITE(ISPEC,*)    "   Minimize                                 "
      WRITE(ISPEC,*)    "   Jacobian                        Dense    "
      WRITE(ISPEC,*)    "   Nonlinear objective variables     ",NOBJ
      WRITE(ISPEC,*)    "   Nonlinear jacobian variables      ",NJAC
      WRITE(ISPEC,*)    "   Major iterations                50000    "
      WRITE(ISPEC,*)    "   Minor iterations                  100    "
      WRITE(ISPEC,*)    "   Penalty parameter                  10    "
      WRITE(ISPEC,*)    "   Feasibility tolerance          1.0D-14    "
      WRITE(ISPEC,*)    "   Function precision             1.0D-14    "
      WRITE(ISPEC,*)    "   Optimality tolerance           1.0D-14    "
*     WRITE(ISPEC,*)    "*   LU Factor tolerance                2    "
*     WRITE(ISPEC,*)    "*   LU Update tolerance                2    "
      WRITE(ISPEC,*)    "   Hessian dimension                 33    "
      WRITE(ISPEC,*)    "   Derivative level                   0    "
*     WRITE(ISPEC,*)    "*   Verify gradients                      "
      WRITE(ISPEC,*)    "   Verify level                       1    "
      WRITE(ISPEC,*)    "   Summary file                      -6    "
      WRITE(ISPEC,*)    "   Summary frequency                  0    "
      WRITE(ISPEC,*)    "   Scale option                       1    "
*     WRITE(ISPEC,*)    "*  Scale tolerance                  0.9    "
      WRITE(ISPEC,*)    "   Iterations                      5000    "
      WRITE(ISPEC,*)    "   Debug level                        0    "
      WRITE(ISPEC,*)    "   Print level (jflxb)            00000    "
      WRITE(ISPEC,*)    "   Print frequency                    0    "
      WRITE(ISPEC,*)    "   Summary level                     -1    "
      WRITE(ISPEC,*)    "   Summary frequency                  0    "
      WRITE(ISPEC,*)    "   Linesearch tolerance            0.00001"
*     WRITE(ISPEC,*)    "*  Linesearch debug after iteration  30    "
      WRITE(ISPEC,*)    "   Subspace tolerance                 0.5  "
      WRITE(ISPEC,*)    "End Relaxed_Dual                           "

      CLOSE(UNIT=3)
10    CONTINUE

      RETURN
      END
c_____________________________________________________________________
      SUBROUTINE MATMOD
      RETURN
      END
```

```
      SUBROUTINE ACTIVITY( XMOLES, NSPECIES, ID,
     &                     NCALC,   IDCALC, MODEDERIV,
     &                     ACTIV,   DACTIV             )

c XMOLES     molar values of species in system, only those species
c            corresponding to ID need be specified.
c NSPECIES   the number of species in the system for which the  activity
c            is to be calculated.
c ID         ID(i) i=1,..,NSPECIES identifies the species to be included
c            in the system
c NCALC      the number of activities to be calculated
c IDCALC     the identities of the species for which the activities are
c            to be calculated
c MODEDERIV  1 if derivatives are to be calculated
c            0 otherwise
c ACTIV      activity coefficients
c DACTIV     derivatives of activity coefficients

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"
      INCLUDE "actcom.com"

      INTEGER NSPECIES,       ID(MAXCI),      NCALC,
     &        IDCALC(MAXCI),  MODEDERIV

      REAL*8  PHI1(MAXCI),    PHI2(MAXCI),    PHI3(MAXCI),
     &        ACTIV(MAXCI),   Y(MAXCI),       XMOLES(MAXCI),
     &        DACTIV(MAXCI,MAXCI)



92    FORMAT("    I ID(I) XMOLES(ID(I)) IDCALC(I) MODEDERIV  NCALC")
91    FORMAT(2I5,D15.5,3I10)

c calculate the values of Y

      TOT=0.D0
      DO 200 I=1,NSPECIES
         I2=ID(I)
         TOT=TOT+XMOLES(I2)
200   CONTINUE
      DO 240 I=1,NSPECIES
         I2=ID(I)
         Y(I2)=XMOLES(I2)/TOT
240   CONTINUE


c for the NRTL phases:

      DO 20 I=1,NSPECIES
         I2=ID(I)
         PHI1(I2)=0.D0
         PHI2(I2)=0.D0
         DO 10 J=1,NSPECIES
            J2=ID(J)
               PHI1(I2)=PHI1(I2)+TAU(J2,I2)*GNRTL(J2,I2)*Y(J2)
               PHI2(I2)=PHI2(I2)+GNRTL(J2,I2)*Y(J2)
10       CONTINUE
20    CONTINUE
```



```
c identify the species that the activity is to be calculated for
      DO 90 I=1,NCALC
         ICALC=IDCALC(I)
         PHI3(ICALC)=0.D0
         DO 30 J=1,NSPECIES
            J2=ID(J)
            PHI3(ICALC)=PHI3(ICALC)+GNRTL(ICALC,J2)*Y(J2)/PHI2(J2)*
     &      (TAU(ICALC,J2)-PHI1(J2)/PHI2(J2))
30       CONTINUE

         ACTIV(ICALC)=PHI1(ICALC)/PHI2(ICALC)+PHI3(ICALC)
         ACTIV(ICALC)=DEXP(ACTIV(ICALC))

90    CONTINUE

c note that the derivatives are with respect to the mole fractions
c and not mole numbers.

      IF (MODEDERIV.EQ.1) THEN

         DO 110 II=1,NCALC
            I=IDCALC(II)
            DO 110 JJ=1,NSPECIES
               J=ID(JJ)

               DPHI3=0.0D0

               DO 100 J2=1,NSPECIES
                L=ID(J2)
                DPHI3=DPHI3
     &          -GNRTL(I,L)*Y(L)/(PHI2(L)**2)
     &          *GNRTL(J,L)*(TAU(I,L)-PHI1(L)/PHI2(L))
     &          +GNRTL(I,L)*Y(L)/PHI2(L)*(-TAU(J,L)*GNRTL(J,L)/PHI2(L)
     &          + GNRTL(J,L)*PHI1(L)/PHI2(L)**2    )
100            CONTINUE

               DPHI3=DPHI3+
     &         GNRTL(I,J)/PHI2(J)*(TAU(I,J)-PHI1(J)/PHI2(J))

               DACTIV(I,J)=1/PHI2(I)*(TAU(J,I)*GNRTL(J,I))
     &               -PHI1(I)/(PHI2(I)**2)*GNRTL(J,I)
     &               +DPHI3
               DACTIV(I,J)=DACTIV(I,J)*ACTIV(I)

110      CONTINUE

      ENDIF
99    FORMAT("DACTIV",2I3,D15.5)

      RETURN
      END
c_____________________________________________________________________
      SUBROUTINE ACTOBJECT(N, X, F, DF, MODE2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"
      INCLUDE "actcom.com"

      REAL*8   XNTOT(2),       X(2),          F(2),
```

```
     &          DF(2,2),        Z(2,2),        PX(2,2),
     &          ACTIV(MAXCI),   XMOLES(MAXCI),
     &          DACTIV(MAXCI,MAXCI)

       INTEGER  NSPECIES,       ID(MAXCI),     NCALC,
     &          IDCALC(MAXCI),  MODEDERIV

       LOGICAL  IPRINT

c only the species that have previously been equilibriated are included
c with the current species.
c There are two different problems to be set up:
c
c 1. The species with the highest activity is equilibriated with another
c    species which has a high binary activity with the former.  This second
c    species is usually that which has the highest binary activity.
c    In cases where this species is already present in large quantities
c    in a liquid phase the second species may be a one of lower binary
c    activity.  In either case the list BINL will have been adjusted to
c    take this into account so that species involved are BINL(0) and
c    BINL(1).
c
c 2. A species on the list BINL(i) where i>=2 is distributed between two
c    phases already containing fixed amounts of species BINL(j) ,j=0,1,..,i
c
c 3. When the subroutine is called from Walraven's adaptation of the
c    algorithm and IPASS = 1 the calculation involves fixed amounts of
c    all species besides the one being adjusted.
c
c XPHAS : the moles of the species of interest in trial phases 1 and 2
c Z(species,trial phase)


901    FORMAT("IB1SUBS      ",I3)
902    FORMAT("IPASS        ",I3)
903    FORMAT("N            ",I3)
904    FORMAT("X(1)         ",D15.5)
905    FORMAT("X(2)         ",D15.5)
906    FORMAT("F(1)         ",D15.5)
907    FORMAT("F(2)         ",D15.5)
908    FORMAT("DF(1,1)      ",D15.5)
909    FORMAT("DF(1,2)      ",D15.5)
910    FORMAT("DF(2,1)      ",D15.5)
911    FORMAT("DF(2,2)      ",D15.5)
912    FORMAT("MODE2        ",I3)
913    FORMAT("I",3X,"BINL(I)",4X,"NSOURCE(BINL(I)",4X,
     & "NTRIAL(BINL(I),1)",4X,"NTRIAL(BINL(I),2)")
914    FORMAT(I3,1X,I7,3X,D15.5,3X,D15.5,3X,D15.5)
915    FORMAT("IDPHAST(ISPLIT, )",2I3)
916    FORMAT("ISPLIT",I3)
917    FORMAT("PX",4D15.5)
918    FORMAT("Z",4D15.5)
919    FORMAT("XNTOT",2D15.5)

       MODE=3

       IF (IB1SUBS.EQ.1.AND.IPASS.EQ.0) THEN

          Z(1,1)=X(1)
          Z(1,2)=NSOURCE(BINL(0))-X(1)
```



```
          Z(2,1)=X(2)
          Z(2,2)=NSOURCE(BINL(1))-X(2)

          XNTOT(1)=Z(1,1)+Z(2,1)
          XNTOT(2)=Z(1,2)+Z(2,2)

       ELSE

          Z(1,1)=X(1)
          Z(1,2)=NSOURCE(BINL(IB1SUBS))-X(1)

          XNTOT(1)=0.D0
          XNTOT(2)=0.D0

          IF (IPASS.EQ.0) THEN
             J=IB1SUBS
          ELSE
             J=CI
          ENDIF

          DO 50 I=0,J-1
             XNTOT(1)=XNTOT(1)
     &              +NTRIAL(ISPLIT,BINL(I),1)
             XNTOT(2)=XNTOT(2)
     &              +NTRIAL(ISPLIT,BINL(I),2)
50        CONTINUE

          XNTOT(1)=XNTOT(1)+Z(1,1)
          XNTOT(2)=XNTOT(2)+Z(1,2)

       ENDIF


c case1: trial phase 1 - vapour; trial phase 2 - liquid NRTL

       IF(IDPHAST(ISPLIT,1).EQ.1) THEN

c determine the liquid phase activity

          IF (IB1SUBS.NE.1) THEN
             XMOLES(BINL(IB1SUBS))=Z(1,2)
             DO 10 I=1,CI
                IF (BINL(IB1SUBS).NE.I) THEN
                   XMOLES(I)=NTRIAL(ISPLIT,I,2)
                ENDIF
10           CONTINUE

             DO 100 J=0,IB1SUBS
                ID(J+1)=BINL(J)
100          CONTINUE
             NCALC     =1
             IDCALC(1) =BINL(IB1SUBS)
             NSPECIES  =IB1SUBS+1
             MODEDERIV =0

             CALL ACTIVITY( XMOLES, NSPECIES, ID,
     &                      NCALC,   IDCALC, MODEDERIV,
     &                      ACTIV,   DACTIV             )

             PX(1,1)=PRESSURE
```

```
            PX(1,2)=ACTIV(BINL(IB1SUBS))*PSAT(BINL(IB1SUBS))
         ELSE
            XMOLES(BINL(0))=Z(1,2)
            XMOLES(BINL(1))=Z(2,2)
            NSPECIES=2
            ID(1)=BINL(0)
            ID(2)=BINL(1)
            NCALC=2
            IDCALC(1)=BINL(0)
            IDCALC(2)=BINL(1)
            MODEDERIV=0
            CALL ACTIVITY( XMOLES, NSPECIES, ID,
     &                     NCALC,   IDCALC, MODEDERIV,
     &                     ACTIV,   DACTIV             )
            PX(1,1)=PRESSURE
            PX(2,1)=PRESSURE
            PX(1,2)=ACTIV(BINL(0))*PSAT(BINL(0))
            PX(2,2)=ACTIV(BINL(1))*PSAT(BINL(1))
         ENDIF

      ELSEIF(IDPHAST(ISPLIT,1).EQ.2) THEN

c case 2: trial phase 1 - liquid NRTL; trial phase 2 - vapour

         IF (IDPHAST(ISPLIT,2).EQ.1) THEN

            IF (IB1SUBS.NE.1) THEN
               XMOLES(BINL(IB1SUBS))=Z(1,1)
               DO 20 I=1,CI
                  IF (BINL(IB1SUBS).NE.I) THEN
                     XMOLES(I)=NTRIAL(ISPLIT,I,1)
                  ENDIF
20             CONTINUE

               DO 110 J=0,IB1SUBS
                  ID(J+1)=BINL(J)
110            CONTINUE
               NCALC     =1
               IDCALC(1) =BINL(IB1SUBS)
               NSPECIES  =IB1SUBS+1
               MODEDERIV =0

               CALL ACTIVITY( XMOLES, NSPECIES, ID,
     &                        NCALC,   IDCALC, MODEDERIV,
     &                        ACTIV,   DACTIV             )

               PX(1,1)=ACTIV(BINL(IB1SUBS))*PSAT(IB1SUBS)
               PX(1,2)=PRESSURE
            ELSE
               XMOLES(BINL(0))=Z(1,1)
               XMOLES(BINL(1))=Z(2,1)
               NSPECIES=2
               ID(1)=BINL(0)
               ID(2)=BINL(1)
               NCALC=2
               IDCALC(1)=BINL(0)
               IDCALC(2)=BINL(1)
               MODEDERIV=0
               CALL ACTIVITY( XMOLES, NSPECIES, ID,
     &                        NCALC,   IDCALC, MODEDERIV,
```



```
     &                        ACTIV,   DACTIV             )
               PX(1,1)=ACTIV(BINL(0))*PSAT(BINL(0))
               PX(2,1)=ACTIV(BINL(1))*PSAT(BINL(1))
               PX(1,2)=PRESSURE
               PX(2,2)=PRESSURE
            ENDIF

c case 3: trial phase 1 - liquid NRTL; trial phase 2 - liquid NRTL

         ELSE

c determine the liquid phase activities

            DO 140 K=1,2
               IF (IB1SUBS.NE.1.OR.IPASS.EQ.1) THEN
                  DO 30 I=1,CI
                     XMOLES(I)=NTRIAL(ISPLIT,I,K)
30                CONTINUE
                  XMOLES(BINL(IB1SUBS))=Z(1,K)

                  IF (IPASS.EQ.0) THEN
                     I         =IB1SUBS
                  ELSE
                     I=CI-1
                  ENDIF

                  DO 130 J=0,I
                     ID(J+1)=BINL(J)
130               CONTINUE

                  NSPECIES  =I+1
                  NCALC     =1
                  IDCALC(1) =BINL(IB1SUBS)
                                 MODEDERIV =0

                  CALL ACTIVITY( XMOLES, NSPECIES, ID,
     &                           NCALC,   IDCALC, MODEDERIV,
     &                           ACTIV,   DACTIV             )

                  PX(1,K)=ACTIV(BINL(IB1SUBS))

               ELSE
                  XMOLES(BINL(0))=Z(1,K)
                  XMOLES(BINL(1))=Z(2,K)
                  NSPECIES=2
                  ID(1)=BINL(0)
                  ID(2)=BINL(1)
                  NCALC=2
                  IDCALC(1)=BINL(0)
                  IDCALC(2)=BINL(1)
                  MODEDERIV=0

                  CALL ACTIVITY( XMOLES, NSPECIES, ID,
     &                           NCALC,   IDCALC, MODEDERIV,
     &                           ACTIV,   DACTIV             )

                  PX(1,K)=ACTIV(BINL(0))
                  PX(2,K)=ACTIV(BINL(1))
               ENDIF
140         CONTINUE
```

```
        ENDIF
      ENDIF


c calculate the objective function

        IF (MODE2.EQ.1.OR.MODE2.EQ.3) THEN

          IF (IB1SUBS.NE.1.OR.IPASS.EQ.1) THEN

            F(1)=Z(1,1)/XNTOT(1)*PX(1,1)
     &           -Z(1,2)/XNTOT(2)*PX(1,2)

          ELSE

            F(1)=Z(1,1)/XNTOT(1)*PX(1,1)
     &           -Z(1,2)/XNTOT(2)*PX(1,2)
            F(2)=Z(2,1)/XNTOT(1)*PX(2,1)
     &           -Z(2,2)/XNTOT(2)*PX(2,2)

          ENDIF

        ENDIF

      RETURN
      END
c
```

```
c subroutine to set up MINOS to solve the problem of finding the point
c which together with a specified fixed point forms a line which is
c tangential to the Gibbs energy curve.
c returns INFBIN=0 if an unstable binary fluid is not found
c returns INFBIN=1 if an unstable binary fluid is found

      SUBROUTINE UNSTABL(XBIN,OBJ)

      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      INCLUDE "walgaut.com"
      INCLUDE "actcom.com"
      INTEGER IDS(2)
      COMMON /UNSTAB/ IDS
      PARAMETER(MAXM=1,MAXN=1,MAXNB=MAXM+MAXN,MAXNE=MAXM*MAXN,
     &       NWCORE=1000,NNAME=1)

c data type declarations for MINOS variables

      REAL*8       A(MAXNE),    BL(MAXNB),     BU(MAXNB),
     &             XN(MAXNB),   PI(MAXM),      RC(MAXNB),
     &             Z(NWCORE),   XBIN(2),       OBJ,
     &             ACOMP(2,2),  RHS(2)

      CHARACTER*8 NAMES(5)
      INTEGER*4   HA(MAXNE),   HS(MAXNB)
      INTEGER     KA(MAXN+1),  NAME1(NNAME),  NAME2(NNAME)
      INTEGER     IPVT2(2)

c declaration of arguments passed as arrays to OBJ1

      REAL*8  X(MAXCI*MAXCK), DELF(MAXCI*MAXCK), XR(2*MAXCI),
     &         TNTOT(MAXCI)

c set up problem to be solved by MINOS

      NPROB=5
      NOBJ=1
      NJAC=0
      CALL SPECWRITER(NPROB,NOBJ,NJAC)
      ISPECS=3
      IPRINT=10
      ISUMM =6
      NOUT  =6
      BPLUS =1.D+10
      BMINUS=-BPLUS


      M=1
      NN=1
      NB=NN+M
      NE=NN*M
      IOBJ = 0
      NNCON=0
      NNOBJ=1
      NNJAC=0

      BL(1)=1.D-9
      BU(1)=1.0-1.D-9

      DO 60 I=1,M
```



```
         PI(I)=0
60     CONTINUE

       DO 70 I=1,NB
         HS(I)=0
70     CONTINUE

       DO 80 I=1,MAXNB
          RC(I)=0
80     CONTINUE

       DO 100 I=1,NWCORE
          Z(I)=0
100    CONTINUE


       INFORM=0
       MINCORE=0
       NS=0
       NINF=0
       SINF=0
       OBJ=0
       OBJADD=0


       XN(1)=0.9

       WRITE(*,*) "IDS",IDS(1),IDS(2)

       CALL MISPEC(ISPECS,IPRINT,ISUMM,NWCORE,INFORM)
       CALL MINOSS('COLD',M,NN,NB,NE,NNAME,
     &              NNCON,NNOBJ,NNJAC,
     &              IOBJ,OBJADD,NAMES,
     &              A,HA,KA,BL,BU,NAME1,NAME2,
     &              HS,XN,PI,RC,
     &              INFORM,MINCORE,NS,NINF,SINF,OBJ,
     &              Z,NWCORE)

       OBJ=OBJ-OBJADD
       INFORM1=INFORM

       WRITE(*,*) "INFORM=",INFORM

       IF (OBJ.LT.0) THEN
         WRITE(*,*) " Unstable binary fluid composition located "
         INFBIN=1
       ELSE
         WRITE(*,*) " Unstable binary fluid not found "
         WRITE(*,*) " OBJECTIVE=",OBJ
         INFBIN=0
       ENDIF

c minimize the Gibbs free energy of the unstable binary system
c initialize the mole values with equimolar amounts of the unstable
c mixture in both phases


c the number of chemical species to be included in the equilibrium
c calculation
```

```
      CIR=2
      CKT=2
      CER=2

c the identity of these species

      DO 90 I=1,CIR
         IDR(I)=IDS(I)
90    CONTINUE

c the total number of moles of each species

      TNTOT(IDR(1))=XN(1)
      TNTOT(IDR(2))=1-XN(1)

c the initial guess for phase equilibrium

      XR(1)=0.4*XN(1)
      XR(2)=0.5*(1-XN(1))
      XR(3)=(1-0.4)*XN(1)
      XR(4)=(1-0.5)*(1-XN(1))

      CALL NEWRXN(XR,TNTOT,OBJ)

c distribute the mole fraction compositions between the trial phases
c by a mass balance calculation

      ACOMP(1,1)=XR(1)
      ACOMP(1,2)=XR(3)
      ACOMP(2,1)=XR(2)
      ACOMP(2,2)=XR(4)
      LDA2=2
      N1 =2

      CALL DGEFA(ACOMP,LDA2,N1,IPVT2,INFO)

      JOB=0

      RHS(1)=NSOURCE(IDR(1))
      RHS(2)=NSOURCE(IDR(2))

      WRITE(*,*) "rhs",RHS(1),RHS(2)

      CALL DGESL(ACOMP,LDA2,N1,IPVT2,RHS,JOB)
      WRITE(*,*) "rhs",RHS(1),RHS(2)
      XBIN(1)=RHS(1)*XR(1)
      XBIN(2)=RHS(1)*XR(2)
      IF (XBIN(1).LT.0.D0.OR.XBIN(2).LT.0.D0) THEN
         WRITE(*,*) "XBIN",XBIN(1),XBIN(2)
         WRITE(*,*) " Negative mole values
     & found in subroutine unstabl "
      ENDIF

      RETURN
      END

c_____________________________________________________________
c subroutine used to set up MINOSS for the nonconvex problem with
c only two chemical species.  Note the XN on entry must be such that
c the initial guess phases are equimolar
```



```
      SUBROUTINE NEWRXN(XR,TNTOT,OBJ)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      INCLUDE "walgaut.com"
      INCLUDE "MINOSPAR.INC"
      INCLUDE "actcom.com"

c data type declarations for MINOS variables

      REAL*8   A(MAXNE),   BL(MAXNB),   BU(MAXNB),
     &         XN(MAXNB),  PI(MAXM),    RC(MAXNB),
     &         Z(NWCORE),  XR(2*MAXCI), TNTOT(MAXCI)

      CHARACTER*8 NAMES(5)
      INTEGER*4   HA(MAXNE),HS(MAXNB)
      INTEGER     KA(MAXN+1),NAME1(NNAME),NAME2(NNAME)

      NPROB=6
      NOBJ=CIR*CKT
      NJAC=0
      CALL SPECWRITER(NPROB,NOBJ,NJAC)
      ISPECS=3
      IPRINT=10
      ISUMM =6
      NOUT  =6
      SMALL =1.D-8
      BPLUS =1.D+10
      BMINUS=-BPLUS


      M=CER
      NN=CIR*CKT
      NB=NN+M
      NE=NN*M
      IOBJ = 0
      NNCON=0
      NNOBJ=CIR*CKT
      NNJAC=0

*     ****************************************************************
*     set the values in the constraint matrix
*
*              1             2   ....                      nn
*
*          1   ATOMS(1,1,1)  ATOMS(1,2,1)  ATOMS(1,1,2)  ATOMS(1,2,2)
*
*          .   ATOMS(2,1,1)
*
*         CER  ATOMS(CER,CIR,1)                          ATOMS(CER,CIR,CKX)
*
*
*     ****************************************************************
c clean up A

      DO 10 I=1,NE
         A(I)=0
10    CONTINUE
```

```
c mass balance constraints

      ICOUNT=0
      DO 20 K=1,CKT
         DO 20 II=1,CIR
            I=IDR(II)
            DO 20 JJ=1,CER
               J=IDR(JJ)
               ICOUNT=ICOUNT+1
               A(ICOUNT)=ATOMS(J,I,K)
               HA(ICOUNT)=JJ
20    CONTINUE

      DO 30 I=1,NN+1
         KA(I)=(I-1)*M+1
30    CONTINUE

c Bounds on mole variables

      ICOUNT=0
      DO 40 K=1,CKT
         DO 40 II=1,CIR
            ICOUNT=ICOUNT+1
            BL(ICOUNT)=SMALL
            BU(ICOUNT)=BPLUS
40    CONTINUE

c Bounds on the slacks for the mass balance constraints

      DO 50 II=1,CER
         I=IDR(II)
         BL(NN+II)=-TNTOT(I)
         BU(NN+II)=-TNTOT(I)
50    CONTINUE


      DO 60 I=1,M
         PI(I)=0
60    CONTINUE

      DO 70 I=1,NB
        HS(I)=0
70    CONTINUE

      DO 80 I=1,MAXNB
         RC(I)=0
80    CONTINUE

      DO 90 I=1,NN
         XN(I)=XR(I)
90    CONTINUE


      INFORM=0
      MINCORE=0
      NS=0
      NINF=0
      SINF=0
      OBJ=0
      OBJADD=0
```



```
      CALL MISPEC(ISPECS,IPRINT,ISUMM,NWCORE,INFORM)
      CALL MINOSS('COLD',M,NN,NB,NE,NNAME,
     &            NNCON,NNOBJ,NNJAC,
     &            IOBJ,OBJADD,NAMES,
     &            A,HA,KA,BL,BU,NAME1,NAME2,
     &            HS,XN,PI,RC,
     &            INFORM,MINCORE,NS,NINF,SINF,OBJ,
     &            Z,NWCORE)

      OBJ=OBJ-OBJADD
      INFORM1=INFORM

      DO 120 I=1,NN
         XR(I)=XN(I)
120   CONTINUE

      RETURN
      END


c_____________________________________________________________________
c subroutine calculates the 2nd derivative of G with respect to
c the mole fraction x1

      SUBROUTINE OBJS(NN,X,F,DELF,NSTATE,NPROB)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"
      INTEGER IDS(2)
      COMMON /UNSTAB/ IDS
      INTEGER   ID(MAXCI),    IDCALC(MAXCI)
      REAL*8    X(NN),        DELF(NN),      XMOLES(MAXCI),
     &          ACTIV(MAXCI), DACTIV(MAXCI,MAXCI)

      EPS=1.D-8
      NSPECIES=2
      ID(1)=IDS(1)
      ID(2)=IDS(2)
      NCALC=2
      IDCALC(1)=IDS(1)
      IDCALC(2)=IDS(2)
      MODEDERIV=1

c note that the "moles" sum up to 1 and are thus also mole fractions

      XMOLES(IDS(1))=X(1)
      XMOLES(IDS(2))=1-X(1)

c calculate the activities and activity derivatives

      CALL ACTIVITY( XMOLES, NSPECIES, ID,
     &                       NCALC,   IDCALC, MODEDERIV,
     &                       ACTIV,   DACTIV            )


      ITESTPOT=0
      IF (ITESTPOT.EQ.1) THEN  ! diagnostic only

c test potential 1
```

```
          F1= DLOG(XMOLES(IDS(1)))+DLOG(ACTIV(IDS(1)))
     &      -(DLOG(XMOLES(IDS(2)))+DLOG(ACTIV(IDS(2))))

          XMOLES(IDS(1))=XMOLES(IDS(1))+EPS
          XMOLES(IDS(2))=XMOLES(IDS(2))-EPS

          CALL ACTIVITY( XMOLES, NSPECIES, ID,
     &                        NCALC,   IDCALC, MODEDERIV,
     &                        ACTIV,   DACTIV             )

c test potential 2

          F2= DLOG(XMOLES(IDS(1)))+DLOG(ACTIV(IDS(1)))
     &      -(DLOG(XMOLES(IDS(2)))+DLOG(ACTIV(IDS(2))))

          XMOLES(IDS(1))=XMOLES(IDS(1))-EPS
          XMOLES(IDS(2))=XMOLES(IDS(2))+EPS

c numerical approximation of 2nd derivative

          FNUM=(F2-F1)/EPS
        ENDIF

c analytic evaluation of 2nd derivative of G

        F=1/X(1)+1/(1-X(1))+1/ACTIV(IDS(1))*
     &     (DACTIV(IDS(1),IDS(1))-DACTIV(IDS(1),IDS(2)))
     &                    -1/ACTIV(IDS(2))*
     &     (DACTIV(IDS(2),IDS(1))-DACTIV(IDS(2),IDS(2)))

c the Gibbs energy of the system

         F3=XMOLES(IDS(1))*(DLOG(XMOLES(IDS(1)))+DLOG(ACTIV(IDS(1))))
     &     +XMOLES(IDS(2))*(DLOG(XMOLES(IDS(2)))+DLOG(ACTIV(IDS(2))))

900      FORMAT(4D15.5)
901      FORMAT(14X,"X",14X,"G"13X,"G'",12X,"G''")

        RETURN
        END
c-------------------------------------------------------------------
c Objective and obj gradients for the nonconvex problem.

        SUBROUTINE NEWOBJ1(NN,X,F,DELF,NSTATE,NPROB)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INCLUDE "walgaut.com"
        INCLUDE "actcom.com"

        REAL*8    X(NN),          DELF(NN),          NPHAS(CK),
     &            TPSI(CI,CK),    PSI(CI,CK),        N(CI,CK)

        INTEGER    IDPHASTEMP(MAXCK)

        TINY=1.D-2*SMALL
        IX2N=0

        DO 10 K=1,CKT
           DO 10 II=1,CIR
             I=IDR(II)
             IX2N=IX2N+1
```



```
             N(I,K)=X(IX2N)
10      CONTINUE

        DO 20 K=1,CKT
           NPHAS(K)=0.D0


           IF (NPROB.EQ.0) THEN
              IF (K.EQ.ISOURCE) THEN
                 IDPHASTEMP(K)=IDPHAST(ISPLIT,1)
              ELSEIF (K.EQ.CKX+1) THEN
                 IDPHASTEMP(K)=IDPHAST(ISPLIT,2)
              ENDIF
           ELSEIF (NPROB.EQ.4.OR.NPROB.EQ.6) THEN
              IDPHASTEMP(K)=2
           ELSE
              IDPHASTEMP(K)=IDPHAS(K)
           ENDIF



           DO 20 II=1,CIR
              I=IDR(II)
              NPHAS(K)=NPHAS(K)+N(I,K)
              TPSI(I,K)=0.D0
              IF ( IDPHASTEMP(K).EQ.2) THEN
                 DO 40 JJ=1,CIR
                    J=IDR(JJ)
                    TPSI(I,K)=TPSI(I,K)+GNRTL(J,I)*N(J,K)
40               CONTINUE
                 PSI(I,K)= N(I,K)/TPSI(I,K)
              ENDIF
20      CONTINUE


        F=0
        ICOUNT=0
        DO 50 K=1,CKT
           DO 50 II=1,CIR
              I=IDR(II)
              ICOUNT=ICOUNT+1

              TPCUR=0.D0

c   Note phase 1 is an ideal vapour phase and this section applies only to
c   the liquid phase.
c             IF (IFLAGCV(K).EQ.1) THEN

              IF (IDPHASTEMP(K).EQ.2) THEN

                  DO 30 JJ=1,CIR
                     J=IDR(JJ)
                     TPCUR=TPCUR+GNRTL(I,J)*TAU(I,J)*PSI(J,K)
30                CONTINUE
              ENDIF

              F=F+N(I,K)*(TPCUR+
     &           ( GRT(I,K)+DLOG(N(I,K))-DLOG(NPHAS(K)) ))

c  determine gradients only if this function is being called from the
```

```
c  NLP solver.

            IF (NPROB.NE.0) THEN
               DELF(ICOUNT)=(TPCUR+
     &             ( GRT(I,K)+DLOG(N(I,K))-DLOG(NPHAS(K)) ))

               IF (IDPHASTEMP(K).EQ.2) THEN
                  DO 60 K1=1,CKT
                     DO 60 II1=1,CIR
                        I1=IDR(II1)
                        TEMP=0.D0
                        DO 70 JJ=1,CIR
                           J=IDR(JJ)
                           IF (K1.EQ.K) THEN
                              TEMP=TEMP-GNRTL(I1,J)*TAU(I1,J)*
     &                        GNRTL(I,J)*N(J,K1)/(TPSI(J,K1)**2)
                              IF (J.EQ.I) THEN
                                 TEMP = TEMP + GNRTL(I1,J)*TAU(I1,J)
     &                           /TPSI(J,K)
                              ENDIF
                           ENDIF
70                      CONTINUE
                        DELF(ICOUNT)=DELF(ICOUNT)+N(I1,K1)*TEMP
60                   CONTINUE
                  ENDIF
               ENDIF

50    CONTINUE
      RETURN
      END

c
```

```
c this datinit applies to the walgaut program only

      SUBROUTINE DATINIT(DATAFILE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "walgaut.com"

c declarations for local variables

      DOUBLE PRECISION ALPHA(MAXCI,MAXCI), ANTA(MAXCI),
     &                 ANTB(MAXCI),ANTC(MAXCI),GIBBS(MAXCI)
      CHARACTER*72 DATAFILE

      CALL READDATA (DATAFILE,ALPHA,ANTA,ANTB,ANTC,
     &               GIBBS,IRXN,IALPHA)

c Gas constant in Cal/K.mol.

       R   =1.9872


       DO 5 K=2,CK
          DO 5 I=1,CI
             DO 5 J=1,CE
                ATOMS(J,I,K)=ATOMS(J,I,1)
5      CONTINUE

c NRTL model parameters
       WRITE(*,*) "IALPHA",IALPHA
       IF (IALPHA.EQ.1) THEN
          DO 10 I = 1,CI
             DO 10 J = 1,CI
                GNRTL(I,J) =DEXP(-ALPHA(I,J)*TAU(I,J))
10        CONTINUE
       ENDIF


c Calculation of vapour pressure using the Antoine equation.

       DO 260 I=1,CI
         PSAT(I)=10**(ANTA(I)-ANTB(I)/(TEMP+ANTC(I)))
         PSAT(I)=PSAT(I)/101.3D3
260    CONTINUE

c Setting the objective function constants appropriate to the
c type of problem to be solved.
c If the system is reacting

        IF (IRXN.EQ.1) THEN

c If a gas phase is included

           IF (IFLAGCV(1).EQ.0) THEN
              DO 310 I=1,CI
                 GRT(I,1)=GIBBS(I)/R/TEMP+DLOG(PRESSURE)
                 DO 320 K=2,CK
                    GRT(I,K)=GIBBS(I)/R/TEMP+ DLOG(PSAT(I))
320              CONTINUE
310           CONTINUE
           ELSE
```



```
c If a gas phase is not included

              DO 330 K=1,CK
                 DO 330 I=1,CI
                    GRT(I,K)=GIBBS(I)/R/TEMP + DLOG(PSAT(I))
330           CONTINUE
           ENDIF

c If the system is not reacting

        ELSE

c If a gas phase is included

           IF (IFLAGCV(1).EQ.0) THEN
              DO 340 I=1,CI
                 GRT(I,1)=0.00+DLOG(PRESSURE)
                 DO 350 K=2,CK
                    GRT(I,K)=GIBBS(I)/R/TEMP + DLOG(PSAT(I))
350              CONTINUE
340           CONTINUE

c If a gas phase is not included

           ELSE
              DO 360 K=1,CK
                 DO 360 I=1,CI
                    GRT(I,K)=0.00
360           CONTINUE
           ENDIF
        ENDIF

        RETURN
        END

*-------------------------------------------------------------------------
      SUBROUTINE READDATA (DATAFILE,ALPHA,ANTA,ANTB,ANTC,
     &                     GIBBS,IRXN,IALPHA)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE 'walgaut.com'
      DOUBLE PRECISION ALPHA(MAXCI,MAXCI),ANTA(MAXCI),ANTB(MAXCI),
     &                 ANTC(MAXCI),GIBBS(MAXCI)
      CHARACTER*72 DATAFILE
      OPEN(UNIT=2,
     &FILE=DATAFILE,
     &STATUS='UNKNOWN')
      REWIND 2

      DO 2000 I=1,6
         READ(2,*)
2000  CONTINUE

      READ (2,901)  IRXN
      READ(2,901)   !CI
      READ(2,901)   !CK

      DO 2010 I=1,CK
         READ(2,901) IFLAGCV(I)
2010  CONTINUE
```

```
      DO 2020 I=1,MAXCK-CK
         READ(2,*)
2020  CONTINUE

      READ(2,901)  !CE
      READ(2,902) TEMP
      READ(2,902) PRESSURE

      DO 2030 I=1,CE
         READ(2,902) NTOT(I)
2030  CONTINUE

      DO 2035 I=1,MAXCE-CE
         READ(2,*)
2035  CONTINUE

      READ(2,*)
      READ(2,901) IALPHA
      DO 2040 I=1,4
         READ(2,*)
2040  CONTINUE

      DO 2052 J=1,CE
         READ(2,906) (ATOMS(J,I,1),I=1,CI)
2052  CONTINUE

      DO 2057 I=1,MAXCE-CE
         READ(2,*)
2057  CONTINUE

      DO 2051 I=1,5
         READ(2,*)
2051  CONTINUE

      DO 2060 I=1,CI
         READ(2,906) (TAU(I,J),J=1,CI)
2060  CONTINUE

      DO 2070 I=1,MAXCI-CI
         READ(2,*)
2070  CONTINUE

      DO 2080 I=1,5
         READ(2,*)
2080  CONTINUE

      DO 2090 I=1,CI
        READ(2,906) (GNRTL(I,J),J=1,CI)
2090  CONTINUE

      DO 2100 I=1,MAXCI-CI
         READ(2,*)
2100  CONTINUE

      DO 2110 I=1,5
         READ(2,*)
2110  CONTINUE

      DO 2120 I=1,CI
         READ(2,906) (ALPHA(I,J),J=1,CI)
```



```
2120  CONTINUE

      DO 2130 I=1,MAXCI-CI
         READ(2,*)
2130  CONTINUE

      DO 2140 I=1,5
         READ(2,*)
2140  CONTINUE

      DO 2200 I=1,CI
         READ(2,905) ANTA(I),ANTB(I),ANTC(I)
2200  CONTINUE

      DO 2205 I=1,MAXCI-CI
         READ(2,*)
2205  CONTINUE

      DO 2210 I=1,3
         READ(2,*)
2210  CONTINUE

      DO 2220 I=1,CI
         READ(2,907) GIBBS(I)
2220  CONTINUE

      DO 2230 I=1,MAXCI-CI
         READ(2,*)
2230  CONTINUE

       READ(2,*)
       READ(2,908) EPSILON
       READ(2,908) SMALL

       DO 2240 I=1,2
          READ(2,*)
2240   CONTINUE
       DO 2250 I=1,NBASIC
          READ(2,909) !IBASICI(I),IBASICK(I)
2250   CONTINUE

       CLOSE (UNIT=2)
901    FORMAT(T30,I2)
902    FORMAT(T30,D13.7D2)
905    FORMAT(T15,3D14.6D2)
906    FORMAT(T15,6D9.2)
907    FORMAT(T15,D14.6D2)
908    FORMAT(T30,D8.2D2)
909    FORMAT(T20,I2,T30,I2)
       RETURN
       END
C
```

```
c       This file contains global parameters used throughout the algorithm

        INTEGER CB,CI,CK,CE,CNB,MB,MAXNOD,NX

c******* P R O B L E M   S P E C I F I C   P A R A M E T E R S *********
c
c  CI                   Number of chemical types in the system
c  CK                   Number of phases postulated
c  CE                   Number of atomic species
c  CB=2**(CI*CK)        Systems with no gas phase
c  CB=2**(CI*(CK-1))    Systems with a gas phase

        PARAMETER(CI=3,CK=3,CE=3,MB=CE,CB=2**6,CNB=(CI*CK-CE),
     &            MAXNOD=20000,MAXCUT=20)


c*************************************************************************
        PARAMETER (MAXCI = 6,MAXCK = 4,MAXCE = 6)
        PARAMETER(NX=CI*CK+1)

c       data type declarations for common variables
        DOUBLE PRECISION EPSILON,PSAT,
     &                   TAU,    GNRTL,   GRT,     TEMP,
     &                   PRESSURE     ,   NTOT,
     &                   ATOMS        ,   P


        INTEGER CKX

        COMMON /SYSDATA/ TAU(MAXCI,MAXCI),GNRTL(MAXCI,MAXCI),
     &                   PSAT(MAXCI),GRT(CI,CK),TEMP,PRESSURE,
     &                   NTOT(MAXCI),ATOMS(MAXCE,MAXCI,MAXCK),
     &                   P(MAXCI),IFLAGCV(CK)


        COMMON /SMALLVAL/EPSILON

        COMMON /TPLANEDAT/CKX
```

Appendix D

Tangent Plane FORTRAN Program

```
      PROGRAM TANGRID

      INCLUDE "tplane.com"
      CHARACTER*72 DATAFILE
      CHARACTER*72 OUTFILEA,OUTFILEB,OUTFILEC,OUTFILED

      REAL*8 NTOTR(MAXCI),X(MAXCI,MAXCK),XU(3),XL(3)
      REAL*8 DELF(MAXCI),   Y(CI)

      DATAFILE="/home/cliff/fortran/split/datafile/prop-bute-wate.dat"
      OUTFILEA ="/home/cliff/fortran/split/outfiles/prop-bute-wate.tpd"
      CALL DATINIT(DATAFILE)
      OPEN (UNIT=8,FILE=OUTFILEA,STATUS="unknown")

c lowest value for X(1)
       XL(1)=0.0D0
c highest value for X(1)
       XU(1)=0.4D0

c lowest value for X(3)
      XL(3)=0.6D0
c higest value for X(3)
      XU(3)=1.0D0

      DELTA=0.0025

      NTOTR(3)=XL(3)
      DO 10 WHILE(NTOTR(3).LE.XU(3))
         REWIND (UNIT=7)
         REWIND (UNIT=10)

         NTOTR(1)=XL(1)

         IFEAS=1
         IFEAS2=1
         DO 20 WHILE(IFEAS2.EQ.1.AND.NTOTR(1).LE.XU(1))
            NTOTR(2)=1.0D0-NTOTR(1)-NTOTR(3)

            IFEAS=1
            I=1
            DO 40 I=1,3

               IF(NTOTR(I).LT.1.0D0) THEN
                  IF(NTOTR(I).LT.1.0D-5) THEN
                     IFEAS=0
                  ENDIF
               ELSE
                  IFEAS=0
                  IFEAS2=0
               ENDIF

40             CONTINUE

            IF (IFEAS.EQ.1) THEN

               DO 30 I=1,3
                  NTOT(I)=NTOTR(I)
                  Y(I)=NTOT(I)
30                CONTINUE
```



```
               CALL TPLANE(X)
               WRITE(8,1002)  NTOTR(1),NTOTR(2),NTOTR(3),
     &X(1,1),X(2,1),X(3,1),X(1,2),X(2,2),X(3,2)

            ENDIF
            NTOTR(1)=NTOTR(1)+DELTA
20       CONTINUE
         NTOTR(3)=NTOTR(3)+DELTA
10    CONTINUE

      CLOSE(UNIT=8)

1000   FORMAT(9(D10.4,2X))
1002   FORMAT(9(D12.6,2X))
1003   FORMAT(6(D12.6,2X))
1004   FORMAT(4(D12.6,2X))

       END

c_____________________________________________________________
c program uses Michelsen's tangent plane idea to check for phase instability
c and to initiate the equilibrium computation.

       SUBROUTINE TPLANE(X)

       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       INCLUDE "tplane.com"
       INCLUDE "tpminos.com"
       REAL*8  N(CI,MAXCK),   Y(CI),   PHAS(MAXCK)
       REAL*8 X(MAXCI,MAXCK)
       EPSDIST=1.D-8

       CKX=1
       DO 10 K=1,CKX
          DO 10 I=1,CI
             N(I,K)=NTOT(I)
10     CONTINUE

       XDIST=-1
       GMIN=1.D10
       ITER=0
       DO 20 WHILE (XDIST.LT.-EPSDIST)
          ITER=ITER+1
          IF (ITER.GT.10) RETURN
          CALL EQUILCALC(N,PHAS,Y,XDIST,OBJ)
          IF (XDIST.LT.-EPSDIST.AND.CKX.LT.CK) THEN
             DO 30 I=1,CI

                N(I,CKX+1) = 1.D-1*PHAS(CKX)*Y(I)
                WRITE (*,*) "N",I,CKX+1,N(I,CKX+1)
                N(I,CKX)    = N(I,CKX)-N(I,CKX+1)
                WRITE (*,*) "N",I,CKX,N(I,CKX)

30           CONTINUE
             CKX=CKX+1
          ENDIF
20     CONTINUE

       WRITE(1,900)
       DO 40 K=1,CKOPT
```

```
          DO 40 I=1,CI
             WRITE(7,901) I,K,NOPT(I,K)
40      CONTINUE
        WRITE(7,*) "OBJECTIVE=",GMIN
900     FORMAT(" CHEM  PHASE  MOLES  ")
901     FORMAT(I5,2X,I5,2X,D10.5)

        DO 50 K=1,CKOPT
           PHAS(K)=0.0D0
           DO 50 I=1,CI
              PHAS(K)=PHAS(K)+NOPT(I,K)
50      CONTINUE

        DO 60 K=1,CKOPT
           DO 60 I=1,CI
              X(I,K)=NOPT(I,K)/PHAS(K)
60      CONTINUE

        RETURN
        END
C_______________________________________________________________________
        SUBROUTINE EQUILCALC(N,PHAS,YMIN,XMINDIST,OBJ)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INCLUDE "tplane.com"
        INCLUDE "actcom.com"
        INCLUDE "tpminos.com"
        DOUBLE PRECISION N(CI,MAXCK),Y(CI),Y2(CI,MAXCK),ICOAL(MAXCK),
     &                   PHAS(MAXCK),YMIN(CI),XN(MAXNB)


c minimize the equilibrium problem

        WRITE(*,*) "**  E Q U I L C A L C **"
        WRITE(*,*) "CKX=",CKX

c initialize the mole variables

        ICOUNT=0
        DO 250 K=1,CKX
           DO 250 I=1,CI
              ICOUNT=ICOUNT+1
             XN(ICOUNT)=N(I,K)
250     CONTINUE


c phase equilibrium problems are trivial if there is only one phase!

        IF (CKX.GT.1) THEN
           CKT=CKX
           CALL RXN(XN,OBJ)
           WRITE(7,*) 'Gibbs Energy Minimization Results'
           ICOUNT=0
           DO 260 K=1,CKX
              DO 260 I=1,CI
                 ICOUNT=ICOUNT+1
                 N(I,K)=XN(ICOUNT)
                 WRITE(7,*) I,K,N(I,K)
260        CONTINUE

        ELSE
```



```
           NN=CI
           CALL OBJ1(NN,XN,OBJ,DELF,NSTATE,NPROB)
           WRITE(7,*) "Trivial Solution Energy",OBJ

        ENDIF

C store the best solution encountered

        IF (OBJ.LT.GMIN) THEN
           GMIN=OBJ
           CKOPT=CKX
           DO 265 K=1,CKOPT
              DO 265 I=1,CI
                 NOPT(I,K)=N(I,K)
265        CONTINUE
        ENDIF


c the source phase

        KTEST=1
        BPHAS=0    ! the phase with most moles

c calculate the total moles in each phase

        DO 40 K=1,CKX
           PHAS(K) = 0
           DO 45 I = 1,CI
              PHAS(K) = PHAS(K)+N(I,K)
45         CONTINUE
           IF (PHAS(K).GT.BPHAS) THEN
              KTEST=K
              BPHAS=PHAS(K)
           ENDIF
40      CONTINUE


C calculate the mole fractions in each phase

        DO 30 K=1,CKX
           DO 30 I=1,CI
              Y2(I,K) = N(I,K)/PHAS(K)
              Y(I)=Y2(I,K)
30      CONTINUE

c check for phase coalescence

        DO 110 K=1,CKX
           IF (ICOAL(K).EQ.0) THEN
              DO 50 K2=K+1,CKX
                 IF ( PHAS(K2) .LT. 1.D-9 ) THEN
                    ICOAL(K2)=1
                 ELSE
                    I=0
                    ISPLIT=0

                    DO 60 WHILE ((I.LT.CI).AND.(ISPLIT.EQ.0))
                       I=I+1
                       IF ( ABS(Y2(I,K2)-Y2(I,K)) .GT. 1.D-6) THEN
```

```
                        ISPLIT=1
                     ENDIF
60                CONTINUE

                  IF (ISPLIT.EQ.0) THEN
                     ICOAL(K2)=1
                     DO 70 I=1,CI
                        N(I,K)=N(I,K)+N(I,K2)
                        N(I,K2)=0.D0
                        WRITE(*,*) "COALESCE",I,K,N(I,K)
                        IF (KTEST.EQ.K2) KTEST=K
70                   CONTINUE
                  ENDIF
               ENDIF
50          CONTINUE
         ENDIF
110   CONTINUE

      DO 80 K=1,CKX
         IF (ICOAL(K).EQ.1) THEN
            CKX=CKX-1
            IF (KTEST.GT.K) KTEST=KTEST-1
            DO 90 K1=K,CKX
               ICOAL(K)=ICOAL(K+1)
               DO 100 I=1,CI
                  N(I,K1)=N(I,K1+1)
100            CONTINUE
90          CONTINUE
         ENDIF
80    CONTINUE

      DO 120 K=1,CKX
         DO 120 I=1,CI
            WRITE(*,*) "N",N(I,K)
120   CONTINUE


      CALL POTENTIAL(KTEST,N)

c initialize the mole fractions for the tangent plane minimization
c the algorithm is initialized using a number of different starting points
c to improve the chances of it finding the global minimum.

      XMINDIST=1.D10
      WRITE(7,*)
      WRITE(7,*)
      WRITE(7,1000)
      WRITE(7,*)
      WRITE(7,1002)
      DO 150 J=1,CI
         DO 140 I=1,CI
            IF (J.NE.I) THEN
               Y(I)=SMALL
            ELSE
               Y(I)=1.0D0-(CI-1)*SMALL
            ENDIF
            WRITE(*,*) "I,Y(I)",I,Y(I)
140      CONTINUE

         CALL TPLANEMIN(Y,XDIST)
```



```
         WRITE(7,1001) Y(1),Y(2),Y(3),XDIST

         IF (XDIST.LT.XMINDIST) THEN
            XMINDIST=XDIST
            DO 160 I=1,CI
               YMIN(I)=Y(I)
160         CONTINUE
         ENDIF
150   CONTINUE

1000  FORMAT('Tangent plane minimization results')
1001  FORMAT(4D14.5,2X)
1002  FORMAT('mole frac Y(1)',2X,'mole frac Y(2)',2X,
     &'mole frac Y(3)',2X,'min distance')

      RETURN
      END



c_____________________________________________________________________
c     subroutine used to set up MINOSS for the nonconvex problem.
      SUBROUTINE RXN(XN,OBJ)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      INCLUDE "tplane.com"
      INCLUDE "actcom.com"
      INCLUDE "tpminos.com"

c     data type declarations for MINOS variables

      DOUBLE PRECISION  A(MAXNE),BL(MAXNB),BU(MAXNB),
     &                  XN(MAXNB),PI(MAXM),RC(MAXNB),Z(NWCORE)
      DOUBLE PRECISION N(CI,CK)

      CHARACTER*8 NAMES(5)
      INTEGER*4   HA(MAXNE),HS(MAXNB)
      INTEGER     KA(MAXN+1),NAME1(NNAME),NAME2(NNAME)

      NPROB=1
      NOBJ=CI*CKT
      NJAC=0
      CALL SPECWRITER(NPROB,NOBJ,NJAC)
      ISPECS=3
      IPRINT=10
      ISUMM =6
      NOUT  =6
      BPLUS =1.D+10
      BMINUS=-BPLUS


      M=CE
      NN=CI*CKT
      WRITE(7,*) 'CKT',CKT
      NB=NN+M
      NE=NN*M
      IOBJ = 0
      NNCON=0
      NNOBJ=CI*CKT
```

```
      NNJAC=0

*************************************************************************
*     set the values in the constraint matrix
*
*            1            2    ....                      nn
*
*         1  ATOMS(1,1,1) ATOMS(1,2,1) ATOMS(1,1,2) ATOMS(1,2,2)
*
*         .  ATOMS(2,1,1)
*
*        CE  ATOMS(CE,CI,1)                          ATOMS(CE,CI,CKT)
*
*
*************************************************************************
c clean up A

      DO 230 I=1,NE
         A(I)=0
230   CONTINUE


c mass balance constraints

      ICOUNT=0
      DO 100 K=1,CKT
         DO 100 I=1,CI
            DO 100 J=1,CE
               ICOUNT=ICOUNT+1
               A(ICOUNT)=ATOMS(J,I,K)
               HA(ICOUNT)=J
100   CONTINUE

      DO 50 I=1,NN+1
         KA(I)=(I-1)*M+1
50    CONTINUE

c Bounds on mole variables

      ICOUNT=0
      DO 70 K=1,CKT
         DO 70 I=1,CI
            ICOUNT=ICOUNT+1
            BL(ICOUNT)=SMALL
            BU(ICOUNT)=BPLUS
70    CONTINUE

c Bounds on the slacks for the mass balance constraints

      DO 110 I=1,CE
         BL(NN+I)=-NTOT(I)
         BU(NN+I)=-NTOT(I)
110   CONTINUE


      DO 25 I=1,M
         PI(I)=0
25    CONTINUE

      DO 150 I=1,NB
```



```
         HS(I)=0
150   CONTINUE

      DO 210 I=1,MAXNB
         RC(I)=0
210   CONTINUE

      INFORM=0
      MINCORE=0
      NS=0
      NINF=0
      SINF=0
      OBJ=0
      OBJADD=0

      CALL MISPEC(ISPECS,IPRINT,ISUMM,NWCORE,INFORM)
      CALL MINOSS('COLD',M,NN,NB,NE,NNAME,
     &            NNCON,NNOBJ,NNJAC,
     &            IOBJ,OBJADD,NAMES,
     &            A,HA,KA,BL,BU,NAME1,NAME2,
     &            HS,XN,PI,RC,
     &            INFORM,MINCORE,NS,NINF,SINF,OBJ,
     &            Z,NWCORE)

      OBJ=OBJ-OBJADD
      INFORM1=INFORM

      ICOUNT=0
      DO 250 K=1,CKT
         DO 250 I=1,CI
            ICOUNT=ICOUNT+1
            N(I,K)=XN(ICOUNT)
            WRITE(*,*) N(I,K)
250   CONTINUE

      WRITE(*,*) 'OBJECTIVE=',OBJ
      RETURN
      END


c_____________________________________________________________________
      SUBROUTINE TPLANEMIN(XN,OBJ)

c sets up the tangent plane problem for MINOS 5.4

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "tplane.com"
      INCLUDE "tpminos.com"


c data type declarations for MINOS variables

      DOUBLE PRECISION  A(MAXNE),BL(MAXNB),BU(MAXNB),
     &                  XN(MAXNB),PI(MAXM),RC(MAXNB),Z(NWCORE)
      CHARACTER*8 NAMES(5)
      INTEGER*4   HA(MAXNE),HS(MAXNB)
      INTEGER     KA(MAXN+1),NAME1(NNAME),NAME2(NNAME)

      NPROB=2
      NOBJ=CI
```

```
      NJAC=0
      CALL SPECWRITER(NPROB,NOBJ,NJAC)
      ISPECS=3
      IPRINT=10
      ISUMM =6
      NOUT  =6
      BPLUS =1.D+10
      BMINUS=-BPLUS


      M=1
      NN=CI
      NB=NN+M
      NE=NN*M
      IOBJ=0
      NNCON=0
      NNOBJ=NN
      NNJAC=0

*-----------------------------------------------------------------
*     set the values in the constraint matrix
*
*            1       2        ...    CI
*
*     1      1       1                1
*
*
*-----------------------------------------------------------------


      NAMES(1)='Tangent Plane'
      NAMES(2)='Tangent Plane Distance'
      NAMES(3)='Mole fractions   '
      NAMES(4)='Ranges   '
      NAMES(5)='Bounds   '

      DO 10 I=1,CI
         A(I)=1
10    CONTINUE

      DO 20 I=1,NN+1
         KA(I)=(I-1)*M+1
20    CONTINUE

      ICOUNT=0
      DO 30 I=1,NN
            ICOUNT=ICOUNT+1
            HA(ICOUNT)=1
30    CONTINUE

c bounds on mole fractions

      DO 40 I=1,CI
         BL(I)=SMALL
         BU(I)=1
40    CONTINUE

c sum of mole fractions

      BL(CI+1)=-1
```



```
      BU(CI+1)=-1

      DO 25 I=1,M
         PI(I)=0
25    CONTINUE

      DO 150 I=1,NB
         HS(I)=0
150   CONTINUE

      DO 210 I=1,MAXNB
         RC(I)=0
210   CONTINUE

      INFORM  =0
      MINCORE =0
      NS      =0
      NINF    =0
      SINF    =0
      OBJ     =0
      OBJADD  =0

      DO 220 I=1,NWCORE
         Z(I)=0
220   CONTINUE

      CALL MISPEC(ISPECS,IPRINT,ISUMM,NWCORE,INFORM)
      CALL MINOSS('COLD',M,NN,NB,NE,NNAME,
     &            NNCON,NNOBJ,NNJAC,
     &            IOBJ,OBJADD,NAMES,
     &            A,HA,KA,BL,BU,NAME1,NAME2,
     &            HS,XN,PI,RC,
     &            INFORM,MINCORE,NS,NINF,SINF,OBJ,
     &            Z,NWCORE)

      OBJ=OBJ-OBJADD
      INFORM1=INFORM

      RETURN
      END


c-----------------------------------------------------------------
      SUBROUTINE FUNOBJ(MODE,NN,X,F,DELF,NSTATE,NPROB,Z,NWCORE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "tplane.com"
      DOUBLE PRECISION X(NN),DELF(NN),Z(NWCORE)
      COMMON  /M1FILE/IREAD,IPRINT,ISUMM

      IF (NPROB.EQ.1) CALL OBJ1(NN,X,F,DELF,NSTATE,NPROB)
      IF (NPROB.EQ.2) CALL TDISTANCE(NN,X,F,DELF,NSTATE,NPROB)

      RETURN
      END

c-----------------------------------------------------------------
      SUBROUTINE FUNCON(MODE,M,NN,NJAC,X,G,DELG,
     &                  NSTATE,NPROB,Z,NWCORE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

```
      INCLUDE "tplane.com"
      DOUBLE PRECISION G(M),DELG(M,NN),Z(NWCORE)
      COMMON  /M1FILE/IREAD,IPRINT,ISUMM
      RETURN
      END
c-----------------------------------------------------------------------
c     Objective and obj gradients for the nonconvex problem.

      SUBROUTINE OBJ1(NN,X,F,DELF,NSTATE,NPROB)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "tplane.com"
      DOUBLE PRECISION X(NN),DELF(NN),NPHAS(CK),TPSI(CI,CK),
     &                 PSI(CI,CK),N(CI,CK)
      TINY=1.D-2*SMALL
      IX2N=0

      DO 10 K=1,CKX
         DO 10 I=1,CI
           IX2N=IX2N+1
           N(I,K)=X(IX2N)
10       CONTINUE


      DO 20 K=1,CKX
         NPHAS(K)=0.D0
         DO 20 I=1,CI
            NPHAS(K)=NPHAS(K)+N(I,K)
            TPSI(I,K)=0.D0
            DO 40 J=1,CI
               TPSI(I,K)=TPSI(I,K)+GNRTL(J,I)*N(J,K)
40          CONTINUE
         PSI(I,K)= N(I,K)/TPSI(I,K)
20    CONTINUE


      F=0
      ICOUNT=0
      DO 50 K=1,CKX
         DO 50 I=1,CI
            ICOUNT=ICOUNT+1

            TPCUR=0.D0

c  this section applies only to the liquid phase.

            IF (IFLAGCV(K).EQ.1) THEN
               DO 30 J=1,CI
                  TPCUR=TPCUR+GNRTL(I,J)*TAU(I,J)*PSI(J,K)
30             CONTINUE
            ENDIF

            F=F+N(I,K)*(TPCUR+
     &          ( GRT(I,K)+DLOG(N(I,K))-DLOG(NPHAS(K)) ))

            DELF(ICOUNT)=(TPCUR+
     &          ( GRT(I,K)+DLOG(N(I,K))-DLOG(NPHAS(K)) ))

            DO 60 K1=1,CK
               DO 60 I1=1,CI
                  TEMP=0.D0
```



```
                  DO 70 J=1,CI
                     IF (K1.EQ.K) THEN
                        TEMP=TEMP-GNRTL(I1,J)*TAU(I1,J)*
     &                  GNRTL(I,J)*N(J,K1)/(TPSI(J,K1)**2)
                        IF (J.EQ.I) THEN
                           TEMP = TEMP + GNRTL(I1,J)*TAU(I1,J)
     &                     /TPSI(J,K)
                        ENDIF
                     ENDIF
70                CONTINUE
                  DELF(ICOUNT)=DELF(ICOUNT)+N(I1,K1)*TEMP
60          CONTINUE

50    CONTINUE

      RETURN
      END
*-----------------------------------------------------------------------
c  calculates the tangent plane distance objective function

      SUBROUTINE TDISTANCE(NN,Y,F,DELF,NSTATE,NPROB)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "tplane.com"

      REAL*8   DELF(NN),   Y(CI),   SUM1(CI),   SUM2(CI)

      DO 10 J=1,CI
         SUM1(J)=0.D0
         DO 10 L=1,CI
            SUM1(J)=SUM1(J)+GNRTL(L,J)*Y(L)
10    CONTINUE

      DO 20 J=1,CI
         SUM2(J)=0.D0
         DO 20 L=1,CI
            SUM2(J)=SUM2(J)+TAU(J,L)*GNRTL(J,L)*Y(L)/SUM1(L)
20    CONTINUE


      F=0

      K=1
      DO 30 I=1,CI
         F= F+Y(I)*(GRT(I,K)+DLOG(Y(I))+SUM2(I)-MUZ(I))
30    CONTINUE


      DO 40 J=1,CI
         DELF(J)=GRT(J,K)+DLOG(Y(J))+SUM2(J)-MUZ(J)
         DO 50 I=1,CI
            IF (I.EQ.J) THEN
               DELF(J)=DELF(J)+1
            ENDIF
            DO 60 L=1,CI
               IF (J.EQ.L) THEN
                  DELF(J)=DELF(J)+Y(I)*(TAU(I,L)*GNRTL(I,L)/SUM1(L))
               ENDIF
               DELF(J)=DELF(J)-Y(I)*(TAU(I,L)*GNRTL(I,L)*Y(L)
     &           *GNRTL(J,L)/SUM1(L)**2)
```

```
60          CONTINUE
50        CONTINUE
40      CONTINUE
        RETURN
        END

c_____________________________________________________________
        SUBROUTINE POTENTIAL(K,N)

c calculates the chemical potential

        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INCLUDE "tplane.com"
        DOUBLE PRECISION Y(CI),SUM1(CI),SUM3(CI),SUM4(CI),
     &    NPHAS,N(CI,CK)

        NPHAS=0
        DO 40 I=1,CI
           NPHAS=NPHAS + N(I,K)
40      CONTINUE

        DO 50 I=1,CI
           Y(I)=N(I,K)/NPHAS
50      CONTINUE

        DO 10 J=1,CI
           SUM1(J)=0
           SUM3(J)=0
              DO 10 L=1,CI
              SUM1(J)=SUM1(J)+GNRTL(L,J)*Y(L)
              SUM3(J)=SUM3(J)+TAU(L,J)*GNRTL(L,J)*Y(L)
10      CONTINUE

        DO 20 I=1,CI
           SUM4(I)=0
           DO 20 J=1,CI
              SUM4(I)=SUM4(I)+(GNRTL(I,J)*Y(J)/SUM1(J))*
     &                (TAU(I,J)-SUM3(J)/SUM1(J))
20      CONTINUE

        DO 30 I=1,CI
           MUZ(I)=GRT(I,K) + DLOG(Y(I)) + SUM3(I)/SUM1(I) + SUM4(I)
30      CONTINUE


        WRITE(7,*)
        WRITE(7,*)
        WRITE(7,1000)
        DO 200 I=1,CI
           WRITE(7,1001) I,MUZ(I)
200     CONTINUE

1000    FORMAT('Chem',2X,'Chemical Potential')
1001    FORMAT( I4,2X,D12.5)

        RETURN
        END
c_____________________________________________________________

      SUBROUTINE DATINIT(DATAFILE)
```



```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE "tplane.com"

c declarations for local variables

       REAL*8 ALPHA(MAXCI,MAXCI),  ANTA(MAXCI),  ANTB(MAXCI),
     &        ANTC(MAXCI),         GIBBS(MAXCI)

       CHARACTER*72 DATAFILE

       CALL READDATA (DATAFILE,ALPHA,ANTA,ANTB,ANTC,
     &                GIBBS,IRXN,IALPHA)

c Gas constant in Cal/K.mol.

       R   =1.9872


       DO 5 K=2,CK
          DO 5 I=1,CI
             DO 5 J=1,CE
                ATOMS(J,I,K)=ATOMS(J,I,1)
5      CONTINUE

c NRTL model parameters
       WRITE(*,*) "IALPHA",IALPHA
       IF (IALPHA.EQ.1) THEN
          DO 10 I = 1,CI
             DO 10 J = 1,CI
                GNRTL(I,J) =DEXP(-ALPHA(I,J)*TAU(I,J))
10        CONTINUE
       ENDIF


c Calculation of vapour pressure using the Antoine equation.

       DO 260 I=1,CI
         PSAT(I)=10**(ANTA(I)-ANTB(I)/(TEMP+ANTC(I)))
         PSAT(I)=PSAT(I)/101.3D3
260    CONTINUE

c Setting the objective function constants appropriate to the
c type of problem to be solved.
c If the system is reacting

        IF (IRXN.EQ.1) THEN

c If a gas phase is included

           IF (IFLAGCV(1).EQ.0) THEN
              DO 310 I=1,CI
                 GRT(I,1)=GIBBS(I)/R/TEMP+DLOG(PRESSURE)
                 DO 320 K=2,CK
                    GRT(I,K)=GIBBS(I)/R/TEMP+ DLOG(PSAT(I))
320              CONTINUE
310           CONTINUE
           ELSE

c If a gas phase is not included
```

```
               DO 330 K=1,CK
                  DO 330 I=1,CI
                     GRT(I,K)=GIBBS(I)/R/TEMP + DLOG(PSAT(I))
330            CONTINUE
             ENDIF

c If the system is not reacting

          ELSE

c If a gas phase is included

             IF (IFLAGCV(1).EQ.0) THEN
                DO 340 I=1,CI
                   GRT(I,1)=0.00+DLOG(PRESSURE)
                   DO 350 K=2,CK
                      GRT(I,K)=GIBBS(I)/R/TEMP + DLOG(PSAT(I))
350                CONTINUE
340             CONTINUE

c If a gas phase is not included

             ELSE
                DO 360 K=1,CK
                   DO 360 I=1,CI
                      GRT(I,K)=0.00
360             CONTINUE
             ENDIF
          ENDIF

          RETURN
          END

*-----------------------------------------------------------------------
      SUBROUTINE READDATA (DATAFILE,ALPHA,ANTA,ANTB,ANTC,
     &                     GIBBS,IRXN,IALPHA)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE 'tplane.com'
      DOUBLE PRECISION ALPHA(MAXCI,MAXCI),ANTA(MAXCI),ANTB(MAXCI),
     &                 ANTC(MAXCI),GIBBS(MAXCI)
      CHARACTER*72 DATAFILE
      OPEN(UNIT=2,
     &FILE=DATAFILE,
     &STATUS='UNKNOWN')
      REWIND 2

      DO 2000 I=1,6
         READ(2,*)
2000  CONTINUE

      READ (2,901)  IRXN
      READ(2,901)   !CI
      READ(2,901)   !CK

      DO 2010 I=1,CK
         READ(2,901) IFLAGCV(I)
2010  CONTINUE
      DO 2020 I=1,MAXCK-CK
         READ(2,*)
2020  CONTINUE
```



```
      READ(2,901)  !CE
      READ(2,902) TEMP
      READ(2,902) PRESSURE

      DO 2030 I=1,CE
         READ(2,902) NTOT(I)
2030  CONTINUE
      DO 2035 I=1,MAXCE-CE
         READ(2,*)
2035  CONTINUE
      READ(2,*)
      READ(2,901) IALPHA
      DO 2040 I=1,4
         READ(2,*)
2040  CONTINUE
      DO 2052 J=1,CE
         READ(2,906) (ATOMS(J,I,1),I=1,CI)
2052  CONTINUE
      DO 2057 I=1,MAXCE-CE
         READ(2,*)
2057  CONTINUE

      DO 2051 I=1,5
         READ(2,*)
2051  CONTINUE

      DO 2060 I=1,CI
         READ(2,906) (TAU(I,J),J=1,CI)
2060  CONTINUE
      DO 2070 I=1,MAXCI-CI
         READ(2,*)
2070  CONTINUE

      DO 2080 I=1,5
         READ(2,*)
2080  CONTINUE

      DO 2090 I=1,CI
        READ(2,906) (GNRTL(I,J),J=1,CI)
2090  CONTINUE
      DO 2100 I=1,MAXCI-CI
         READ(2,*)
2100  CONTINUE

      DO 2110 I=1,5
         READ(2,*)
2110  CONTINUE

      DO 2120 I=1,CI
         READ(2,906) (ALPHA(I,J),J=1,CI)
2120  CONTINUE
      DO 2130 I=1,MAXCI-CI
         READ(2,*)
2130  CONTINUE

      DO 2140 I=1,5
         READ(2,*)
2140  CONTINUE
```

```
      DO 2200 I=1,CI
         READ(2,905) ANTA(I),ANTB(I),ANTC(I)
2200  CONTINUE
      DO 2205 I=1,MAXCI-CI
         READ(2,*)
2205  CONTINUE

      DO 2210 I=1,3
         READ(2,*)
2210  CONTINUE

      DO 2220 I=1,CI
         READ(2,907) GIBBS(I)
2220  CONTINUE
      DO 2230 I=1,MAXCI-CI
         READ(2,*)
2230  CONTINUE

       READ(2,*)
       READ(2,908) EPSILON
       READ(2,908) SMALL

       DO 2240 I=1,2
          READ(2,*)
2240   CONTINUE
       DO 2250 I=1,NBASIC
          READ(2,909) IBASICI(I),IBASICK(I)
2250   CONTINUE


       CLOSE (UNIT=2)


901   FORMAT(T30,I2)
902   FORMAT(T30,D13.7D2)
905   FORMAT(T15,3D14.6D2)
906   FORMAT(T15,6D9.2)
907   FORMAT(T15,D14.6D2)
908   FORMAT(T30,D8.2D2)
909   FORMAT(T20,I2,T30,I2)
      RETURN
      END

C_____________________________________________________________________
      SUBROUTINE SPECWRITER(NPROB,NOBJ,NJAC)

      REAL      DTIME
      REAL      TIMEA(2)

      T1=DTIME(TIMEA)
      WRITE(*,*) T1,TIMEA(1),TIMEA(2)
      ISPEC=3
      OPEN (UNIT=ISPEC,FILE="fort.3",STATUS="unknown")
      REWIND ISPEC

      WRITE(ISPEC,*)    "Begin Nonconvex                                "
      WRITE(ISPEC,*)    "   Problem number                    ",NPROB
      WRITE(ISPEC,*)    "   Minimize                                    "
      WRITE(ISPEC,*)    "   Jacobian                        Dense       "
```



```
      WRITE(ISPEC,*)    "   Nonlinear objective variables     ",NOBJ
      WRITE(ISPEC,*)    "   Nonlinear jacobian variables      ",NJAC
      WRITE(ISPEC,*)    "   Major iterations                50000       "
      WRITE(ISPEC,*)    "   Minor iterations                  100       "
      WRITE(ISPEC,*)    "   Penalty parameter                  10       "
      WRITE(ISPEC,*)    "   Feasibility tolerance          1.0D-14      "
      WRITE(ISPEC,*)    "   Function precision             1.0D-14      "
      WRITE(ISPEC,*)    "   Optimality tolerance           1.0D-14      "
*     WRITE(ISPEC,*)    "*  LU Factor tolerance                 2       "
*     WRITE(ISPEC,*)    "*  LU Update tolerance                 2       "
      WRITE(ISPEC,*)    "   Hessian dimension                  33       "
      WRITE(ISPEC,*)    "   Derivative level                    0       "
*     WRITE(ISPEC,*)    "*  Verify gradients                            "
      WRITE(ISPEC,*)    "   Verify level                        1       "
      WRITE(ISPEC,*)    "   Summary file                       -6       "
      WRITE(ISPEC,*)    "   Summary frequency                   0       "
      WRITE(ISPEC,*)    "   Scale option                        1       "
*     WRITE(ISPEC,*)    "*  Scale tolerance                   0.9       "
      WRITE(ISPEC,*)    "   Iterations                       5000       "
      WRITE(ISPEC,*)    "   Debug level                         0       "
      WRITE(ISPEC,*)    "   Print level (jflxb)             00000       "
      WRITE(ISPEC,*)    "   Print frequency                     0       "
      WRITE(ISPEC,*)    "   Summary level                       1       "
      WRITE(ISPEC,*)    "   Summary frequency                   0       "
      WRITE(ISPEC,*)    "   Linesearch tolerance          0.00001       "
*     WRITE(ISPEC,*)    "*  Linesearch debug after iteration   30       "
      WRITE(ISPEC,*)    "   Subspace tolerance                0.5       "
      WRITE(ISPEC,*)    "End Relaxed_Dual


      CLOSE(UNIT=3)
10    CONTINUE

      T1=DTIME(TIMEA)
      WRITE(*,*) T1,TIMEA(1),TIMEA(2)

      RETURN
      END
C_____________________________________________________________________
      SUBROUTINE MATMOD
      RETURN
      END

C_____________________________________________________________________
```

```
C       This file contains global parameters used throughout the tangent
C       plane algorithm

        INTEGER CB,CI,CK,CE,CNB,LDA,MB,NBASIC,NX

C******* P R O B L E M   S P E C I F I C   P A R A M E T E R S *********
C
C  CI                   Number of chemical types in the system
C  CK                   Number of phases postulated
C  CE                   Number of atomic species

        PARAMETER(CI=3,CK=3,CE=3,MB=CE,CB=2**6,CNB=(CI*CK-CE))


C***********************************************************************
        PARAMETER (MAXCI = 6,MAXCK = 4,MAXCE = 6)
        PARAMETER(NX=CI*CK+1)
        PARAMETER(LDA=MB,NBASIC=MB)

C       data type declarations for common variables
        DOUBLE PRECISION EPSILON, PSAT,
     &                   TAU,    GNRTL,   GRT,     TEMP,
     &                   PRESSURE     ,   NTOT,   ATOMS ,
     &                   ABASIC, SMALL,   EPSILON, EPSILON2,
     &                   FEATOL

        DOUBLE PRECISION MUZ,NOPT,GMIN

        INTEGER CKOPT

        INTEGER IBASICI ,IBASICK,
     &          INBASICI,INBASICK,IFLAGCV,CKX

        INTEGER  IPVT


        COMMON /SYSDATA/ TAU(MAXCI,MAXCI),GNRTL(MAXCI,MAXCI),
     &                   PSAT(MAXCI),GRT(CI,CK),TEMP,PRESSURE,
     &                   NTOT(MAXCI),ATOMS(MAXCE,MAXCI,MAXCK),
     &                   ABASIC(LDA,NBASIC),
     &                   IPVT(NBASIC),
     &                   IBASICI(CE),IBASICK(CE),
     &                   INBASICI(CNB),INBASICK(CNB),IFLAGCV(CK)


        COMMON /SMALLVAL/SMALL,EPSILON,EPSILON2,FEATOL

        COMMON /TPLANEDAT/ GMIN,NOPT(CI,CK),MUZ(CI),CKX,CKOPT
```

```
      REAL*8    NSOURCE(MAXCI),        NTRIAL(2*MAXCI,MAXCI,2),
     &          GSPLIT(MAXCK)

      INTEGER   IDPHAST(2*MAXCI,2),    IDPHAS(2*MAXCK),BINL(0:MAXCI),
     &          IB1,        IB2,       MODE,
     &          ISPLIT,     IB1SUBS,   CKT,
     &          ISOURCE,    ISPLITL(2*MAXCK),
     &          IPASS,      CIR,       IDR(MAXCI),
     &          CER

      COMMON /ACTCOM/     NSOURCE,     NTRIAL,     GSPLIT,    IDPHAST,
     &                    BINL,        IB1,        IB2,
     &                    MODE,        ISPLIT,     IB1SUBS,
     &                    IDPHAS,      CKT,        ISOURCE,
     &                    ISPLITL,     IPASS,      CIR,
     &                    IDR,         CER
```

Appendix E

Data Files

Data file "buol-wate-buAc.dat"
System n Butanol + Water + n Butyl Acetate

Reacting or non reacting	0	1 for reacting
Number of species	3	
Number of potential phases	2	
Identity of phase 1	1	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	-	
Identity of phase 4	-	
Number of atomic species	3	
Temperature K	3.4315000D+02	
Pressure atm	1.0000000D+00	
Number moles of atom type 1	0.1400000D+00	
2	0.6400000D+00	
3	0.2200000D+00	
4	-	
5	-	
6	-	

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Atomic	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+0.90047	+1.15161			
Chemical	2	+3.51307	+0.00000	+5.04652			
Species	3	-0.30827	+1.75717	+0.00000			
i	4						
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Chemical	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
i	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+0.48000	+0.30000	+0.00000	+0.00000	+0.00000
Chemical	2	+0.48000	+0.00000	+0.34000	+0.00000	+0.00000	+0.00000
Species	3	+0.30000	+0.34000	+0.00000	+0.00000	+0.00000	+0.00000
i	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
Chemical	3	0.000000D+00	0.000000D+00	0.000000D+00
Species	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

G		G
	1	0.000000D+00
	2	0.000000D+00
Chemical	3	0.000000D+00
Species	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance	1.00D-05
Lower bound on molar species	1.00D-06

		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3	3	2
	4		
	5		
	6		

Appendix E

Data Files

```
      REAL*8     NSOURCE(MAXCI),         NTRIAL(2*MAXCI,MAXCI,2),
     &           GSPLIT(MAXCK)

      INTEGER    IDPHAST(2*MAXCI,2),     IDPHAS(2*MAXCK),BINL(0:MAXCI),
     &           IB1,         IB2,       MODE,
     &           ISPLIT,      IB1SUBS,   CKT,
     &           ISOURCE,     ISPLITL(2*MAXCK),
     &           IPASS,       CIR,       IDR(MAXCI),
     &           CER

      COMMON /ACTCOM/     NSOURCE,     NTRIAL,     GSPLIT,    IDPHAST,
     &                    BINL,        IB1,        IB2,
     &                    MODE,        ISPLIT,     IB1SUBS,
     &                    IDPHAS,      CKT,        ISOURCE,
     &                    ISPLITL,     IPASS,      CIR,
     &                    IDR,         CER
```

```
C       This file contains global parameters used throughout the tangent
C       plane algorithm

        INTEGER CB,CI,CK,CE,CNB,LDA,MB,NBASIC,NX

C******* P R O B L E M   S P E C I F I C   P A R A M E T E R S *********
C
C  CI                   Number of chemical types in the system
C  CK                   Number of phases postulated
C  CE                   Number of atomic species

        PARAMETER(CI=3,CK=3,CE=3,MB=CE,CB=2**6,CNB=(CI*CK-CE))


C***********************************************************************
        PARAMETER (MAXCI = 6,MAXCK = 4,MAXCE = 6)
        PARAMETER(NX=CI*CK+1)
        PARAMETER(LDA=MB,NBASIC=MB)

C       data type declarations for common variables
        DOUBLE PRECISION EPSILON, PSAT,
     &                   TAU,    GNRTL,   GRT,     TEMP,
     &                   PRESSURE     ,   NTOT,   ATOMS ,
     &                   ABASIC, SMALL,   EPSILON, EPSILON2,
     &                   FEATOL

        DOUBLE PRECISION MUZ,NOPT,GMIN

        INTEGER CKOPT

        INTEGER IBASICI ,IBASICK,
     &          INBASICI,INBASICK,IFLAGCV,CKX

        INTEGER  IPVT


        COMMON /SYSDATA/ TAU(MAXCI,MAXCI),GNRTL(MAXCI,MAXCI),
     &                   PSAT(MAXCI),GRT(CI,CK),TEMP,PRESSURE,
     &                   NTOT(MAXCI),ATOMS(MAXCE,MAXCI,MAXCK),
     &                   ABASIC(LDA,NBASIC),
     &                   IPVT(NBASIC),
     &                   IBASICI(CE),IBASICK(CE),
     &                   INBASICI(CNB),INBASICK(CNB),IFLAGCV(CK)


        COMMON /SMALLVAL/SMALL,EPSILON,EPSILON2,FEATOL

        COMMON /TPLANEDAT/ GMIN,NOPT(CI,CK),MUZ(CI),CKX,CKOPT
```

```
      PARAMETER(MAXM=CE+1,MAXN=CI*CK+1,MAXNB=MAXM+MAXN,
     &          MAXNE=MAXM*MAXN,NWCORE=5000,NNAME=1)
```

```
 Begin
    Problem number                    2
    Minimize
    Jacobian                      Dense
    Nonlinear objective variables     3
    Nonlinear jacobian variables      0
    Major iterations              50000
    Minor iterations                100
    Penalty parameter                10
    Feasibility tolerance       1.0D-14
    Function precision          1.0D-14
    Optimality tolerance        1.0D-14
    Hessian dimension                33
    Derivative level                  0
    Verify level                      1
    Summary file                     -6
    Summary frequency                 0
    Scale option                      1
    Iterations                     5000
    Debug level                       0
    Print level (jflxb)           00000
    Print frequency                   0
    Summary level                     1
    Summary frequency                 0
    Linesearch tolerance        0.00001
    Subspace tolerance              0.5
End
```

Data file "bute-wate.dat"
system: n-Butyl acetate + Water @ 298K 0.1 atm

Reacting or non reacting	0	1 for reacting
Number of species	2	
Number of potential phases	2	
Identity of phase 1	1	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	-	
Identity of phase 4	-	
Number of atomic species	2	
Temperature K	3.0000000D+02	
Pressure atm	0.1000000D+00	
Number moles of atom type 1	0.5D+00	
2	0.5D+00	
3	-	
4	-	
5	-	
6	-	

IALPHA 0 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Atomic	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+3.00498				
Chemical	2	+4.69071	+0.00000				
Species	3						
i	4						
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	1.000000	0.307940	0.000000	0.000000	0.000000	0.000000
Chemical	2	0.159040	1.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
i	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+0.	+0.	+0.00000	+0.00000	+0.00000
Chemical	2	+0.	+0.00000	+0.	+0.00000	+0.00000	+0.00000
Species	3	+0.	+0.	+0.00000	+0.00000	+0.00000	+0.00000
i	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
Chemical	3	0.000000D+00	0.000000D+00	0.000000D+00
Species	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
	1	0.000000D+00
	2	0.000000D+00
Chemical	3	0.000000D+00
Species	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance 1.00D-10
Lower bound on molar species 1.00D-10

		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3		
	4		
	5		
	6		

Data file "etOH-Acet-EtAc-Wate.dat"
System Ethanol + Acetic Acid + Ethyl-Acetate + Water

Reacting	1	
Number of chemical species	4	
Number of potential phases	2	
Identity of phase 1	0	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	0	
Identity of phase 4	0	
Number of atomic species	3	
Temperature K	3.5500000D+02	
Pressure atm	1.0000000D+00	
Number moles of atom type 1	2.0000000D+00	
2	5.0000000D+00	
3	1.5000000D+00	
4	0.0000000D+00	
5	0.0000000D+00	
6	0.0000000D+00	

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
	1	2.000000	2.000000	4.000000	0.000000	0.000000	0.000000
Atomic	2	6.000000	4.000000	8.000000	2.000000	0.000000	0.000000
Species	3	1.000000	2.000000	2.000000	1.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+1.39410	+0.67310	-0.20190	+0.00000	+0.00000
Chemical	2	-1.01820	+0.00000	+0.00700	-0.47350	+0.00000	+0.00000
Species	3	+0.16520	+0.58170	+0.00000	+1.70020	+0.00000	+0.00000
i	4	+2.17150	+1.63630	+1.92570	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Chemical	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
i	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+0.30000	+0.30000	+0.30000	+0.00000	+0.00000
Chemical	2	+0.30000	+0.00000	+0.30000	+0.30000	+0.00000	+0.00000
Species	3	+0.30000	+0.30000	+0.00000	+0.30000	+0.00000	+0.00000
i	4	+0.30000	+0.30000	+0.30000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
	1	9.956140D+00	1.440520D+03	-6.044000D+01
	2	9.684500D+00	1.644050D+03	-3.963000D+01
Chemical	3	9.222980D+00	1.238710D+03	-5.615000D+01
Species	4	1.009171D+01	1.668210D+03	-4.514000D+01
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
	1	-3.709180D+04
	2	-8.730310D+04
Chemical	3	-7.284060D+04
Species	4	-5.402340D+04
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance		1.00D-06
Lower bound on molar species		1.00D-06

		Chemical	Phase
Basic Variables	1	1	1
	2	2	1
	3	3	1
	4	4	1
	5		
	6		

Data file "etOH-etAc-wate.dat"
System Ethanol + Ethyl-Acetate + Water

Reacting or non reacting	0	1 for reacting
Number of species	3	
Number of potential phases	2	
Identity of phase 1	1	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	-	
Identity of phase 4	-	
Number of atomic species	3	
Temperature K	3.4315000D+02	
Pressure atm	1.0000000D+00	
Number moles of atom type 1	0.0400000D+00	
2	0.3000000D+00	
3	0.6600000D+00	
4	-	
5	-	
6	-	

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Atomic	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	-0.70446	-0.03940			
Chemical	2	+1.68476	+0.00000	+0.89721			
Species	3	+1.71068	+2.74214	+0.00000			
i	4						
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Chemical	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
i	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+0.10000	+0.30000	+0.00000	+0.00000	+0.00000
Chemical	2	+0.10000	+0.00000	+0.30000	+0.00000	+0.00000	+0.00000
Species	3	+0.30000	+0.30000	+0.00000	+0.00000	+0.00000	+0.00000
i	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
Chemical	3	0.000000D+00	0.000000D+00	0.000000D+00
Species	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
	1	0.000000D+00
	2	0.000000D+00
Chemical	3	0.000000D+00
Species	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance	1.00D-05
Lower bound on molar species	1.00D-06

		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3	3	2
	4		
	5		
	6		

Data file "prol-buol-wat2.dat"
System n-Propanol + n-Butanol + Water: feed 2

Reacting / Not reacting	0	1/0
Number of chemical species	3	
Number of potential phases	2	
Identity of phase 1	1	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	-	
Identity of phase 4	-	
Number of atomic species	3	
Temperature K	3.5500000D+02	
Pressure atm	1.0000000D+00	
Number moles of atom type 1	0.0400000D+00	
2	0.1600000D+00	
3	0.8000000D+00	
4	-	
5	-	
6	-	

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S

CHEMICAL SPECIES

		1	2	3	4	5	6
Atomic Species i	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
	3	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S

Chemical Species j

		1	2	3	4	5	6
Chemical Species i	1	+0.00000	-0.61259	-0.07149			
	2	+0.71640	+0.00000	+0.90047			
	3	+2.74250	+3.51307	+0.00000			
	4						
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S

Chemical Species j

		1	2	3	4	5	6
Chemical Species i	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S

Chemical Species j

		1	2	3	4	5	6
Chemical Species i	1	+0.00000	+0.30000	+0.30000	+0.00000	+0.00000	+0.00000
	2	+0.30000	+0.00000	+0.48000	+0.00000	+0.00000	+0.00000
	3	+0.30000	+0.48000	+0.00000	+0.00000	+0.00000	+0.00000
	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
Chemical Species	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
	3	0.000000D+00	0.000000D+00	0.000000D+00
	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
Chemical Species	1	0.000000D+00
	2	0.000000D+00
	3	0.000000D+00
	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance	1.00D-06
Lower bound on molar species	1.00D-10

		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3	3	2
	4		
	5		
	6		

Data file "tolu-wate-anil.dat"
System Toluene + Water + Aniline

Reacting / Not reacting		0
Number of chemical species		3
Number of potential phases		2
Identity of phase 1		1
Identity of phase 2		1
Identity of phase 3		-
Identity of phase 4		-
Number of atomic species		3
Temperature K		2.9800000D+02
Pressure atm		1.0000000D+00
Number moles of atom type	1	0.2995000D+00
	2	0.1998000D+00
	3	0.4994000D+00
	4	-
	5	-
	6	-

0 = ideal gas
1 = NRTL

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
Atomic Species	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
	3	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
Chemical Species i	1	+0.00000	+4.93035	+1.59806			
	2	+7.77063	+0.00000	+4.18462			
	3	+0.03509	+1.27932	+0.00000			
	4						
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
Chemical Species i	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
Chemical Species i	1	+0.00000	+0.24850	+0.30000	+0.00000	+0.00000	+0.00000
	2	+0.24850	+0.00000	+0.34120	+0.00000	+0.00000	+0.00000
	3	+0.30000	+0.34120	+0.00000	+0.00000	+0.00000	+0.00000
	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
Chemical Species	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
	3	0.000000D+00	0.000000D+00	0.000000D+00
	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
Chemical Species	1	0.000000D+00
	2	0.000000D+00
	3	0.000000D+00
	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance	1.00D-10
Lower bound on molar species	1.00D-10

		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3	3	2
	4		
	5		
	6		

Data file "wat-MTBE-TMP.dat"
System water + tert-methyl-butyl ether + 2,2,4-trimethylpentane at 25 C

Reacting or non reacting	0	1 for reacting
Number of species	3	
Number of potential phases	2	
Identity of phase 1	1	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	-	
Identity of phase 4	-	
Number of atomic species	3	
Temperature K	298.15	
Pressure atm	1.0000000D+00	
Number moles of atom type 1	0.0400000D+00	
2	0.3000000D+00	
3	0.6600000D+00	
4	-	
5	-	
6	-	

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Atomic	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	3.576052	4.367265			
Chemical	2	1.458662	+0.00000	-0.03824			
Species	3	5.068254	-2.30455	0.00000			
i	4						
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Chemical	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
i	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+0.20000	+0.20000	+0.00000	+0.00000	+0.00000
Chemical	2	+0.20000	+0.00000	+0.20000	+0.00000	+0.00000	+0.00000
Species	3	+0.20000	+0.20000	+0.00000	+0.00000	+0.00000	+0.00000
i	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
Chemical	3	0.000000D+00	0.000000D+00	0.000000D+00
Species	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
	1	0.000000D+00
	2	0.000000D+00
Chemical	3	0.000000D+00
Species	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance	1.00D-05
Lower bound on molar species	1.00D-06

		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3	3	2
	4		
	5		
	6		

Data file "wat-TAME-TMP.dat"
System water + tert-amyl alcohol + 2,2,4-trimethylpentane at 25 C

Reacting or non reacting	0	1 for reacting
Number of species	3	
Number of potential phases	2	
Identity of phase 1	1	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	-	
Identity of phase 4	-	
Number of atomic species	3	
Temperature K	298.15	
Pressure atm	1.0000000D+00	
Number moles of atom type 1	0.0400000D+00	
2	0.3000000D+00	
3	0.6600000D+00	
4	-	
5	-	
6	-	

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Atomic	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	4.466544	3.884622			
Chemical	2	2.248533	+0.00000	-1.20543			
Species	3	4.733523	-0.41355	+0.00000			
i	4						
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Chemical	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
i	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+0.20000	+0.20000	+0.00000	+0.00000	+0.00000
Chemical	2	+0.20000	+0.00000	+0.20000	+0.00000	+0.00000	+0.00000
Species	3	+0.20000	+0.20000	+0.00000	+0.00000	+0.00000	+0.00000
i	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
Chemical	3	0.000000D+00	0.000000D+00	0.000000D+00
Species	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
	1	0.000000D+00
	2	0.000000D+00
Chemical	3	0.000000D+00
Species	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance		1.00D-05	
Lower bound on molar species		1.00D-06	
		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3	3	2
	4		
	5		
	6		

Data file "wat-TAOH-TMP.dat"
System water + tert-amyl alcohol + 2,2,4-trimethylpentane at 25 C

Reacting or non reacting	0	1 for reacting
Number of species	3	
Number of potential phases	2	
Identity of phase 1	1	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	-	
Identity of phase 4	-	
Number of atomic species	3	
Temperature K	298.15	
Pressure atm	1.0000000D+00	
Number moles of atom type 1	0.0400000D+00	
2	0.3000000D+00	
3	0.6600000D+00	
4	-	
5	-	
6	-	

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Atomic	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	4.55106	5.533792			
Chemical	2	-1.02599	+0.00000	0.148918			
Species	3	4.55140	-1.04310	0.000000			
i	4						
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Chemical	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
i	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	+0.20000	+0.20000	+0.00000	+0.00000	+0.00000
Chemical	2	+0.20000	+0.00000	+0.20000	+0.00000	+0.00000	+0.00000
Species	3	+0.20000	+0.20000	+0.00000	+0.00000	+0.00000	+0.00000
i	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
Chemical	3	0.000000D+00	0.000000D+00	0.000000D+00
Species	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
	1	0.000000D+00
	2	0.000000D+00
Chemical	3	0.000000D+00
Species	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance		1.00D-05	
Lower bound on molar species		1.00D-06	
		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3	3	2
	4		
	5		
	6		

Data file "wat-etOH-TMP.dat"
System water + ethanol + 2,2,4-trimethylpentane at 25 C

Reacting or non reacting	0	1 for reacting
Number of species	3	
Number of potential phases	2	
Identity of phase 1	1	0 = ideal gas
Identity of phase 2	1	1 = NRTL
Identity of phase 3	-	
Identity of phase 4	-	
Number of atomic species	3	
Temperature K	298.15	
Pressure atm	1.0000000D+00	
Number moles of atom type 1	0.0400000D+00	
2	0.3000000D+00	
3	0.6600000D+00	
4	-	
5	-	
6	-	

IALPHA 1 1 if Alpha is supplied

S P E C I E S S U B S C R I P T S
CHEMICAL SPECIES

		1	2	3	4	5	6
	1	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Atomic	2	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

T A U (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	+0.00000	-1.99329	7.737716			
Chemical	2	-2.45615	+0.00000	1.056851			
Species	3	2.75164	0.48533	0.00000			
i	4						
	5						
	6						

G N R T L (I , J) V A L U E S
Chemical Species j

		1	2	3	4	5	6
	1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Chemical	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Species	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
i	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

A L P H A (I , J) V A L U E S
Chemical Species j

	1	2	3	4	5	6

	1	+0.00000	+0.20000	+0.20000	+0.00000	+0.00000	+0.00000
Chemical	2	+0.20000	+0.00000	+0.20000	+0.00000	+0.00000	+0.00000
Species	3	+0.20000	+0.20000	+0.00000	+0.00000	+0.00000	+0.00000
i	4	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	5	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000
	6	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000	+0.00000

A N T O I N E C O N S T A N T S

		A	B	C
	1	0.000000D+00	0.000000D+00	0.000000D+00
	2	0.000000D+00	0.000000D+00	0.000000D+00
Chemical	3	0.000000D+00	0.000000D+00	0.000000D+00
Species	4	0.000000D+00	0.000000D+00	0.000000D+00
	5	0.000000D+00	0.000000D+00	0.000000D+00
	6	0.000000D+00	0.000000D+00	0.000000D+00

G G A S P H A S E G I B B S E N E R G Y

		G
	1	0.000000D+00
	2	0.000000D+00
Chemical	3	0.000000D+00
Species	4	0.000000D+00
	5	0.000000D+00
	6	0.000000D+00

Convergence tolerance	1.00D-05
Lower bound on molar species	1.00D-06

		Chemical	Phase
Basic Variables	1	1	2
	2	2	2
	3	3	2
	4		
	5		
	6		