UNIVERSIDADE FEDERAL DE MINAS GERAIS

MÁRCIO RIBEIRO VIANNA NETO

CHEMICAL AND PHASE EQUILIBRIA THROUGH DETERMINISTIC GLOBAL OPTIMIZATION

BELO HORIZONTE

2016

MÁRCIO RIBEIRO VIANNA NETO

CHEMICAL AND PHASE EQUILIBRIA THROUGH DETERMINISTIC GLOBAL OPTIMIZATION

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia Química como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA QUÍMICA.

Orientador: Prof. Dr. Éder Domingos de Oliveira

Belo Horizonte

2016

To my parents, Marcos Vianna and Rita Vianna, to whom I owe all that is best in me.

ACKNOWLEDGEMENTS

I would like to thank my parents, Marcos and Rita, for all their love, patience and never-ending support.

My good friend Bernardo who has always been there and supported all my endeavors.

Also, my thanks to Professor Éder Domingos for his continuous support, for all his trust, for all the freedom that he has given me during my studies, and lastly, for his seemingly endless patience.

My thanks to Professor Marcelo Cardoso for his friendship, for all the opportunities with which he has provided me and for his invaluable advice.

My thanks to all my colleagues at the Industrial Processes Laboratory for being such an engaging, welcoming and supportive team.

My dearest thanks to all the people who have indirectly contributed to this work, either by giving me bits of advice or by keeping me sane.

My many thanks to Professor Ívina Paula for being my first chemistry teacher, and also Professor Miriam Miranda for encouraging me to do research.

Lastly, I would like to thank CNPq for their financial support.

“Valeu a pena? Tudo vale a pena

Se a alma não é pequena

Quem quer passar além do Bojador

Tem que passar além da dor

Deus ao mar o perigo e o abismo deu,

Mas nele é que espelhou o céu”

Fernando Pessoa

ABSTRACT

Chemical and phase equilibrium calculations are commonly performed by solving a constrained optimization problem known as Gibbs energy minimization. This problem is, in general, nonconvex, which implies that it is not a trivial task to solve for its global minimum, as many local minima may exist. The global minimum is the only solution that bears physical significance. Among the various techniques found in the literature that attempt to solve this problem, the αBB algorithm with interval analysis seems particularly interesting due to its generality and to the fact that it mathematically guarantees global optimality. However, in order to apply it directly to the equilibrium problem, it is necessary to circumvent somehow the fact that in its original formulation, lower bounds for mole numbers that are too close to zero may cause numerical underflow, leading the algorithm to fail. An algorithm based on the original αBB is presented and is used to evaluate 8 benchmark equilibrium problems extracted form the literature. The algorithm, despite no longer being able to mathematically guarantee global optimality, was capable of solving all problems correctly and with relative efficiency.

RESUMO

Cálculos de equilíbrio químico e de fases são comumente realizados através da resolução de um problema de otimização restrita conhecido como minimização da energia de Gibbs. O problema é, em geral, não-convexo, o que faz com que a busca pelo mínimo global não seja trivial, já que podem existir vários mínimos locais. O mínimo global é a única solução que tem significado físico. Dentre as várias técnicas encontradas na literatura, o algoritmo αBB com análise de intervalos parece ser particularmente interessante devido à sua generalidade e ao fato de que ele é capaz de garantir matematicamente que o mínimo encontrado será o mínimo global. Apesar disso, para que seja possível aplicá-lo diretamente ao problema de equilíbrio, é necessário contornar de alguma forma o fato de que, em sua formulação original, cotas inferiores muito próximas de zero podem causar *underflow* numérico, fazendo com que o algoritmo não seja bem-sucedido. Um algoritmo baseado no αBB original é apresentado e usado para resolver 8 problemas-teste de equilíbrio extraídos da literatura. O algoritmo, apesar de não mais garantir matematicamente que o ótimo global é alcançado, foi capaz de corretamente resolver todos os problemas com relativa eficiência.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Symbol	Unit	Property
$\| . \|$	-	Euclidean norm
$[A]$	-	Interval matrix.
A_e	-	Total number of moles of atoms of e present in the system.
$a_{e,i}$	-	Number of atoms of the element e present in the i-th component.
$\underline{a_{ij}}$	-	Infimum of interval a_{ij}.
$\overline{a_{ij}}$	-	Supremum of interval a_{ij}.
$B(x_0, R)$	-	Open ball centered in x_0, with radius R
C	-	Set of components present in a system.
$\mathrm{conv}[G(\boldsymbol{x})]$	-	Convex hull of $G(\boldsymbol{x})$.
$CT(\boldsymbol{x})$	-	Convex term.
$\underline{f^*}$	-	Best lower bound found so far.
$\overline{f^*}$	-	Best upper bound found so far.
$\underline{f^0}$	-	Initial lower bound.
$\overline{f^0}$	-	Initial upper bound.
$\bar{f}$	-	Dual function of f.
$\underline{f^i}$	-	Lower bound corresponding to the *i*-th node.
$\overline{f^i}$	-	Upper bound corresponding to the *i*-th node.
f^{dual}	-	Dual function corresponding to f.
$\hat{g}$	J	Transformed total Gibbs energy.
G	J	Total Gibbs energy of the system.
G_i^0	J	Gibbs energy of formation of the *i*-th component.
$G^E(\boldsymbol{n})$	J	Excess Gibbs energy.
H_f	-	Hessian matrix of the function f.
inf X	-	Infimum of a set X
K_{eq}	-	Equilibrium constant.
$LT(\boldsymbol{x})$	-	Linear term.
$\boldsymbol{N}$	-	Stoichiometric coefficients matrix.
$\boldsymbol{n_F}$	mol	Vector of number of moles of each component in feed.

$NT_i(\boldsymbol{x})$	-	General nonlinear term.
$\widehat{n_i}$	mol	Modified mole number of the i-th component.
$n_{i,k}$	mol	Number of moles of the i-th component present in the k-th phase.
$\boldsymbol{n}_{ref}$	mol	Vector of number of moles of the reference components.
P	Pa, bar, atm	Pressure.
P_i^{sat}	Pa, bar, atm	Saturation pressure of the *i*-th component.
Q_i	-	Relative surface area of component *i*.
$\mathbb{R}$	-	Set of real numbers
$\mathbb{R}^{mxn}$	-	Set of all real-valued *mxn* matrices
$\mathbb{R}^n$	-	Set of n-dimensional real vectors
$R_{u,i}$	-	Relative Van der Waals radius of component *i*.
sup X	-	Supremum of a set X
T	K, °C	Temperature.
$u_{a,b}$	J/mol	Interaction energies of components *a* and *b*
$UT_i(x^i)$	-	Univariate concave term.
V_a	m³/mol	Molal volume of component *a*
X_i	-	Transformed mole fraction of the i-th component.
$x_{i,k}$	-	Mole fraction of the i-th component in the k-th phase.
$\boldsymbol{x}^L$	-	Vector of lower bounds.
$\boldsymbol{x}^U$	-	Vector of upper bounds.
Δ	-	Diagonal shift matrix.
Δg		Dimensionless Gibbs energy of mixing.
ΔG^0_{rxs}	J	Gibbs energy of reaction.
$\Delta \hat{g}_{mix}$	J	Dimensionless transformed Gibbs energy of mixing.
Δg_{mix}	J	Dimensionless Gibbs energy of mixing.
$\nabla^2 f$	-	Hessian of the function f
∇f	-	Gradient of the function f
$\Lambda_{a,b}$		Wilson model parameter Λ.
Π	-	Set of phases present in a system.
Ψ		Stability function.
$\alpha_{a,b}$		Nonrandomness parameter of components *a* and *b.*
α_{max}		Maximum allowable value for α.
$\gamma_{i,k}$	-	Activity coefficient of the i-th component in the k-th phase.

ϵ	-	Error tolerance.
θ_i	-	UNIQUAC parameter θ corresponding to the *i*-th component.
λ_i	-	Lagrange multipliers associated to the *i*-th equality constraint.
$\lambda_{min}(A)$	-	Minimum eigenvalue of the interval matrix A.
$\mu_{i,k}^0$	J/mol	Standard chemical potential of formation of the i-th component in the k-th phase.
μ_i	J/mol	Lagrange multipliers associated to the *i*-th inequality constraint.
$\mu_{i,k}$	J/mol	Chemical potential of the i-th component in the k-th phase.
ν_i	-	Stoichiometric coefficient of the i-th component in a given reaction.
$\boldsymbol{\nu}_i$	-	Vector of stoichiometric coefficients of component *i* in each of the independent reactions.
ξ	-	Degree of advancement.
$\tau_{a,b}$	-	Dimensionless interaction energies of components *a* and *b*
ϕ_i	-	UNIQUAC parameter ϕ corresponding to the *i*-th component.

1 INTRODUCTION

The study of chemical and phase equilibria is of paramount importance in chemical engineering, as it constitutes the fundamental principle on which several unit processes are based. Among such unit processes are fractional distillation, flash distillation, absorption, adsorption, leaching, precipitation and extraction. It is also the principle upon which some chemical reactors are designed. Chemical and phase equilibria studies also play a big role in water quality control (Tchobanoglous & Schroeder, 1987), water chemistry (Brezonik & Arnold, 2011) and the prediction and control of scaling in pipelines. It is thus clear that a solid understanding of both the practical and theoretical aspects of chemical equilibria is extremely important.

From a mathematical standpoint, a chemical and phase equilibrium calculation problem can be formulated as a nonlinear constrained optimization problem (Floudas, 2000), whose objective function is the so-called total Gibbs energy of that particular system under study, and the variables are the mole numbers of each component in that same system. The constraints are all linear and account for mass balances. Solving the equilibrium problem thus corresponds to finding the system composition that respects mass balance constraints and minimizes the Gibbs energy.

The literature reveals that several attempts have been made to effectively solve these problems. These attempts branch out in stochastic (or probabilistic) methods and in deterministic methods. Among the deterministic methods, the so-called αBB method (Floudas, 2000) has received much attention as it is capable of deterministically solving for the global optimum of twice-differentiable optimization problems. It also mathematically guarantees global optimality, which is highly desirable.

It is very important that the global minimum be found, as other local solutions do not correspond to physical solutions. The objective function is highly nonlinear and, in general, nonconvex. In practice, that means that these calculations may be very hard to perform, as most local optimization solvers will be at risk of converging to local minima, which would then yield a solution that bears no physical meaning. This remark makes global deterministic algorithms very appealing, as they can provide guarantees of optimality, which stochastic algorithms can not.

The αBB algorithm is based on systematically scanning the objective function's domain for the global optimum by successively splitting it into smaller subdomains. Within each subdomain, the

original problem (which provides an upper bound to the global minimum) and a convex underestimating problem (which provides a lower bound) are locally solved and its solutions are stored. As the iterations progress, the subdomains become smaller and the gap between the best upper and lower bounds tighten. When this gap is small enough, the algorithm halts, and the optimum point is returned.

The underestimating functions may be either calculated (Floudas, 2000) analytically or through interval analysis (Moore, et al., 2009). The latter method has the advantage of being very general, although its underestimating functions tend to be very loose, i.e. they underestimate the original function far too much, which causes the algorithm to be much less efficient. In fact, the underestimating function may evaluate to such large negative numbers that it causes local solvers to fail due to numerical underflow.

In order to address this particular issue, the original αBB was modified in such a way that the underestimators were prevented from becoming too negative. The modified algorithm, in practice, correctly computes chemical equilibria faster than would be the case if the original αBB were used. The tradeoff is the fact that it no longer mathematically guarantees global optimality, although in practice, it performs well.

In order to evaluate its performance, the algorithm's performance was assessed in eight benchmark chemical and phase equilibrium problems (whose global optima were known), extracted from the work of Bonilla-Petricolet and collaborators (2011).

The main objective of this work is to evaluate how well the adapted algorithm behaves when applied to equilibrium problems. More specifically, this work aims at a) checking whether the algorithm converges to the global optimum, b) measuring how many iterations it takes for it to converge and c) gaining insight into how much the underestimators calculated by the algorithm distance themselves from the original function.

2 FOUNDATIONS

2.1 MATHEMATICAL DEFINITIONS

2.1.1 Set theory and topology

Let *X* be a set. If *X* has a finite number of elements, the symbol $|X|$ denotes the number of elements of *X*.

Let $X \subset \mathbb{R}$. A lower bound on X is a real number a such that $a \leq x$ for every $x \in X$. Conversely, an upper bound on X is a real number b such that $b \geq x$ for every $x \in X$.

The infimum of a set X, denoted by $\inf X$ is the greatest lower bound on X. The supremum of a set X, denoted by $\sup X$ is the smallest upper bound on X. Neither the infimum, nor the supremum of a set need to belong to it. As an example, consider the set $X = \{x \mid x > 2 \text{ and } x < 3\}$. Clearly, $\inf X = 2$ and $\sup X = 3$, but $2 \notin X$ and $3 \notin X$ (Munkres, 2000).

An open ball centered in x_0 with radius R is a subset of $\mathbb{R}^n$ defined as $B(x_0, R) = \{x \mid \|x - x_0\| < R\}$, where $\| . \|$ is the normal Euclidean distance. A set $X \subset \mathbb{R}^n$ is said to be bounded if there exists an open ball $B(x_0, R)$ such that $X \subset B(x_0, R)$.

A set $X \subset \mathbb{R}^n$ is said to be open if for every point x_0 in X there exists an open ball centered in x_0, with radius $R > 0$ such that $B(x_0, R) \subset X$. A set $X \subset \mathbb{R}^n$ is closed if its complement $\mathbb{R}^n - X$ is open. A set X that is both closed and bounded is said to be compact.

A more comprehensive treatment of the above definitions can be found in standard references on analysis and on topology, such as (Rudin, 1976) and (Munkres, 2000).

2.1.2 Linear algebra

Matrices will be denoted by uppercase letters, such as *A*. The set of real *mxn* matrices (that is, m rows and n columns) will be denoted by $\mathbb{R}^{mxn}$. We may also represent a matrix by its elements: $A = [a_{i,j}]$. The symbol $a_{i,j}$ denote the element of A in row *i* and column *j*.

Vectors are single-column matrices and will be denoted by lowercase bold letters, such as $\boldsymbol{b}$. The set of real *m*-vectors (vectors with *m* components) will be denoted by $\mathbb{R}^m$.

The transpose of a matrix, A^T is the matrix whose columns are the rows of A. Equivalently, if $A = [a_{i,j}]$, then $A^T = [a_{j,i}]$. A matrix whose transpose is equal to itself is said to be symmetric.

A matrix $A \in \mathbb{R}^{nxn}$ is said to be positive semi-definite if it is such that, for every vector $\boldsymbol{x} \in \mathbb{R}^n$, we have $\boldsymbol{x^T} A\boldsymbol{x} \geq 0$. It is said to be positive definite if strict inequality holds. If the signs are reversed, then it is said to be negative semi-definite and negative definite, respectively.

Given a matrix A and a vector $\boldsymbol{v}$, we say that $\boldsymbol{v}$ is an eigenvector of A if it is such that $A\boldsymbol{v} = \lambda\boldsymbol{v}$ for some real number λ. The number λ is then said to be an eigenvalue of A associated to the eigenvector $\boldsymbol{v}$ (Lax, 2007).

2.1.3 Multivariable calculus

Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be a real-valued function. The gradient ∇f is an n-dimensional vector whose components are the partial derivatives of f with respect to each variable: $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$. The gradient vector always points in the direction along which the function increases the most.

The hessian $\nabla^2 f$ or $H[f]$ is an *nxn* matrix composed of the second derivatives of f. More precisely, $\nabla^2 f = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]$ (Luenberger & Ye, 2010).

2.1.4 Convex analysis

A set X is said to be convex if for every pair of points $x_0, x_1 \in X$ and for every $\theta \in [0,1]$, the point $\theta x_0 + (1-\theta)x_1$ also belongs to X.

Let $f: D \rightarrow \mathbb{R}$ be a function. We say that f is convex if D is a convex set and for every pair of points $d_0, d_1 \in D$, we have that $f[\theta d_0 + (1-\theta)d_1] \leq \theta f(d_0) + (1-\theta)f(d_1)$. If this inequality holds strictly, we say that f is strictly convex. A concave function is a function such that $-f$ is convex (Boyd & Vandenberghe, 2004).

A function f is convex if and only if its hessian is positive semi-definite. Alternatively, a function is concave if and only if its hessian is negative semi-definite (Boyd & Vandenberghe, 2004).

A function f is convex if and only if all of its eigenvalues are positive.

2.2 CHEMICAL AND PHASE EQUILIBRIA

2.2.1 Problem formulation

Let S be a closed system (no mass transfer allowed through its walls) and C be a list of chemical species. The set of all elements that form the chemical species of C will be denoted by A. We assume that S is initially composed of $n_{0,i}$ moles of the i-th component, where $i \in C$. Lastly, we also impose the system be both isothermic and isobaric.

Once the equilibrium state is achieved, the system may display more than one phase, and each phase will exhibit, in general, a different composition. The set of phases in equilibrium will be denoted by Π and the mole number of the i-th component present in the k-th phase (in equilibrium) will be written as $n_{i,k}$.

The problem to which we will refer to as *the chemical and phase equilibrium problem* consists on finding $n_{i,k}$ given the values of $n_{0,i}$, temperature, T, and pressure, P (Abbott, et al., 2001).

Classical thermodynamics provides the theoretical framework necessary to perform such calculations. It indicates that the equilibrium state is achieved when the total Gibbs energy (G) of the system attains its minimum, which in turn characterizes the problem as an optimization problem (Bonilla-Petricolet, et al., 2011).

Mathematically, the function to be minimized can be written as:

$$G = \sum_{k \epsilon \Pi} \sum_{i \in C} n_{i,k} \mu_{i,k} \tag{1}$$

In this expression $\mu_{i,k}$ is the chemical potential of the i-th component in the k-th phase and G is the system total Gibbs energy (Abbott, et al., 2001).

As the system S was assumed to be closed, it is necessary to find a set of values that not only minimizes G, but that is also such that mass conservation holds. It is possible to achieve that by imposing a set of constraints on the optimization problem.

If mass conservation must hold true, then it must also be the case that the total amounts of atoms of each element remain constant (we assume that no nuclear processes take place). In order to construct these constraints, we follow the same process outlined in (Abbott, et al., 2001). For each element $e \in A$ we define the quantity A_e as the total number of moles of atoms of e present in the system. A_e are constants and are functions of $n_{0,i}$. It is also convenient to define the quantity $a_{e,i}$ to be the number of atoms of the element e present in the i-th component.

From the initial composition of the system, the values of A_e can be calculated as follows (Abbott, et al., 2001):

$$A_e = \sum_{i \in C} a_{e,i} n_{0,i} \tag{2}$$

Once these are found, we may finally write the mass balance constraints:

$$A_e = \sum_{k \in \Pi} \sum_{i \in C} a_{e,i} n_{i,k} \tag{3}$$

Since negative mole numbers are physically meaningless, we must also make sure that:

$$n_{i,k} \geq 0 \tag{4}$$

A key remark to be made is that all constraints are *linear*. A more concise notation ensues. Upon denoting by $\boldsymbol{n}$ the vector composed by the variables $n_{i,k}$, by $\boldsymbol{z}$ the vector of the constants A_e and by H the matrix of $a_{e,i}$, all constraints may be rewritten as:

$$\begin{aligned} H\boldsymbol{n} &= \boldsymbol{z} \\ \boldsymbol{n} &\geq 0 \end{aligned} \tag{5}$$

It is still necessary to establish how the chemical potentials are to be calculated. We may write the chemical potential for every component as:

$$\mu_{i,k} = \mu_{i,k}^{0} + RT \ln\left(c_{i,k} x_{i,k}\right) \tag{6}$$

Where $\mu_{i,k}^{0}$ is the standard chemical potential of formation of the i-th component in the k-th phase at pressure P, $x_{i,k}$ is its mole fraction and $c_{i,k}$ is given by:

- $c_{i,k} = P$ if k is an ideal gas mixture.
- $c_{i,k} = 1$ if k is an ideal liquid solution.
- $c_{i,k} = \gamma_{i,k}$ if i is a solute in a non-ideal liquid solution.

The values represented by $\gamma_{i,k}$ are the activity coefficients, and they represent the nonidealities that exist within a system. There are several models employed to calculate $\gamma_{i,k}$. These will be described later. This notation can be easily extended to incorporate non-ideal gases – instead of activity coefficients, there would be, in that case, fugacity coefficients (Tester & Modell, 1997).

Finally, by combining everything, we are left with the mathematical formulation of the chemical and phase equilibrium problem. In standard optimization notation (Luenberger & Ye, 2010):

$$\begin{aligned} &\min \sum_{k \epsilon \Pi} \sum_{i \in C} n_{i,k} \mu_{i,k} \\ &s.t. \\ &\qquad H\boldsymbol{n} = \boldsymbol{z} \\ &\qquad \boldsymbol{n} \geq 0 \end{aligned} \tag{7}$$

2.2.2 Alternative formulations

Even though the aforementioned formulation is sufficient to fully characterize equilibrium problems, one may find useful to restate it in an equivalent formulation. Reasons for that may be: the will to emphasize a particular theoretical nuance of the problem; to obtain a simpler algebraic formulation – and potentially better numerical solution schemes; or to obtain greater insight on the current problem. We now turn to some such alternative formulations.

2.2.2.1 Gibbs excess energy

Excess functions are defined as the amounts by which a state function (such as the Gibbs energy) of a solution exceeds the value it would assume in an ideal solution of the same composition (Denbigh, 1971). Therefore, we can mathematically express this definition for the Gibbs energy as:

$$G^E(\boldsymbol{n}) = G(\boldsymbol{n}) - G^{id}(\boldsymbol{n}) \tag{8}$$

Upon comparing equations (6) and (8) and differentiating the result with respect to $n_{i,k}$, It can be shown that the following relation holds (Abbott, et al., 2001):

$$\left(\frac{\partial G^E}{\partial n_{i,k}}\right)_{T,P,n_j} = RT \ln \gamma_{i,k} \tag{9}$$

In equation (9), the values of T, P and n_j, for all $j \neq i$, are kept constant. This relation shows that by knowing how to calculate the activity coefficients, it is possible to infer the values of the excess Gibbs function and vice–versa. Therefore, it is possible to describe the total Gibbs energy of a system either by the method suggested in the original formulation or by means of applying Equation (8) alongside with the definition of G^E.

2.2.2.2 Degree of advancement of a reaction

In reactive systems the number of moles of the reactants and products are interrelated. In order to express this interrelationship in a clean and unambiguous way, we define an extra variable ξ for each reaction. This variable is referred to as the degree of advancement (or simply, the advancement) of that particular reaction and it measures how much the reaction has progressed. It is defined in such a way that the following relation holds true (Denbigh, 1971):

$$\frac{dn_i}{\nu_i} = d\xi \tag{10}$$

In the preceding relation ν_i denotes the stoichiometric coefficient of component *i*, assuming that it is involved in the reaction ($\nu_i \geq 0$ if *i* is a product and $\nu_i \leq 0$ if *i* is a reactant). This definition suggests the following procedure for writing mass balances: for each component *i* present in the system, define n_i as the total number of moles of *i* in the system. Mathematically:

$$n_i = \sum_{k \in \Pi} n_{i,k} \tag{11}$$

The changes in the total number of moles of a component must come from the progression of chemical reactions. Keeping that in mind, we may further write (Denbigh, 1971):

$$n_i = n_{i,0} + \sum_{j \in R} \nu_{i,j} \xi_j \tag{12}$$

Where *R* is the set of chemical reactions occurring in the system and $\nu_{i,j}$ denotes the stoichiometric coefficient of component *i* on reaction *j*. Assuming that all chemical reactions have been properly balanced, it is unnecessary to write the atomic mass balances represented in equation (3).

It is important to notice that the number of constraints and number of variables of the equilibrium problem may change according to the formulation used. That means that the degrees of freedom of the optimization problem may change as well. As a rule of thumb, greater

degrees of freedom lead to bigger search spaces, thus making the optimum point harder to find. Let us denote by $|X|$ the number of elements in the set$X$. In the first formulation, there would be $|\Pi||C|$ variables ($n_{i,k}$) and $|A|$ equality constraints, thus that formulation leads to$|\Pi||C| - |A|$ degrees of freedom. In the degree of advancement formulation there would be $|\Pi||C| + |C| + |R| = (|\Pi| + 1)|C| + |R|$ variables ($n_{i,k}$, n_i and ξ_i) and $|C| + |C| = 2|C|$ constraints, which leads to $(|\Pi| - 1)|C| + |R|$ degrees of freedom. Even though the pure analysis of degrees of freedom is not a good predictor of how simple it is to solve an optimization problem, it should be kept in mind.

2.2.2.3 Reduction of dimensionality

It has been proposed in the literature that, from a mathematical standpoint, the number of moles and mole fractions may not be the most natural choice of independent variables to the equilibrium problem. This is due to the fact that the Gibbs' phase rule imposes constraints on these quantities and, for this reason, the numbers of moles and compositions would mask the problem's dimensionality. One such approach is the one proposed by (Ung & Doherty, 1995). This approach arises from the observation that the Gibbs' phase rule states that (Ung & Doherty, 1995):

$$F = (c + I) - R + 2 - \Pi \tag{13}$$

F denotes the degrees of freedom of a system, *c* is the number of reacting compounds, *I* is the number of inert compounds, *R* is the number of independent chemical reactions and *Π* is the number of phases. *R* compounds are chosen to be the reference compounds. A modified mole number is then defined for every other component as follows (Ung & Doherty, 1995):

$$\widehat{n_i} = n_i - \boldsymbol{\nu_i^T N^{-1} n_{ref}} \qquad i = 1, \dots, c - R \tag{14}$$

Here $\boldsymbol{n}_{ref}$ denotes the vector of number of moles of the reference components, $\boldsymbol{\nu}_i$ is the vector of stoichiometric coefficients of component *i* in each of the independent reactions, $\boldsymbol{K}_{\boldsymbol{eq}}$ is a row vector containing the chemical equilibrium constants of the independent reactions, and ***N*** is an invertible matrix whose rows are the stoichiometric coefficients of the *R* reference components in the *R* independent reactions. The independence of the reference equations assures that ***N*** is invertible, as none of its rows will be a linear combination of other (independent reactions) rows, and thus its determinant will be different from 0, which is enough to guarantee its nonsingularity.

The transformed mole fractions can then be written as (Ung & Doherty, 1995)

:

$$X_i = \frac{\widehat{n_i}}{\widehat{n_T}} = \frac{x_i - \boldsymbol{\nu_i^T} \boldsymbol{N^{-1}} \boldsymbol{x}_{ref}}{1 - \boldsymbol{\nu_{TOT}^T} \boldsymbol{N^{-1}} \boldsymbol{x}_{ref}} \qquad i = 1, \dots, c - R \tag{15}$$

where

$$\boldsymbol{\nu_{TOT}^T} = \sum_{i=1}^{c-R} \boldsymbol{\nu_i^T} \tag{16}$$

and $\boldsymbol{x}_{ref}$ denotes the vector of mole fractions of the reference components.

Introducing these new variables in the concept of Gibbs energy, we may define, for convenience:

$$\hat{g} = \frac{G}{\widehat{n_T}} \tag{17}$$

It can be shown that the following relation holds true (Wasylkiewicz & Ung, 2000) :

$$\hat{g} = \sum_{i=1}^{c-R} X_i \mu_i \tag{18}$$

In order not to have to compute the standard Gibbs free energies of the pure components, we define the Gibbs energy of mixture as (Wasylkiewicz & Ung, 2000):

$$\frac{\Delta \hat{g}_{mix}}{RT} = \frac{\hat{g} - \hat{g}^0}{RT} = \sum_{k \epsilon \Pi} \sum_{i=1}^{c-R} X_i \ln\left(\gamma_{i,k} x_{i,k}\right) \tag{19}$$

One important result is that minimizing *G* subject to the usual mass-balance constraints is equivalent to minimizing $\Delta \hat{g}_{mix}$ under the same constraints (Bonilla-Petricolet, et al., 2011). If the equilibrium constants of each reaction are known, we may write an equation in terms of it (Bonilla-Petricolet, et al., 2011):

$$\Delta \hat{g}_{mix} = \Delta g_{mix} - \sum_{k \epsilon \Pi} \ln \boldsymbol{K_{eq}} \boldsymbol{N}^{-1} \boldsymbol{n}_{ref,k} \tag{20}$$

where

$$\Delta g_{mix} = \sum_{k \epsilon \Pi} \sum_{i \in C} n_{i,k} \ln\left(\gamma_{i,k} x_{i,k}\right) \tag{21}$$

In the above equations $\boldsymbol{n}_{ref,k}$ denotes the vector of number of moles of the reference components on phase *k*.

2.2.3 Activity coefficient models

There exist several activity coefficient models, both for electrolytic and for non-electrolytic solutions (Denbigh, 1971); (Abbott, et al., 2001); (Walas, 1985). The decision of which model should be used depends highly on the system under consideration. A few of these models are described below.

2.2.3.1 Margules activity model

This model was introduced in 1895 by Max Margules and is suitable for non-electrolytic solutions. It is still commonly used and it is known to perform well for some systems (Walas, 1985). Despite its simplicity, it does possess properties that are very desirable, one of which being the fact that it is able to properly describe the behavior of activity coefficients under conditions of either extreme dilution or extreme concentration. Most modern models are not able do so. The activity coefficient given by the Margules model can be represented in the form (Bonilla-Petricolet, et al., 2011):

$$\ln \gamma_{i,k} = \frac{1}{2T} \sum_{a \in C} \sum_{b \in C} \left(A_{a,i} + A_{b,i} - A_{a,b} \right) x_a x_b \tag{22}$$

In this expression the terms $A_{i,j}$ are parameters of the model and depend on the chemical components involved. T is the system temperature.

2.2.3.2 Wilson activity model

The Wilson model was introduced in 1964. This model is based on the framework provided by statistical mechanics and it assumes that the interactions that take place between the molecules (or components) of a system depend mostly on their "local concentration" (Walas, 1985), or, more precisely, on how neighboring molecules interact. These local concentrations are expressed as probabilities that are based on the Boltzmann distribution. When written in its multicomponent form, the Wilson model is written as follows (Bonilla-Petricolet, et al., 2011):

$$\ln \gamma_{i,k} = 1 - \ln \left(\sum_{a \in C} x_a \Lambda_{i,a} \right) - \sum_{b \in C} \left(\frac{x_b \Lambda_{b,i}}{\sum_{a \in C} x_a \Lambda_{b,a}} \right) \tag{23}$$

where

$$\Lambda_{a,b} = \frac{V_b}{V_a} \exp\left(-\frac{u_{a,b}}{RT}\right) \tag{24}$$

and the quantities V_a and $u_{a,b}$ correspond respectively to the molal volume of component *a* and to the interaction energies of components *a* and *b*. These quantities are taken as parameters to the model (Walas, 1985).

The Wilson model displays suitable properties, such as the fact that it is able to describe both polar and nonpolar mixtures and its capability of representing multicomponent behavior despite the fact that it only considers binary interactions (Walas, 1985). There are, however, a few drawbacks. One of them is the fact that the model does not allow for liquid-liquid immiscibility to be implemented. Another one, which is critical for the scope of this work, is the fact that it introduces nonconvexities on the total Gibbs energy of the system. That, in turn, accounts for a greater difficulty on solving equilibrium problems involving the Wilson model (Floudas, 2000).

2.2.3.3 NRTL activity model

The NRTL (NRTL stands for *nonrandom two-liquid*) model, also known as the Renon model, assumes that the liquids involved organize themselves in a molecular level into small clusters (Walas, 1985). The derivation of the equation for this model is similar to that of the Wilson model, the main difference being that a *nonrandomness parameter* is introduced. The multicomponent form of the NRTL model is written as (Bonilla-Petricolet, et al., 2011):

$$\ln \gamma_{i,k} = \frac{\sum_{a \in C} \tau_{a,i} G_{a,i} x_a}{\sum_{a \in C} G_{a,i} x_a} + \sum_{a \in C} \frac{G_{i,a} x_a}{\sum_{b \in C} G_{i,b} x_b} \left(\tau_{i,a} - \frac{\sum_{b \in C} \tau_{b,a} G_{b,a} x_b}{\sum_{b \in C} G_{b,a} x_b} \right) \tag{25}$$

where

$$G_{a,b} = \exp\left(-\alpha_{a,b} \tau_{a,b}\right) \tag{26}$$

$$\tau_{i,j} = \frac{u_{i,j}}{RT} \tag{27}$$

and the quantities $\alpha_{a,b}$ and $u_{a,b}$ correspond respectively to the nonrandomness parameter of components *a* and *b* and to the interaction energies of components *a* and *b*. These quantities are taken as parameters to the model (Walas, 1985).

As is the case with the Wilson model, the NRTL equations introduce nonconvexities in the equilibrium problem, which often leads to computational and numerical difficulties. It should however be noted that Floudas and collaborators (2000) managed to express NRTL-equilibrium problems as a biconvex optimization problem (see the section on GOP), which greatly reduces their solution complexity. The NRTL model usually represents well binary systems and, unlike the Wilson model, it is capable of representing immiscible liquid-liquid mixtures (Walas, 1985). A practical handicap, however, that may hinder its applicability is the availability of its parameters (Walas, 1985).

2.2.3.4 UNIQUAC activity model

The UNIQUAC (universal quasichemical) model was developed by Abrams and Prausnitz in 1975. Its derivation is also based on the two-liquid model and on the idea of local concentrations (as is the case of the NRTL and Wilson models). Its most distinguishing feature is the fact that it depicts the excess Gibbs energy as composed of two parts (Walas, 1985), namely:

- Differences in shape and size between the molecules (configurational or combinatorial contribution).
- Energetic interactions between the molecules (residual contribution).

The multicomponent UNIQUAC model takes the following form (Bonilla-Petricolet, et al., 2011); (Reid, et al., 1987):

$$\ln \gamma_{i,k} = \ln \gamma_{i,k}^{E} + \ln \gamma_{i,k}^{R} \tag{28}$$

$$\ln \gamma_{i,k}^{E} = \ln \frac{\phi_i}{x_i} + 5Q_i \ln \frac{\theta_i}{\phi_i} + l_i - \frac{\phi_i}{x_i} \sum_{a \in C} x_a l_a \tag{29}$$

$$\ln \gamma_{i,k}^{R} = Q_i \left[1 - \ln\left(\sum_{a \in C} \theta_a \tau_{ai} \right) - \sum_{a \in C} \left(\frac{\theta_a \tau_{ai}}{\sum_{b \in C} \theta_b \tau_{bi}} \right) \right] \quad (30)$$

where

$$\theta_i = \frac{Q_i x_i}{\sum_{j \in C} Q_j x_j} \quad (31)$$

$$\phi_i = \frac{R_{u,i} x_i}{\sum_{j \in C} R_{u,j} x_j} \quad (32)$$

$$l_i = 5\left(R_{u,i} - Q_i\right) - \left(R_{u,i} - 1\right) \quad (33)$$

$$\tau_{i,j} = \exp\left(-\frac{u_{i,j}}{RT}\right) \quad (34)$$

The quantities $R_{u,i}$, Q_i and $u_{i,j}$ are taken as parameters to the model and they correspond to, respectively, the relative Van der Waals radius of component *i*, surface area of component *i* and the binary interaction energy between components *i* and *j*.

The UNIQUAC model is capable of representing well the behavior of mixtures of components whose sizes are quite different (Walas, 1985). It is also capable of describing liquid-liquid equilibria, much like the NRTL model. The difficulty of obtaining parameters may be one of its biggest drawbacks. Not only that, but its complex algebraic structure may introduce severe nonconvexities on the equilibrium problem which greatly increases its computational and numerical difficulty. In some situations, simpler activity models can describe mixtures better than UNIQUAC, leading to simpler problem formulations and potentially easier computational solution schemes (Walas, 1985).

2.3 MATHEMATICAL OPTIMIZATION

2.3.1 Basic definitions

2.3.1.1 Optimality and feasibility

The general mathematical optimization problem is stated in standard notation as follows (Luenberger & Ye, 2010):

$$\begin{aligned} &\min \quad f(\boldsymbol{x}) \\ &s.t. \\ &\qquad h_i(\boldsymbol{x}) = 0, \qquad i = 1 \dots m \\ &\qquad g_i(\boldsymbol{x}) \leq 0, \qquad i = 1 \dots r \\ &\qquad \boldsymbol{x} \in S \end{aligned} \tag{35}$$

In this formulation, $\boldsymbol{x} = [\boldsymbol{x_1}, \dots, \boldsymbol{x_n}]^{\boldsymbol{T}}$ is a vector of *n* variables, often called decision variables, f is the function to be minimized, often referred to as objective function or cost function, h_i are the equality constraints, g_i are the inequality constraints and S is a subset of an n-dimensional space. For the scope of this work, we will always assume that $S = \mathbb{R}^n$ and, for that reason, we will abstain from restating it. Throughout this work, the terms *mathematical optimization* and *mathematical programming* will be used interchangeably. If no constraints are present in the problem, we say that it is an *unconstrained optimization problem*, otherwise, we call it a *constrained optimization problem*.

The general mathematical optimization problem consists on finding a point $\boldsymbol{x}^*$such that *a)* all constraints are satisfied, and *b)* $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for every point $\boldsymbol{x}$ that also obeys the same set of constraints. This motivates the following definition:

Definition: A point $\boldsymbol{x}$ that satisfies all constraints of an optimization problem is said to be *feasible*. (1)

In other words, the general mathematical optimization problem consists on finding a feasible point $\boldsymbol{x}^*$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$ for every other feasible $\boldsymbol{y}$. The set of all feasible points is called

the domain of the optimization problem and is represented by D. The optimum value of a function can be defined more rigorously as follows:

Definition: The optimal value $\boldsymbol{p}^*$ of an optimization problem is the infimum of f over the set of feasible points: $\boldsymbol{p}^* = \inf\{f(x) \mid x \in D\}$ (2)

If $D \neq \emptyset$, we say that the problem is *feasible*, otherwise, it is said to be *infeasible*. As in (Boyd & Vandenberghe, 2004) we allow $\boldsymbol{p}^*$ to take the extended values $\pm\infty$ and adopt the convention that, if $D = \emptyset$, then $\boldsymbol{p}^* = \infty$. If there is a sequence of feasible points $\{\boldsymbol{x_k}\}$ such that $f(\boldsymbol{x_k}) \to -\infty$ as $k \to \infty$, then $\boldsymbol{p}^* = -\infty$ and we say that the problem is *unbounded below* (Boyd & Vandenberghe, 2004).

It is sometimes necessary to check if a feasible solution exists. The problem of checking if a feasible solution exists (and finding it) is called the *feasibility problem* (Boyd & Vandenberghe, 2004). It can be stated as:

$$\begin{aligned} &\text{find} \quad \boldsymbol{x} \\ &s.t. \\ &\qquad h_i(\boldsymbol{x}) = 0, \qquad i = 1 \dots m \\ &\qquad g_i(\boldsymbol{x}) \leq 0, \qquad i = 1 \dots r \\ &\qquad \boldsymbol{x} \in S \end{aligned} \tag{36}$$

2.3.1.2 Global and local optimality

In practice, many optimization problems are numerically solved by means of iterative algorithms. Many of these algorithms take as an input a starting point $\boldsymbol{x_0}$ and, from it, generate a sequence of points $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_k}\}$ that yield progressively better solutions. That brings up two important issues:

- It is necessary to establish a stopping criterion for the algorithm, as we cannot run an algorithm indefinitely. In other words, we must better specify what it means for a solution to be good enough.
- Some of these algorithms cannot guarantee that the solution point is such that it solves problem (35), i.e., that $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ for every other feasible $\boldsymbol{y}$. Despite the fact that the

solution point obtained may not solve the optimization problem, it is often the case that it is a minimum in the sense that its value is smaller than that of any of its closest neighbors.

The second issue can be addressed by introducing the concept of a *local minimum* (Floudas, 2000):

Definition: Let $\boldsymbol{x}^* \in D$. If there exists an $\varepsilon > 0$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for every $\boldsymbol{x} \in D$ for which $\|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon$. Then $\boldsymbol{x}^*$ is a local minimum. (3)

In order for a point to be the solution of an optimization problem it is necessary that it be feasible and that no other feasible point evaluate to a smaller value than it. Notice that it is in general not true that a local minimum is also the solution to the problem, but it is true that the solution is a local minimum. In order to contrast the idea of local minima with the solution to the problem, we shall refer to the latter as the *global minimum*:

Definition: Let $\boldsymbol{x}^* \in D$. If $f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$ for every other $\boldsymbol{y} \in D$, then $\boldsymbol{x}^*$ is a global minimum. (4)

Figure 1 depicts the graph of a function displaying one local minimum ($\boldsymbol{x_1}$) and one global minimum ($\boldsymbol{x_2}$).

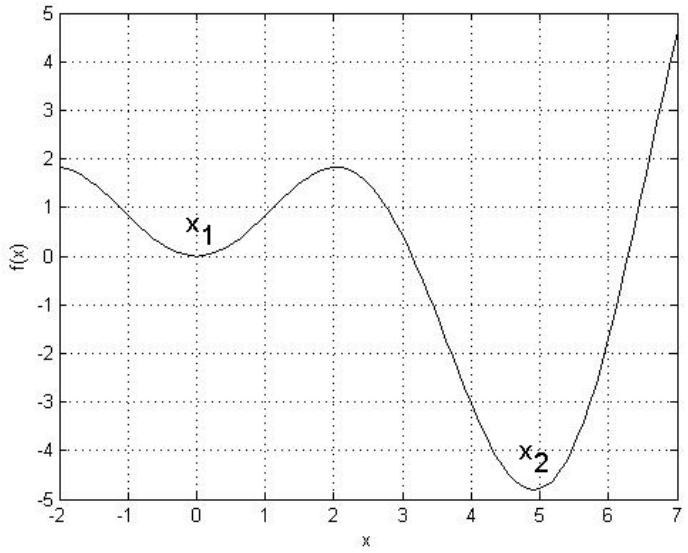


Figure 1- Function exhibiting one global minimum (x_2) and one local minimum (x_1).

The first issue brings forth the definition of ε–optimality (Floudas, 2000):

Definition: Let $\boldsymbol{x}^* \in D$ and let $\varepsilon \geq 0$ be a predetermined tolerance. If $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) + \varepsilon$ for every $\boldsymbol{x} \in D$, then $\boldsymbol{x}^*$ is an ε –global minimum. (5)

2.3.1.3 Categories of optimization problems

In the general definition of an optimization problem, no restrictions were made with respect to the cost function and the constraint functions. It is extremely difficult to devise a unified approach capable of solving the problem for every possible function and, for that reason, it is customary to divide the optimization problem into different categories. It turns out that it is possible to deterministically solve problems pertaining to some of these classes very quickly and accurately. Even though it might seem that, by doing so, we are restricting ourselves to only being able to solve a relatively small set of problems, it turns out, as will be seen later, that these easier problems are the building blocks for algorithms meant to solve more generic problems. Some of the most common categories of optimization problem are briefly explained next.

2.3.1.3.1 Least-squares

We say that an optimization problem is a least-squares problem if it is of the form:

$$\min \quad \|A\boldsymbol{x} - \boldsymbol{b}\| \tag{37}$$

where A is an *mxn* matrix – $A \in \mathbb{R}^{mxn}$ – and $\boldsymbol{x}$ and $\boldsymbol{b}$ are *mx*1 vectors – $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^{n}$. It is essentially an unconstrained optimization problem, even though convex constraints can be easily added – see the section on convex optimization.

Least squares problems are ubiquitous in science and engineering. They arise in applications such as polynomial fitting, optimal control and maximum-likelihood estimation (Boyd & Vandenberghe, 2004).

The solution to a least-squares problem can be found analytically (Boyd & Vandenberghe, 2004):

$$\boldsymbol{x}^* = (A^T A)^{-1} A^T \boldsymbol{b} \tag{38}$$

The above formula, however, serves mostly to illustrate the relative ease of solving this type of problem. In practice, it is very costly to perform matrix inversions and multiplications and, for that reason, it is preferred to solve the linear system $A^T A\boldsymbol{x}^* = A^T \boldsymbol{b}$ for $\boldsymbol{x}^*$**.**

2.3.1.3.2 Linear programming

Linear programs (LP) are optimization problems whose cost functions and constraints are all affine. They can all be expressed by the general formula:

$$\begin{aligned} &\min \quad \boldsymbol{c}^T \boldsymbol{x} \\ &s.t. \\ &\qquad A\boldsymbol{x} = 0, \\ &\qquad B\boldsymbol{x} \leq 0 \end{aligned} \tag{39}$$

In the above problem $\boldsymbol{x}$ and $\boldsymbol{c}$ are $\mathbb{R}^n$ vectors and A and B are *mxn* and *oxn* matrices, respectively. It can be shown that all linear problems can be expressed in a simpler standard form (Luenberger & Ye, 2010):

$$\begin{aligned} &\min \quad \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} \\ &s.t. \\ &\qquad H\boldsymbol{x} = 0, \\ &\qquad \boldsymbol{x} \geq 0 \end{aligned} \tag{40}$$

In the latter formulation, inequality constraints are introduced by means of *slack variables*. For example, if we wanted to incorporate the constraint $3x_1 + x_2 \leq 0$, one way might be to create a slack variable s_1 (which should always be positive) and add the equality constraint $3x_1 + x_2 + s_1 = 0$. If a variable, say x_1, is meant to be unbounded, and not merely positive, we can either eliminate it by back substitution or replace it by the difference of two new positive variables, for example: $x_1 = y_1 - y_2$ (Luenberger & Ye, 2010).

There exist many algorithms that solve LP's. The most popular algorithm is the Simplex method, created by Dantzig. In practice, this algorithm works very well and has the nice feature that the first part of its code – which is called Phase I – can be used to solve the feasibility problem. One major drawback, however, is that this algorithm is exponential on the number of variables in the worst-case scenario. Other more recent algorithms are the interior-point methods, which we will come back to later. A thorough discussion on LP's can be found in (Vanderbei, 2007).

2.3.1.3.3 Quadratic programming

Quadratic programs (QP) are optimization problems whose constraints are affine and whose objective function is quadratic (Boyd & Vandenberghe, 2004):

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^{\boldsymbol{T}} P\boldsymbol{x} - \boldsymbol{q}^{\boldsymbol{T}}\boldsymbol{x} + r \\ s.t. \quad & \\ & A\boldsymbol{x} = b, \\ & C\boldsymbol{x} \leq d \end{aligned} \tag{41}$$

where P is a positive semi-definite matrix. In the special case where no constraints are present, it can be shown (by taking the gradient of the function and equating it to zero) that minimizing the QP is equivalent to solving the linear system (Boyd & Vandenberghe, 2004):

$$Px^* = q \tag{42}$$

Least-squares can be seen a particular case of a QP. QP's arise in applications such as portfolio minimization, regression and optimal control. Aside from that, it is the basis for an extremely common local optimization algorithm, which is Newton's method (Boyd & Vandenberghe, 2004); (Luenberger & Ye, 2010).

2.3.1.3.4 Convex programming

A convex optimization problem takes the form (Boyd & Vandenberghe, 2004):

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ s.t. \quad & \\ & A\boldsymbol{x} = \boldsymbol{b} \\ & g_i(\boldsymbol{x}) \leq 0, \qquad i = 1 \dots r \\ & \boldsymbol{x} \in D \end{aligned} \tag{43}$$

where the cost function and constraints are convex functions and D is a convex set. A formal definition of convexity and some of its properties are given in the Mathematical definitions

section. Intuitively, convex functions are those functions that are bowl-shaped with an upwards-facing curvature, such as an exponential or a parabola with positive second derivative.

There are efficient algorithms designed to efficiently and deterministically solve large convex problems. In fact, all types of problems exposed in the last sections are also convex optimization problems. Many algorithms that solve general optimization problems, such as the one that is the focus of this work, rely on solving convex sub-problems.

2.3.1.4 Deterministic and stochastic optimization

Depending on nature of the algorithm that is used to solve an optimization problem, we divide the optimization process as being either *deterministic* or *stochastic*. We perform stochastic optimization when either randomness is injected (Monte Carlo) in the algorithm or when noise is present in the measurements provided to the algorithm (Spall, 2003). If no randomness is present, that is, if we can exactly predict the outcome of the algorithm from its starting state we are performing deterministic optimization.

Many stochastic algorithms have the distinct advantage of making very few assumptions about the objective function, therefore making them applicable to a very wide range of functions. Deterministic algorithms may run into problems if the objective function has, for instance, several discontinuities or if it is non-differentiable, which may limit their applicability. Stochastic methods may run faster (depending on the implementation and number of variables of the problem) or require less computational effort than many deterministic global optimization algorithms.

Stochastic algorithms, however, do have a few drawbacks. Most of them need parameters to be specified by the user, which means that the user must properly tune them, which may be very laborious and possibly problem-specific. Constraints handling can also be very troublesome with these algorithms. Another drawback is that these methods in general do not provide any analytical way to check whether the global minimum has been reached. It is often the case in deterministic optimization – especially in well-posed convex programs – that a certificate of optimality can be issued to assure that the minimum has, indeed, been found.

2.3.2 Duality

Another concept, which is fundamental in optimization (and which will show up later in the bibliographic review), is that of duality. Given an optimization problem, which we may call *primal problem*, we may define a *dual problem* as follows. Let us consider problem (44), the primal problem.

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ s.t. \quad & \\ & h_i(\boldsymbol{x}) = 0, \qquad i = 1 \dots m \\ & g_i(\boldsymbol{x}) \leq 0, \qquad i = 1 \dots r \end{aligned} \tag{44}$$

Let us then define the Lagrangian associated to it as (Boyd & Vandenberghe, 2004):

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{i=1}^{r} \mu_i g_i(x) \tag{45}$$

where the values of λ_i and μ_i are the so-called Lagrange multipliers. The dual function f^{dual} is then defined as:

$$f^{dual}(\lambda, \mu) = \inf_x L(x, \lambda, \mu) \tag{46}$$

This function possesses several important properties, such as (Boyd & Vandenberghe, 2004):

1. It is always concave;
2. It underestimates the primal function (i.e. provides lower bounds). In particular, it provides lower bounds for the optimum value of the primal problem.

Now we are in position to define the dual problem:

$$\begin{aligned} &\max \quad f^{dual}(\lambda, \mu) \\ &s.t. \\ &\qquad \lambda_i(\boldsymbol{x}) \geq 0, \qquad i = 1 \dots m \end{aligned} \tag{47}$$

As previously stated, the dual function provides lower bounds to the primal problem. It must then be the case that the solution to (47) is also a lower bound. If p^* denotes the optimum value for the primal problem and d^*denotes the optimum value for the dual problem, then:

$$d^* \leq p^* \tag{48}$$

This property is often referred to as *weak duality*. The difference $p^* - d^*$ is the *duality gap*. A stronger and more interesting property is *strong duality*, which corresponds to having zero duality gap. In other words, strong duality holds if $d^* = p^*$. Unlike weak duality, strong duality does not, in general, hold. There are, however, mathematical conditions that ensure strong duality for a given problem (Boyd & Vandenberghe, 2004). These conditions are called *constraint qualifications*. One such example is Slater's condition: if a problem is convex and there exists a strictly feasible point, then strong duality holds. By strictly feasible, we mean a feasible point that strictly satisfies the inequalities ($g_i(\boldsymbol{x}) < 0$ for all i).

2.3.3 Conditions for optimality

We already have a definition for local and global optimality. Nevertheless, we can still obtain further insight into what it means to be an optimum point if we assume that the objective function and the constraints are differentiable. This will lead to what we call the Karush-Kuhn-Tucker (KKT) conditions for constrained problems (Boyd & Vandenberghe, 2004).

The KKT conditions state that if x^* is an optimum point, then there exist constants λ_i^* and μ_i^* such that:

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*) + \sum_{i=1}^{r} \mu^*{}_i \nabla g_i(x^*) = 0$$
$$g_i(x^*) \leq 0 \qquad i = 1 \dots r$$
$$h_i(x^*) = 0 \qquad i = 1 \dots m$$
$$\mu_i^* \geq 0 \qquad i = 1 \dots r$$
$$\mu_i^* g_i(x^*) = 0 \quad i = 1 \dots r \tag{49}$$

If the problem is convex, then the KKT conditions are sufficient to guarantee that the points x^* and (λ_i^*, μ_i^*) are, respectively, primal and dual optimal with zero duality gap (Boyd & Vandenberghe, 2004).

2.3.4 Interior point methods for constrained optimization

Interior point methods (IPM) are a class of iterative algorithms for solving constrained optimization problems (Boyd & Vandenberghe, 2004). They rely on solving a sequence of approximate problems such that the solutions to these problems converge to the one corresponding to the original problem.

Interior point methods make use of the so-called barrier functions, among which is the logarithmic barrier function. Logarithmic barrier functions act upon inequality constraints in such a way that the closer inequalities are of being violated, the greater the value returned by the barrier function. Mathematically, if a constrained optimization problem is of the form:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ s.t. \quad & \\ & A\boldsymbol{x} = \boldsymbol{b}, \\ & g_i(\boldsymbol{x}) \leq 0, \qquad i = 1 \dots r \end{aligned} \tag{50}$$

We may approximate it by an equality constrained optimization problem of the form (Boyd & Vandenberghe, 2004):

$$\begin{aligned} &\min \quad f(\boldsymbol{x}) + \sum_{i=1}^{r} -\left(\frac{1}{t}\right) \ln[-g_i(\boldsymbol{x})] \\ &s.t. \\ &\qquad A\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{51}$$

The logarithmic barrier function is the function represented by the summation. As the parameter t grows larger, the better the original problem will be approximated by Problem (51). The above optimization problem is often written as (Boyd & Vandenberghe, 2004):

$$\begin{aligned} &min \quad tf(\boldsymbol{x}) + \phi(\boldsymbol{x}) \\ &s.t. \\ &\qquad A\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{52}$$

Where

$$\phi(\boldsymbol{x}) = -\sum_{i=1}^{r} \ln[-g_i(\boldsymbol{x})] \tag{53}$$

The IPM initially takes a strictly feasible point $\boldsymbol{x}^0$, a positive scalar $t = t^0 > 0$, another scalar $\beta > 1$ and a prespecified tolerance ϵ. At each iteration, Problem (52) is solved and the parameter t is updated to βt. The process is repeated until $\frac{r}{t} < \epsilon$ (Boyd & Vandenberghe, 2004).

There exist several iterative algorithms designed to solve constrained optimization problems (Luenberger & Ye, 2010) such as those subproblems in the IPM. Matlab does so by attempting to solve the KKT system of equations by either Newton's method or through a conjugate gradient method. Details of its implementation can be found in (Mathworks, 2016).

2.4 INTERVAL ANALYSIS

In order to obtain rigorous upper and lower bounds, one can resort to interval arithmetics, a branch of the field known as interval analysis (Moore, et al., 2009). This field of knowledge,

among other things, seeks to extend basic operations done on real numbers to the set of intervals. The basic operations of addition, subtraction, multiplication and division can be so extended as follows:

$$[a,b]+[c,d]=[a+c,b+d]$$

$$[a,b]-[c,d]=[a-c,b-d]$$

$$[a,b]\times[c,d]=[\min(ac,ad,bc,bd),\max(ac,ad,bc,bd)]$$

$$[a,b]/[c,d]=\left[\min\left(\frac{a}{c},\frac{a}{d},\frac{b}{c},\frac{b}{d}\right),\max\left(\frac{a}{c},\frac{a}{d},\frac{b}{c},\frac{b}{d}\right)\right] \quad (54)$$

It is possible to extend real monotonic functions to the intervals set (Moore, et al., 2009):

$$f([a,b])=[f(a),f(b)] \quad \text{if } f \text{ is monotonically increasing}$$
$$f([a,b])=[f(b),f(a)] \quad \text{if } f \text{ is monotonically decreasing} \quad (55)$$

In particular:

$$\exp([a,b])=[\exp(a),\exp(b)]$$

$$\ln([a,b])=[\ln(a),\ln(b)] \quad (56)$$

Integer, positive powers are also simple:

$$[a,b]^r=[a^r,b^r] \quad \text{if } r \text{ is odd} \quad (57)$$

$$[a,b]^r = \begin{cases} [a^r, b^r] & a,b \geq 0 \\ [b^r, a^r] & a,b, < 0 \\ [0, \max(a^r, b^r)] & \text{otherwise} \end{cases} \quad \text{if } r \text{ is even}$$

3 BIBLIOGRAPHIC REVIEW

Substantial effort has been made towards finding efficient and reliable techniques for solving the phase and chemical equilibrium problem by means of Gibbs energy minimization (Zhang, et al., 2011). However diverse the proposed methodologies may be, it is possible to distinguish common traits among them, which allows us to arrange them into broad categories. We attempt to do so in the following sections.

3.1 GRID-BASED METHODS

By grid-based methods, we are referring to methodologies that involve discretizing the cost function's domain and evaluating it at these grid points. This is strongly related to the idea of finding a function's convex hull and then optimizing it (which should be a simpler optimization problem, as the convex hull of any function is, by definition, convex) (Boyd & Vandenberghe, 2004).

(Greiner, 1988) observed that the chemical equilibrium problem is convex provided that all phases are ideal, that it, that the activity coefficients can be taken as 1. The problem is formulated in a similar way to that presented in the section *Chemical and phase equilibria*. The author also extends the discussion to the case where the free-energy function is non-convex (as it is in general the case when activity coefficients cannot be neglected). It is shown that the chemical equilibrium problem can be restated in terms of the free-energy convex hull as follows:

$$\begin{aligned} \min \quad & N \operatorname{conv}[G(\boldsymbol{x})] \\ s.t. \quad & \\ & NH\boldsymbol{x} = \boldsymbol{z} \\ & \mathbf{x} \geq 0 \end{aligned} \tag{58}$$

Here *N* denotes the total number of moles of all components in a system and $\boldsymbol{x}$ is a vector of global molar fractions. The function $\operatorname{conv}[G(\boldsymbol{x})]$ denotes the convex hull of $G(\boldsymbol{x})$. The author

does not mention in this work any specific convex programming algorithm or any method for efficiently obtaining the convex hull (Greiner, 1988).

A more recent and practical work, written by Ryll and collaborators (Ryll, et al., 2012) attempts to determine fluid phase diagrams through minimizing the free-energy function's convex hull. They refer to this method as the convex envelope method (CEM). The authors begin by introducing a modified molar Gibbs energy of mixture, which is intended to reduce the computational effort:

$$\Delta g_{mix} = g - \sum_{k=1}^{N_C} x_k g_k^{pure} \tag{59}$$

Where g is the molar Gibbs energy, g_k^{pure} is the Gibbs energy of the k-th component in pure form and N_C is the number of components. The function's domain is then discretized (in this paper, all points were evenly spaced) as finely as necessary and the function is evaluated at the grid points. For constructing the convex hull of the points so obtained, Matlab's implementation of the very popular Quickhull algorithm was used (Barber, et al., 1996). The authors successfully managed to determine phase diagrams for ternary and quaternary systems represented by the UNIQUAC model.

A somewhat hybrid approach, combining convex hull calculations and the Newton-Raphson local optimization method was employed by Perevoshchikova and co-workers (Perevoshchikova, et al., 2012). In their work, the authors attempt to design a method for calculating the compositions of two-phase multicomponent alloys. More specifically, this paper deals with the austenite-ferrite transformation in Fe-C-Cr alloys. They point out that the Newton-Raphson (NR) method, despite being frequently used in commercial software and despite having good numerical properties such as quadratic convergence ratio, is very much dependent on the quality of the initial estimate for the optimum point. In fact, it may converge to a non-global minimum which is far from the actual solution. It is proposed that the convex hull of the free-energy be used as a way of obtaining a higher quality initial estimate. Once obtained, this estimate can be used as an input to the NR method and it becomes possible to benefit from its suitable numerical properties. The authors also claim that, despite having been designed to correctly calculate phase diagrams for two-phase alloys, their method appears to be robust enough to handle more complex systems (Perevoshchikova, et al., 2012).

The variables chosen for discretization were the mole fractions. The grid was comprised of a total of βN^d nodes, where $\beta \leq 1$, d is the number of independent species and N is the number of nodes along the axis of a single species. The authors also decided to only investigate part of the function domain, namely, the region where the carbon concentration was such that $0 \leq x_C \leq 0.5$, as they claim that equilibria for higher values of x_C are rarely observed. They also chose the grid to be uneven: it was finer in the Fe-rich region than in the rest of the domain. More precisely, the grid was defined as follows:

$$x_C = \left(\frac{1}{N-1}\right)^2 \qquad i \in \left[0, \left\lfloor \frac{N-1}{\sqrt{2}} \right\rfloor\right]$$
$$x_C = \frac{1}{N-1} \qquad i \in [0, N-1] \tag{60}$$

Once the grid has been created, nodes belonging to non-convex regions are detected by computing a function referred to as the stability function (closely related to the Hessian of the free-energy function), denoted by Ψ.

$$\Psi = \prod_{i>1} x_i \det\left[\frac{\partial^2}{\partial x_j \partial x_k}\left(\frac{G}{RT}\right)\right] \tag{61}$$

The nodes lying in non-convex regions were then used to construct the convex hull of those same regions. The algorithm chosen for that was a modified version of the Quickhull algorithm.

From then on, the convex hull was refined, tie-lines were drawn and phase diagrams were successfully constructed. One of the so constructed phase diagrams is shown in Figure 2. This diagram depicts the phase diagram for an alloy composed of Fe-C-Cr (0.92 mol% C and 2.66 mol% Cr) at 800 °C. The regions corresponding to different phases are separated by the bold curves.

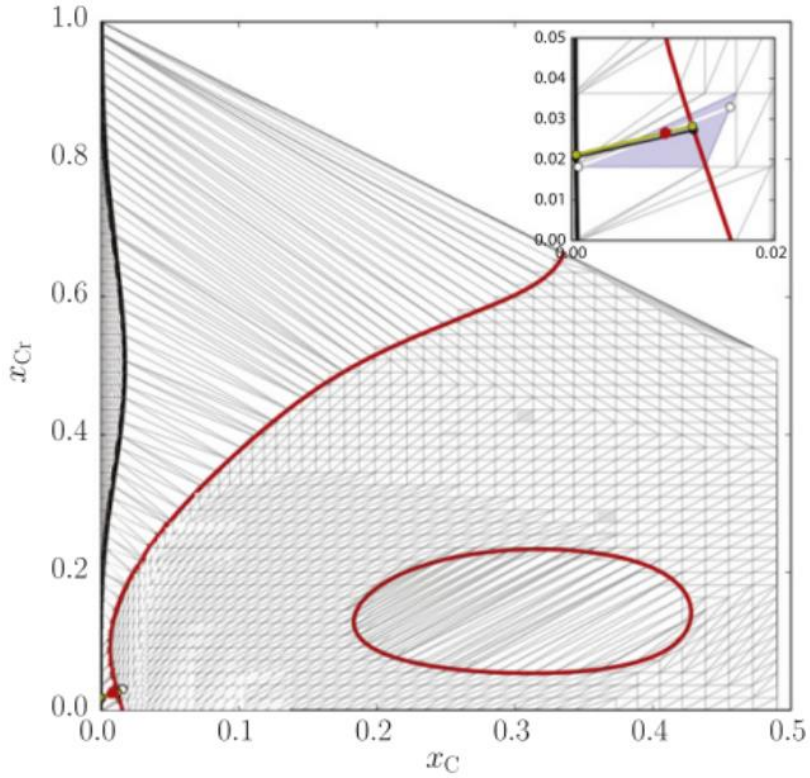


Figure 2 - Calculated phase diagram for a Fe-C-Cr system after tie-line search and refinement (Perevoshchikova, et al., 2012).

It must be noted that one of the major drawbacks of grid-based methods is the fact that he number of grid points to be evaluated (and possibly convexified) grows exponentially with the number of independent components, which may lead to very high running times for relatively complex systems. Another issue is the fact that it is necessary to choose a grid discretization that is fine enough to accurately describe the essential topological features of the Gibbs free-energy function, but also coarse enough to allow for as little computational effort as possible. It may be troublesome to find the ideal number of grid points if one does not possess any *a priori* knowledge of the specific problem.

3.2 LINEAR PROGRAMMING-BASED METHODS

Attempts have been made to somehow fit the equilibrium problem into the LP framework. That is only natural, as there are fast and reliable algorithms for solving this type of optimization problem. Many authors have contributed for the development of the LP-based methods applied

to Gibbs energy minimization. Some of the works that helped lay the foundations for this field are briefly mentioned here. Bullard and Biegler (1991) devised a method for solving systems of equations by means of converting them into optimization problems and further solving them by LP techniques. (Gopal & Biegler, 1997) later applied their method to solve simulation problems, such as the simulation of a flash distillation unit.

An even earlier attempt that is worth mentioning is the one made by (Greiner, 1988), which sets off to solve multiphase chemical equilibria through generalized linear programming. The author claims that there are several advantages to this approach, among which: *a)* it applies to ideal and non-ideal phases alike; *b)* global convergence can be proved; *c)* when used in conjunction with a Newton-based local optimization method the overall algorithm's accuracy and rate of convergence can be boosted. The method is outlined next. We must first define what is a generalized linear program. Let $G^j \quad j = 1, \dots, p$ denote convex subsets of $\mathbb{R}^{m+1}$. A point $\boldsymbol{g}^j \in G^j$ can be represented as an ordered pair $(c^j, \boldsymbol{p}^j) \in \mathbb{R} \times \mathbb{R}^m$. A generalized LP is an optimization problem that can be written as:

$$\begin{aligned} \min \quad & \sum_{j=1}^{p} n_j c^j \\ s.t. \quad & \\ & \sum_{j=1}^{p} n_j \boldsymbol{p}^j = \boldsymbol{b}, \\ & n_j \geq 0 \\ & (c^j, \boldsymbol{p}^j) \in G^j \end{aligned} \tag{62}$$

In order to ensure that a solution exists, it is also asked that $n_j \leq C$, where C is a constant. It is possible to show that the equilibrium problem in its most general form can be thought of as a generalized linear program, provided that the convex hull of the Gibbs energy is used (Greiner, 1988). We now return to the problem of numerically computing convex hulls, which can be solved by methods such as those previously described. In order for us to see that the problems are indeed equivalent, we notice that the equilibrium problem for a system with p phases is determined by the molar free energies of each of its constituent phases by and stoichiometric coefficients matrices for each phase, here denoted by $\boldsymbol{R}^j$:

$$G^j: Q^j \rightarrow \mathbb{R}$$
$$\boldsymbol{R}^j: Q^j \rightarrow \mathbb{R}^m \tag{63}$$

By Q^j we denote a convex and bounded subset of $\mathbb{R}^s$ (in this case, the bounded set of possible mole numbers), and by G^j we denote the Gibbs energy corresponding to the *j*-th phase. The problem consists of:

$$\begin{aligned} \min \quad & \sum_{j=1}^{p} N^j \mathrm{conv}\left[G(\boldsymbol{q}^j)\right] \\ s.t. \quad & \\ & \sum_{j=1}^{p} N^j \boldsymbol{R}^j \boldsymbol{q}^j = \boldsymbol{b}, \\ & N^j \geq 0 \end{aligned} \tag{64}$$

By comparing equations (62) and (64), it is possible to see that the Gibbs minimization problem can be expressed as a generalized linear program. This may be done by setting $\boldsymbol{p} = \boldsymbol{R}^j \boldsymbol{q}^j$ and $c \geq \mathrm{conv}\left[G\left(\boldsymbol{q}^j\right)\right]$. The problem domain, then becomes restricted to the following convex set (Greiner, 1988):

$$G^j = \left\{(c, \boldsymbol{p}) \in \mathbb{R}^{s(j)+1} \mid \exists\, \boldsymbol{q}^j \in \boldsymbol{Q}^j \text{ with } c \geq \mathrm{conv}\left[G\left(\boldsymbol{q}^j\right)\right], \boldsymbol{p} = \boldsymbol{R}^j \boldsymbol{q}^j\right\} \tag{65}$$

This correspondence is explained in much detail in the original paper. Upon restating the problem with the aid of the aforementioned convex sets, an algorithm is presented and a proof of its convergence and correctness is given (Greiner, 1988). This algorithm is guaranteed to terminate in a finite number of steps and to achieve ϵ-global optimality. This work is eminently theoretic and, as such, does not provide any results concerning its actual implementation (Greiner, 1988).

A more recent attempt has been made by Rossi and collaborators (Rossi, et al., 2009). In this approach, the mole fractions of each component are discretized over a uniform grid. The number of dimensions of the grid, for that reason, will be as large as the number of components under consideration. A very important point is that the grid is constructed in such a way that every grid point satisfies the property that the sum of mole fractions over all components equals 1. Each grid point, therefore, represents a potential phase in equilibrium. It is thus expected that, as the grid is set to be finer, the number of potential phases grow accordingly. The authors exemplify their grid-generating procedure for a hypothetical 3-components system. If we let k represent the k-th phase and z_i^k the molar fraction of component i in phase k, a viable grid is:

$$
\begin{aligned}
z_1^k &= 1 - \delta(p-1) \\
z_2^k &= \delta\left[\frac{p(p+1)}{2} - k\right] \\
z_3^k &= \delta\left[k - 1 - \frac{p(p-1)}{2}\right] \\
\delta &= \frac{1}{N} \\
p &= \left\lceil \frac{\sqrt{8k+1}-1}{2} \right\rceil
\end{aligned}
\tag{66}
$$

It can be shown that for every phase the equality $z_1^k + z_2^k + z_3^k = 1$ holds true. Once the grid is properly defined, all quantities that are dependent on composition, including the chemical potential, become determined. The problem is therefore reduced to minimizing the total Gibbs energy provided that the components are distributed somehow among the potential phases. Since the nonlinearity of the original problem resided on the fact that that the chemical potential function is nonlinear – and as it is now determined – we are left with an LP. Its constraints are:

$$
n_i^k = z_i^k \sum_{j \in C} n_j^k \tag{67}
$$

The authors have applied their methodology to binary and ternary systems. Their results were compared both to experimental data and to computational results found in the literature. Good agreement was found (Rossi, et al., 2009). Figure 3 shows the equilibrium compositions found

through this methodology for a ternary system of water, ethanol and hexane at 1 atm. Two activity models were considered, namely, UNIQUAC and NRTL.

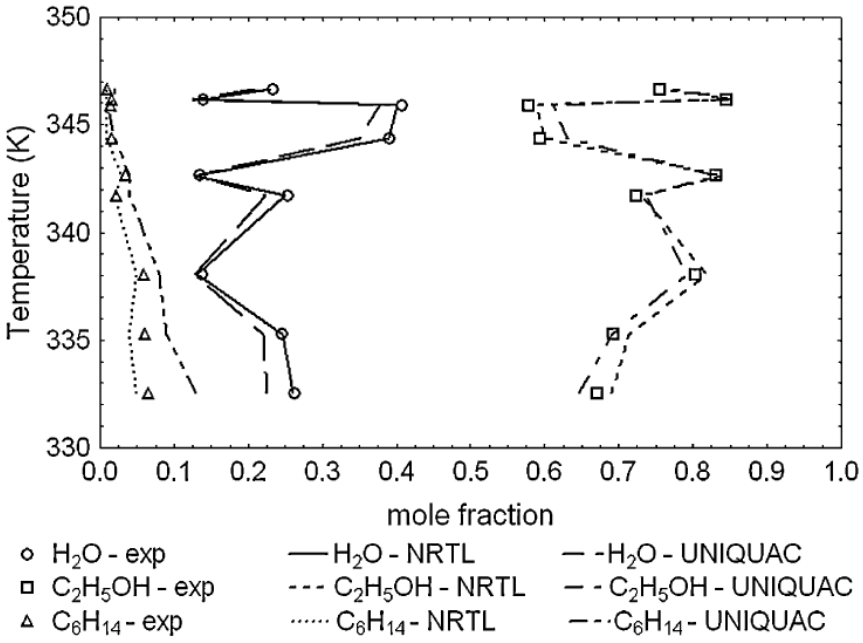


Figure 3 - Experimental validation of the results obtained by means of the methodology proposed by Rossi and collaborators (Rossi, et al., 2009).

The authors warn that the quality of the global minimum is very much dependent on how finely the grid is discretized. In addition, results indicate that this methodology may lead to considerably high computational running times.

3.3 INTERIOR POINT METHODS

These methods make use of both the primal and of the dual formulations of the optimization problem to solve it. A paper which was of seminal importance is the one by (Karpov, et al., 1997). This paper supports the use of Interior Points Methods (IPMs) in Gibbs energy minimization problem and introduces a set-theory notation for the components of the system of interest that would be adopted in more recent articles. Their work is focused in geochemical modeling, for

which reason it pays special attention to electrolytic solutions. We now turn to some of the major insights provided by their methodology.

The molar Gibbs energy of the system is written as (Karpov, et al., 1997):

$$G(\boldsymbol{x}) = \sum_{j \in L} c_j x_j + \sum_{j \in l_\alpha} x_j \ln \frac{x_j}{X_\alpha} - \sum_{j \in S_w^0} x_j \ln \frac{x_{jw}}{X_\alpha}$$

$$X_\alpha = \sum_{j \in L_\alpha} x_j \tag{68}$$

Where L designates the set of components chosen to be independent, l_α is the set of indices of dependent components in phase α and $S_w^0 = S_w - \{j_w\}$, where S_w is the set of indices for a phase in which the standard states of its components follow an asymmetric reference scale (aqueous electrolytes) and j_w is the index of water (Karpov, et al., 1997). The values taken by c_j are component-dependent:

$$c_j = \begin{cases} \dfrac{G_i^0}{RT} + \ln \gamma_i & j \in L - \left(S_g \cup S_w^0\right) \\ \dfrac{G_i^0}{RT} + \ln \gamma_i + \ln P & j \in S_g \\ \dfrac{G_i^0}{RT} + \ln \gamma_i + \ln 55.51 & j \in S_w^0 \end{cases} \tag{69}$$

Here S_g denotes the set of gaseous species. The mass balance and positivity constraints are written in the usual way, only taking the molar fractions as variables instead of mole numbers. The optimization problem may therefore be expressed as:

$$\begin{aligned} &\min \quad G(\boldsymbol{x}) \\ &s.t. \\ &\qquad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{70}$$

$$\boldsymbol{x} \geq 0$$

It follows from the previous definitions that (Karpov, et al., 1997):

$$v_j = \frac{\partial G}{\partial x_j} = \begin{cases} c_j + \ln \dfrac{x_j}{X_\alpha} & j \in L - S_w \\ c_j + \ln \dfrac{x_j}{X_w} - \ln \dfrac{x_{j_w}}{X_w} - \dfrac{x_{j_w}}{X_w} + 1 & j \in S_w^0 \\ c_{j_w} + \ln \dfrac{x_{j_w}}{X_w} - \dfrac{X_w}{x_{j_w}} - \dfrac{x_{j_w}}{X_w} + 2 & j = j_w \end{cases}$$

$$X_w = \sum_{j \in S_w} x_j \tag{71}$$

Here, v_j represents the partial derivatives of G with respect to x_j, which is, by definition, the normalized chemical potential of species j (Karpov, et al., 1997). Now that it is possible to evaluate v_j, the KKT conditions for the problem become (Karpov, et al., 1997):

$$\begin{gathered} \boldsymbol{A x^*} = \boldsymbol{b} \\ (\boldsymbol{x^*})^{\boldsymbol{T}} \left(\boldsymbol{v} - \boldsymbol{A^T u} \right) = \boldsymbol{0} \\ \boldsymbol{v} - \boldsymbol{A^T u} \geq \boldsymbol{0} \\ \boldsymbol{x^*} \geq \boldsymbol{0} \end{gathered} \tag{72}$$

The KKT conditions are key in this methodology. By checking whether all KKT conditions are satisfied for all potential components and phases in the system, it is possible to rule out local minima, as this verification is bound to fail at them (Karpov, et al., 1997). A slight modification to this formulation – one that includes upper bounds for the molar fractions – is also discussed. The only major change that ensues is the fact that a few extra KKT inequalities arise.

The actual IPM algorithm used is an ellipsoid method, whose implementation is shown in Appendix A.

(Kulik, 2012) have built upon the methodology just described. A complete geochemical modeling package – GEM-Selektor – was developed by them in C/C++ and its numerical kernel makes use

of essentially the same procedure, with a few modifications. Their package has a very broad scope, being able to deal with to deal not only with chemical equilibria in its purest sense, but also with transportation problems through the finite-elements numerical method. Moreover, the authors integrate both these approaches in a unified framework. Figure 4, extracted from their article, schematically depicts the models used and their interactions (Kulik, 2012).

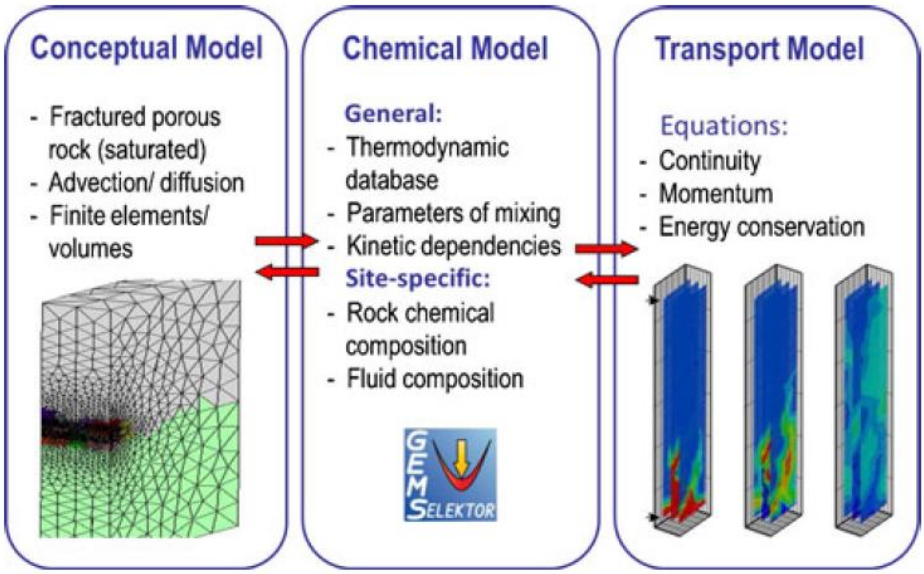


Figure 4 - Interactions between the models of the GEM-Selektor package (Kulik, 2012).

With respect to the numerical kernel concerning Gibbs energy minimization, their approach differs in a few ways from that of (Karpov, et al., 1997). The choice variables here are the number of moles of each component, in contrast with the latter approach, which made use of their molar fractions. In case no initial estimates for the number of moles of each component in equilibrium is provided, a feasible initial approximation (FIA) is calculated by linear programming. In this case, the following LP is solved via the simplex method (Vanderbei, 2007):

$$\begin{aligned} &\min \quad \sum_{j \in L_s} n_{j(s)}^{(y)} c_j \\ &s.t. \\ &\qquad \boldsymbol{A n}_{(s)}^{(y)} = \boldsymbol{n}^{(b)}, \\ &\qquad \boldsymbol{n}_{(s)}^{(y)} \geq \boldsymbol{0} \end{aligned} \tag{73}$$

Where, adhering to the article's notation, $\boldsymbol{n}_{(s)}^{(y)}$ is the initial approximation of mole numbers given by the simplex solution of the LP and $\boldsymbol{n}^{(b)}$ is the input vector of total amounts of independent components. The values of c_j are calculated exactly as before in Equation (69). This initial guess can be further improved through a mass balance refinement (MBR). This procedure is briefly described in Appendix A.

3.4 GLOBAL OPTIMIZATION METHODS

We now turn our attention to techniques that attempt to rigorously either ensure that the global minimum will be reached within a finite number of iteration (as is the case with the deterministic methods) or to ensure that it is expected, in a probabilistic sense, that the algorithm will eventually converge to the global minimum given enough iterations (which is the case for stochastic methods).

3.4.1 Stochastic approaches

A few studies have been conducted regarding the application of stochastic methods for solving reactive phase equilibrium problems (Bonilla-Petricolet, et al., 2011). Bonilla-Petricolet and collaborators (2011) studied the applicability of genetic algorithms (GA) and differential evolution with tabu list (DETL) to solving reactive equilibrium problems in transformed thermodynamic coordinates, exactly as described in the section *Reduction of dimensionality*. They ran the tests in 8 benchmark problems, which covered 4 different activity models, namely, Margules, NRTL, Wilson and UNIQUAC . They also compared their results to those obtained with simulated annealing (SA). They found that DETL displays better performance for this kind of problem.

An earlier approach by Nichita and collaborators (2002) used the so-called Tunneling method to perform chemical equilibrium calculations for one, two and three-phase systems. The algorithm requires an initial estimate $\boldsymbol{x}^0$ of the equilibrium composition and consists of two

repeating steps (Nichita, et al., 2002). In the first step, a local constrained optimization solver is used to find a local solution. In the second step, which is called the Tunneling step, another point (located in another valley) is picked at random according to a certain probability distribution. The function describing this distribution is the Tunneling function. The authors have found this method is capable of solving problems with several local minima and claim that their method has shown to be faster than other global optimization methods found In the literature (Nichita, et al., 2002).

More recently, Bhargava and collaborators (2013) applied the relatively new Cuckoo Search (CS) algorithm, which is a population-based method for solving phase equilibria (Bhargava, et al., 2013). They ran the algorithm in 8 different benchmark equilibrium problems and compared their results with other stochastic methods, namely, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), the Firefly Algorithm and Integrated Differential Evolution. The authors claim that CS may be among the best stochastic methods for equilibrium calculations (Bhargava, et al., 2013).

3.4.2 Deterministic approaches

The deterministic approaches to solving the equilibrium problem are branch-and-bound algorithms. Branch-and-bound algorithms rely on systematically subdividing and scanning the cost function's domain until it is mathematically guaranteed that the global minimum has been found. One such algorithm is the so-called αBB, which makes very few assumptions on the cost function and its constraints; namely, it demands that all the functions involved be twice-differentiable. This allows for a very broad array of activity coefficients models to be successfully evaluated. However, as very few assumptions on the function are made, in general, a big part of the optimization domain has to be examined before one can assure that the global optimum has been found. Another approach, the so-called Global Optimization (GOP) algorithm, is restricted to functions that are biconvex (to be explained later), which therefore limits its application. An earlier approach, which is also worth mentioning, is the approach by McDonald and Floudas (1995) of writing the chemical and phase equilibrium problem for the Wilson, ASOG and UNIFAC activity models as a difference of convex functions (McDonald & Floudas, 1995).

3.4.2.1 GOP

The so-called Global Optimization (GOP) algorithm for biconvex functions has been used by (McDonald & Floudas, 1995) for solving Gibbs energy minimization that employed the NRTL equation. The GOP is suitable for problems that can be written as:

$$\begin{aligned} &\min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y}) \\ &s.t. \\ &\quad \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = \mathbf{0}, \\ &\quad \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq \mathbf{0}, \\ &\quad \boldsymbol{x} \in X \\ &\quad \boldsymbol{y} \in Y \end{aligned} \tag{74}$$

Where:

I. f and g are biconvex
II. h is affine in $\boldsymbol{x}$ for every fixed $\boldsymbol{y}$ and vice-versa
III. X and Y are nonempty, compact, convex sets such that a constraint qualification such as Slater's (Boyd & Vandenberghe, 2004) is satisfied. Qualification constraints have been briefly discussed in the *Foundations* part of this work.

For a function to be biconvex, it means that its set of input variables can be partitioned in two subsets $\boldsymbol{x}$ and $\boldsymbol{y}$ such that for a fixed $\boldsymbol{x}^0$, $f(\boldsymbol{x}^0,\boldsymbol{y})$ is convex in $\boldsymbol{y}$ and for a fixed $\boldsymbol{y}^0$, $f(\boldsymbol{x},\boldsymbol{y}^0)$ is convex in $\boldsymbol{x}$. If our problem is posed correctly, all inequality constraints will be of the form $\boldsymbol{x}^L \leq x \leq \boldsymbol{x}^U$, i.e. box constraints, which constitutes a convex and compact set (provided that $\boldsymbol{x}^L \neq \boldsymbol{x}^U$). The equality constraints (mass balances) are linear equations, hence, the set of equality constraints is an affine set.

If the activity coefficients model chosen is the NRTL model, it can be shown that the Gibbs energy minimization problem is biconvex. This results follows from the remarkable property of the NRTL model that is detailed in Appendix B, along with a deeper discussion of the algorithm.

The authors have examined seven phase and chemical equilibria systems and compared their results with other results reported in the literature, having found satisfactory agreement. An interesting point made by the authors can be seen in Figure 5.

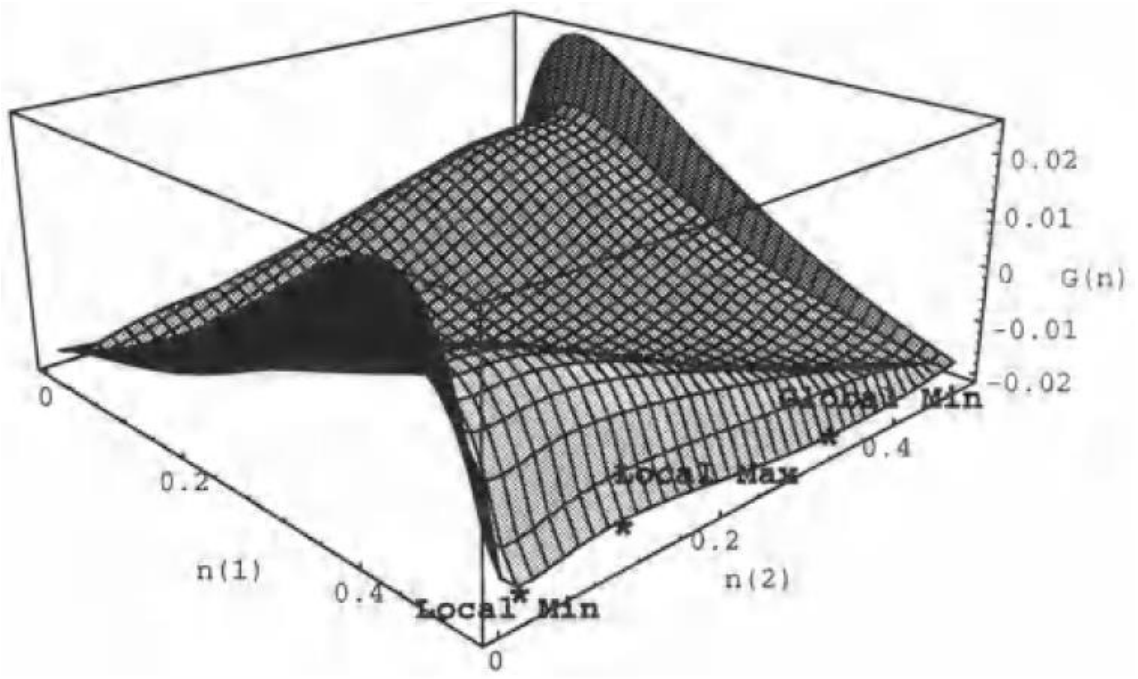


Figure 5- Gibbs energy for a system composed of n-Butyl-Acetate and water: local and global minima (Floudas, 2000)

.

It displays a surface plot of the Gibbs energy (all points on that surface are such that the mass balance constraints are satisfied) where it is possible to see a local minimum, whose value is quite close to that of the global minimum – a potential trap for local solvers. Also notice that the molar composition for these minima are quite different. This system, composed of n-Butyl-Acetate and water, despite having this somewhat undesirable numerical property is not particularly uncommon, which shows that even relatively simple systems may present serious difficulties to local solvers.

Even though the authors only apply the GOP algorithm to the NRTL model, it may be possible to extend its use to other models.

3.4.2.2 αBB (α-Branch-and-Bound)

The algorithm with which we will mostly be concerned in this work is the αBB algorithm, originally proposed by (Androulakis & Maranas, 1995). The αBB is a branch and bound algorithm meant for constrained global optimization of twice-differentiable functions. One of its merits is the generality of the problems for which it is suitable, namely, problems whose cost function and constraints are twice-differentiable. The phase and chemical equilibrium problem happens to be such a problem, provided that the activity coefficients model is also twice-differentiable, which is generally the case.

The algorithm solves the original problem by solving a series of convex underestimating functions all over the feasible region. As the underestimators are convex, they can be solved by means of any constrained-optimization local solver. At the same time, the original (primal) problem is solved locally over the feasible domain. Each of these solutions provides an upper bound for the global minimum. These lower and upper bounds are then used to gauge the quality of the minimum found, as the difference between the upper and lower bound at the global minimum must be zero. It can be shown that ϵ-optimality is guaranteed for this algorithm (Floudas, 2000).

In mathematical terms, the αBB algorithm is meant to solve optimization problems of the form:

$$\begin{aligned} &\min_{x} f(x) \\ &s.t. \\ &\quad \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \\ &\quad \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \\ &\quad x \in X \subseteq \mathbb{R}^n \end{aligned} \tag{75}$$

Where $f, g \text{ and } h$ belong to C^2 (i.e., are twice-differentiable). Notice that this algorithm accepts a much broader range of problems in contrast with the GOP. As the name suggests, it is a branch and bound algorithm, which means that during its execution there will be a branching step, at which the feasible domain is partitioned into two sub-problems and a bounding step, at which lower and upper bounds will be updated. One important aspect of the αBB is the quality of the underestimators used, which considerably impacts the convergence characteristics (such as the number of iterations required for convergence) of the problem.

Depending on the form of the cost function, it is possible that it can be split into terms that allow for rigorous, high-quality underestimators. Having that in mind that Floudas (2000) decomposes the cost function as a sum of simpler functions:

$$f(\boldsymbol{x}) = LT(\boldsymbol{x}) + CT(\boldsymbol{x}) + \sum_{i=1}^{bt} b_i x_{B_i,1} x_{B_i,2} + \sum_{i=1}^{tt} t_i x_{T_i,1} x_{T_i,2} x_{T_i,3} + \sum_{i=1}^{ft} f_i \frac{x_{F_i,1}}{x_{F_i,2}} + \sum_{i=1}^{ft} ft_i \frac{x_{FT_i,1} x_{FT_i,2}}{x_{FT_i,3}} + \sum_{i=1}^{ut} UT_i(x^i) + \sum_{i=1}^{nt} NT_i(\boldsymbol{x}) \tag{76}$$

In the above expression $LT(\boldsymbol{x})$ is the linear term, $CT(\boldsymbol{x})$ is the convex term, the sums on $x_{B_i,1}x_{B_i,2}$ represent the bilinear terms with coefficient b_i, the sums on $x_{T_i,1}x_{T_i,2}x_{T_i,3}$ are the trilinear terms with coefficient t_i. The remaining terms represent the fractional terms, fractional trilinear terms, univariate concave functions, $UT_i(x^i)$, and general nonconvex-terms, $NT_i(\boldsymbol{x})$. Clearly, all non-convex terms could be embedded into the category "general non-convex terms", but this decomposition allows for rigorous underestimators to be employed. As the convergence properties of the algorithm depend on the quality of the underestimators, the decomposition should be performed whenever possible. It can be shown that a valid underestimator for the above decomposed function is (Floudas, 2000):

$$L(\boldsymbol{x}, \boldsymbol{w}) = LT(\boldsymbol{x}) + CT(\boldsymbol{x}) + \sum_{i=1}^{bt} b_i w_{B_i} + \sum_{i=1}^{tt} t_i w_{T_i} + \sum_{i=1}^{ft} f_i w_{F_i} + \sum_{i=1}^{ft} ft_i w_{FT_i} + \sum_{i=1}^{ut} UT_i\left(x^{i,L}\right) + \frac{UT_i\left(x^{i,U}\right) - UT_i\left(x^{i,L}\right)}{x^{i,U} - x^{i,L}}\left(x - x^{i,L}\right) + \sum_{i=1}^{nt}\left[NT_i(\boldsymbol{x}) + \sum_{i=1}^{n} \alpha_{ij}\left(x_j^L - x_j\right)\left(x_j^U - x_j\right)\right] \tag{77}$$

More will be said later on how to calculate the coefficients α_{ij}, which give name to the method. The new variables, contained in the vector $\boldsymbol{w}$ must also satisfy certain inequality constraints. Even though in this work we will only make use of the bilinear underestimators, all inequalities constraints will be presented for completeness. For a bilinear term xy with $x \in [x^L, x^U]$, $y \in [y^L, y^U]$ (Floudas, 2000):

$$w_B = \begin{cases} x^L y + y^L x - x^L y^L & \text{if } y \leq -\dfrac{y^U - y^L}{x^U - x^L} x + \dfrac{x^U y^U - x^L y^L}{x^U - x^L} \\ x^U y + y^U x - x^U y^U & \text{otherwise} \end{cases} \tag{78}$$

That is in fact the convex hull for a bilinear term. For a trilinear term, a valid underestimator can be achieved with the following eight inequalities (Floudas, 2000):

$$\begin{aligned}
w_T &\geq xy^L z^L + x^L yz^L + x^L y^L z - 2x^L y^L z^L \\
w_T &\geq xy^U z^U + x^U yz^L + x^U y^L z - x^U y^L z^L - x^U y^U z^U \\
w_T &\geq xy^L z^L + x^L yz^U + x^L y^U z - x^L y^U z^U - x^L y^L z^L \\
w_T &\geq xy^U z^L + x^U yz^U + x^L y^U z - x^L y^U z^L - x^U y^U z^U \\
w_T &\geq xy^L z^U + x^L yz^L + x^U y^L z - x^U y^L z^U - x^L y^L z^L \\
w_T &\geq xy^L z^U + x^L yz^U + x^U y^U z - x^L y^L z^U - x^U y^U z^U \\
w_T &\geq xy^U z^L + x^U yz^L + x^L y^L z - x^U y^U z^L - x^L y^L z^L \\
w_T &\geq xy^U z^U + x^U yz^U + x^U y^U z - 2x^U y^U z^U
\end{aligned} \tag{79}$$

For fractional terms, it suffices to introduce (Floudas, 2000):

$$w_F \geq \begin{cases} \dfrac{x^L}{y} + \dfrac{x}{y^U} - \dfrac{x^L}{y^U} & x^L \geq 0 \\ \dfrac{x}{y^U} - \dfrac{x^L y}{y^L y^U} + \dfrac{x^L}{y^L} & x^L < 0 \end{cases}$$

$$w_F \geq \begin{cases} \dfrac{x^U}{y} + \dfrac{x}{y^L} - \dfrac{x^U}{y^L} & x^U \geq 0 \\ \dfrac{x}{y^L} - \dfrac{x^U y}{y^L y^U} + \dfrac{x^U}{y^U} & x^U < 0 \end{cases} \tag{80}$$

Finally, for fractional trilinear terms (Floudas, 2000):

$$\begin{aligned} w_{FT} &\geq xy^L/z^U + x^L y/z^U + x^L y^L/z - 2x^L y^L/z^U \\ w_{FT} &\geq xy^L/z^U + x^L y/z^L + x^L y^U/z - x^L y^U/z^L - x^L y^L/z^U \\ w_{FT} &\geq xy^U/z^L + x^U y/z^U + x^U y^L/z - x^U y^L/z^U - x^U y^U/z^L \\ w_{FT} &\geq xy^U/z^U + x^U y/z^L + x^L y^U/z - x^L y^U/z^U - x^U y^U/z^L \\ w_{FT} &\geq xy^L/z^U + x^L y/z^L + x^U y^L/z - x^U y^L/z^L - x^L y^L/z^U \\ w_{FT} &\geq xy^U/z^U + x^U y/z^L + x^L y/z - x^L y^U/z^U - x^U y^U/z^L \\ w_{FT} &\geq xy^L/z^U + x^L y/z^L + x^U y^L/z - x^U y^L/z^L - x^L y^L/z^U \\ w_{FT} &\geq xy^U/z^L + x^U y/z^L + x^U y^U/z - 2x^U y^U/z^L \end{aligned} \tag{81}$$

It remains to be seen how the coefficients α_{ij} are to be calculated. The idea is to take a value for α_{ij} that is high enough that the so-constructed convex underestimators are convex. First, we notice that the terms of the form:

$$NT_i(\boldsymbol{x}) + \sum_{i=1}^{n} \alpha_{ij}\left(x_j^L - x_j\right)\left(x_j^U - x_j\right) \tag{82}$$

are, indeed, valid underestimators for $NT_i(\boldsymbol{x})$, provided that $\alpha_{ij} > 0$ and that all x_j lie within the upper and lower bounds. That is indeed the case, because if $x_j > x_j^L$, $x_j < x_j^U$ and $\alpha_{ij} > 0$, it follows that:

$$\alpha_{ij}\left(x_j^L - x_j\right)\left(x_j^U - x_j\right) < 0 \Longrightarrow \sum_{i=1}^{n} \alpha_{ij}\left(x_j^L - x_j\right)\left(x_j^U - x_j\right) < 0$$

$$\Longrightarrow NT_i(\boldsymbol{x}) + \sum_{i=1}^{n} \alpha_{ij}\left(x_j^L - x_j\right)\left(x_j^U - x_j\right) < NT_i(\boldsymbol{x}) \tag{83}$$

Not only that, but it is clear that at the edges of the bounded domain, i.e., when $x_j = x_j^L$ or when $x_j = x_j^U$, the underestimator exactly matches the original function. We now claim that there always exists a high enough value of α_{ij} such that the underestimator so formed is convex. A proof for this claim can be found in (Maranas, 1994). We now turn to the very useful property of convex functions that a function is convex if and only if its Hessian is positive semi-definite. If Equation (77) is rewritten in simpler terms (Floudas, 2000),

$$L(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{n} \alpha_{ij}\left(x_j^L - x_j\right)\left(x_j^U - x_j\right) \tag{84}$$

Upon differentiation:

$$\frac{\partial^2 L(\boldsymbol{x})}{\partial x_i \partial x_j} = \begin{cases} \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} + 2\alpha_{ij} & \text{if } i = j \\ \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} & \text{if } i \neq j \end{cases} \tag{85}$$

Therefore, the Hessian matrix of the underestimator (H_L) coincides with the Hessian of the original function (H_f) everywhere except in its diagonal entries, which are shifted by a value of $2\alpha_{ij}$. A more succinct way of writing these relations is:

$$H_L = H_f + 2\Delta \tag{86}$$

Where Δ is the so-called *diagonal shift matrix*. It is a diagonal matrix such that $\Delta_{ii} = \alpha_{ii}$. If we wish to ensure that the underestimators are convex, one approach is to construct shift matrices in such a way that the resulting H_L is positive semi-definite. There are many approaches to this (Floudas, 2000). In very simple cases it is actually possible to compute analytical values for the α_{ii}, as is shown by Floudas (2000). However, for most situations, it is necessary that we somehow bound each entry of the Hessian and, through these bounds, construct a shift matrix that renders H_L positive semi-definite for all possible cases. These bounds can be obtained through interval analysis (Moore, et al., 2009).

It is customary to divide the shift matrices as being either uniform or non-uniform. Uniform shift matrices are such that all diagonal entries are equal, i.e. all the diagonal elements of H_L are shifted by the same value in contrast with H_f. If that is not the case, the shift matrix is non-uniform. Let us first consider the uniform methods and state the following theorem – adapted from (Floudas, 2000):

Theorem: Let $H_f(\boldsymbol{x})$ be the Hessian matrix of a function whose second-order derivatives are continuous and $L(\boldsymbol{x})$ be defined as in Equation (77). Let $[H_f]$ be a real, symmetric interval matrix such that $H_f(\boldsymbol{x}) \subseteq [H_f]\ \forall \boldsymbol{x} \in [\boldsymbol{x}^L, \boldsymbol{x}^U]$. If $[H_L] = [H_f] + 2\Delta$ is positive semi-definite, then $L(\boldsymbol{x})$ is convex. (1)

This theorem provides us with the theoretical basis upon which the methods for estimating α_{ii} are laid. The fact that the eigenvalues of a matrix is intimately connected with its positive semi-definiteness can be exploited to find ways of calculating α_{ii} from the interval matrix $[H_f]$ (Floudas, 2000).

One such technique is based on the Gershgorin's theorem. Its standard form can be used to estimate (sometimes quite crudely) the location of the eigenvalues of a matrix. Its enunciation can be found in standard linear algebra textbooks, such as (Lax, 2007). Here we will apply an adaptation of Gershgorin's theorem to interval matrices, as done in (Floudas, 2000).

Theorem: Let $[A] = [\underline{a_{ij}}, \overline{a_{ij}}]$ be an interval matrix. A lower bound on its minimum eigenvalue is given by:

$$\lambda_{min}(A) \geq \min_i \left[\underline{a_{ii}} - \sum_{j \neq i} \max\left(\left|\underline{a_{ij}}\right|, \left|\overline{a_{ij}}\right|\right)\right] \tag{2}$$

Here, $\underline{a_{ij}}$ and $\overline{a_{ij}}$ denote the infimum and supremum of the interval a_{ij}, respectively. This theorem gives us a (sometimes very crude) way of calculating a_{ii}. If we take $\alpha = \alpha_{ii}\ \forall i$ and also set:

$$\alpha \geq \max\left\{0, -\frac{1}{2}\lambda_{min}(A)\right\} \tag{87}$$

Then the lower bounding function will be convex (Maranas, 1994). Several other methods involving uniform shift matrices have been reported in the literature, and have been detailed in works such as (Floudas, 2000) and (Floudas & Paradalos, 2009).

There are also several methods concerning non-uniform shift matrices, among which is the one based on the scaled Gershgorin's theorem for interval matrices.

Theorem: Let $[A] = [\underline{a_{ij}}, \overline{a_{ij}}]$ be an interval matrix and let $\boldsymbol{d} > 0$ be any vector with positive entries. Define α_i as:

$$\alpha_i = \max\left\{0, -\frac{1}{2}\left(\underline{a_{ii}} - \sum_{j \neq i} |a|_{ij} \frac{d_j}{d_i}\right)\right\} \tag{3}$$

Where $|a|_{ij} = \max\left\{\left|\underline{a_{ij}}\right|, \left|\overline{a_{ij}}\right|\right\}$. If Δ is a diagonal matrix whose elements are the α_i, then $A_L = A + 2\Delta$ is positive semi-definite (Floudas, 2000).

The above theorem suggests a technique for finding Δ. It remains to be seen how to choose the vector $\boldsymbol{d}$ (Floudas, 2000). One way to do it is to pick $\boldsymbol{d}$ as a vector of ones ($d_i = 1\ \forall i$), which would yield an expression similar to that of Equation (87). However, all of its values will be

smaller when compared to the former matrix. Another choice of $\boldsymbol{d}$ is $\boldsymbol{d} = \boldsymbol{x}^U - \boldsymbol{x}^L$. This choice (which is a scaling procedure) compensates for the fact that variables with wider ranges have greater impact on the quality of the underestimators.

We now possess all the tools required to construct convex underestimators for general twice-differentiable functions. We can now turn our attention to the algorithm itself. The following discussion is mostly based on the works of Floudas and collaborators (Floudas & Paradalos, 2009); (Floudas, 2000).

A high-level description of the algorithm is presented next. The algorithm takes as input an arbitrary tolerance ϵ, the function f to be minimized, its constraints $\boldsymbol{h}(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$ and bounds $\boldsymbol{x}^U$, $\boldsymbol{x}^L$ on the variables. As we will be partitioning the domain and solving convex subproblems, it will be necessary to keep track of the solutions to these underestimating problems. Whenever we solve a convex underestimating problem, we find a lower bound to the global solution. We will denote by $\underline{f^k}$ the lower bound on the objective function's optimum value – f^* – obtained by solving the k-th underestimating problem. It is also true that by attempting to solve the original nonconvex problem by any local solver, an upper bound on the global solution will be found. As before, we will denote the upper bound found by solving the i-th nonconvex problem by $\overline{f^i}$. The pseudocode for the algorithm is:

Initialize upper and lower bounds: $\underline{f^*} = \underline{f^0} = -\infty$ and $\overline{f^*} = \overline{f^0} = +\infty$;
Add $\underline{f^0}$ to the list of lower bounds;
Set the tolerance ϵ;
While $\overline{f^*} - \underline{f^*} > \epsilon$
    Select the node k with smallest lower bound $\underline{f^k}$;
    Set $\underline{f^*} = \underline{f^k}$;
    Select a branching variable;
    Partition the domain around the branching variable and create new nodes corresponding to each subdomain so created;
    Do for each new node i
        Generate a convex underestimator;
        If necessary, introduce new variables and constraints;
        Compute α values;

Solve the i-th underestimating problem and find $\underline{f^i}$;

If the problem is infeasible or if $\underline{f^i} > \overline{f^*} + \epsilon$

Fathom node;

Else

Add $\underline{f^i}$ to the list of lower bounds;

Find a solution $\overline{f^\iota}$ of the original nonconvex problem;

If $\overline{f^\iota} < \overline{f^*}$

Set $\overline{f^*} = \overline{f^\iota}$;

Return $\overline{f^*}$ and variables at the corresponding node.

From the pseudocode it is easy to see that the algorithm implies a tree structure for the nodes. Four our purposes, we will always assume that these trees are binary trees. Figure 6 represents a possible solution process of the algorithm. The full domain, represented by node 0, is split into two child nodes, namely nodes 1 and 2. Each of these is checked for feasibility, underestimated and solved as explained before. The process is repeated for both child nodes and so on. If a node is either infeasible or yields an upper bound that is higher than the best upper bound (within a tolerance), it is fathomed. Fathomed nodes are colored gray. Notice that once the node has been fathomed, it no longer needs to be partitioned. The process goes on until a node is reached where the stopping criterion is met – the node with a black stripe. The algorithm then terminates and returns the global minimum and the variables corresponding to that node.

Many different strategies have been conceived in order to select the branching variable and perform the domain partitioning (Floudas, 2000). A naïve way to do it would be to simply select a variable at random. A more interesting approach would be to pick the branching variable to be the one that has the greatest corresponding variable range (i.e. the difference between its upper and lower bounds). The choice of branching variables is critical, as it determines which regions of the domain, and in which order, the algorithm will check for the solution. Poor choices of branching variables may lead to large number of iterations, which is particularly troublesome for problems with many variables. Figure 7 shows how the domain of an imaginary 2-variable problem will be partitioned according to the greatest variable range strategy. The gray areas denote which portion of the original domain is under analysis – i.e. solution of the original nonconvex problem and of the convex underestimating problem – at that iteration.

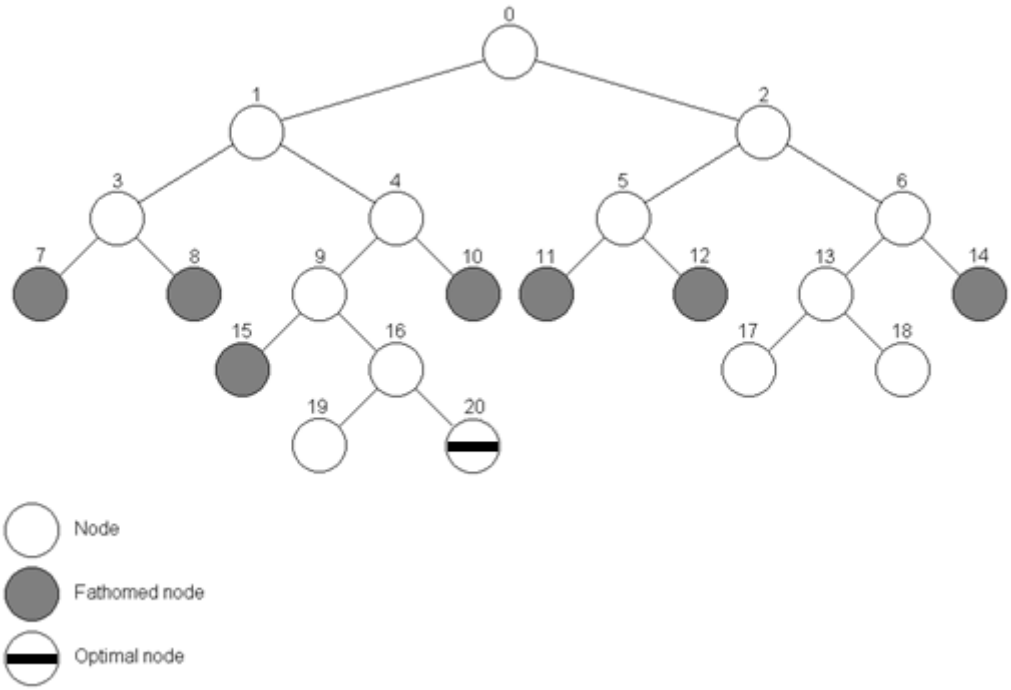


Figure 6 - Binary tree representing a possible solution process of the αBB algorithm. Adapted from (Floudas, 2000).

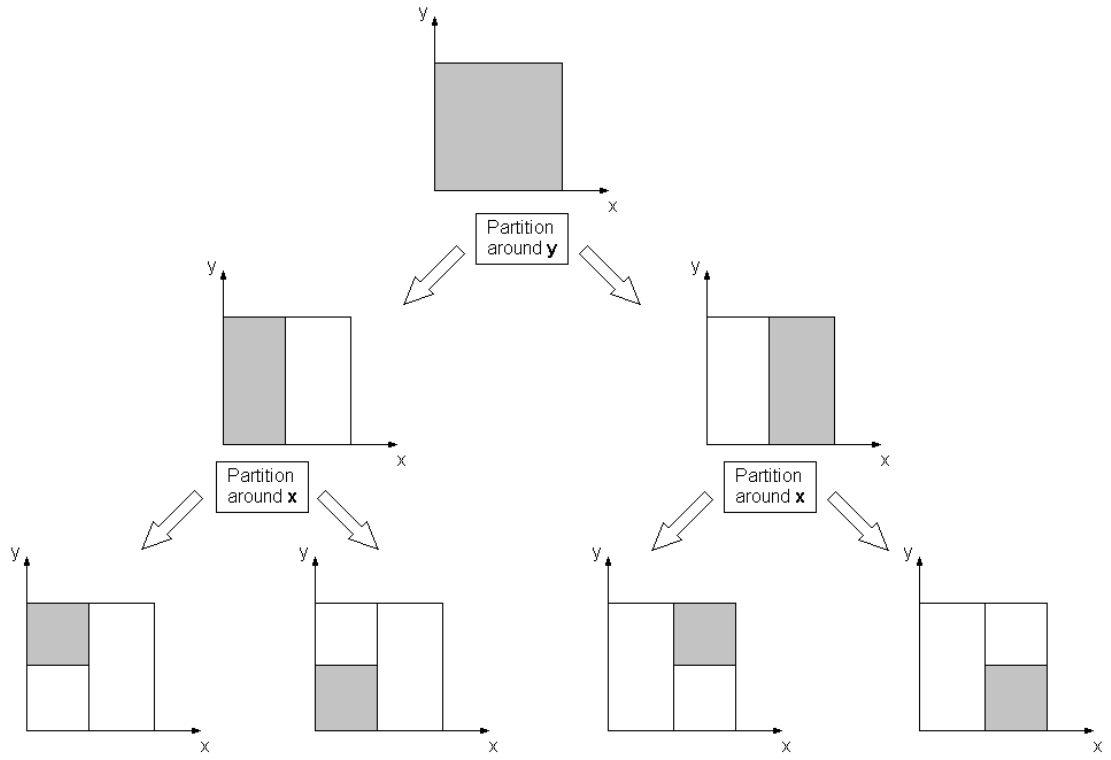


Figure 7 - Variable branching for a fictitious 2-variable optimization problem. The gray areas correspond to the part of the domain being analyzed at the current iteration.

4 METHODOLOGY

In this work, we have implemented an algorithm for solving chemical and phase equilibrium problems by means of the αBB algorithm. The implementation was carried out entirely in Matlab® 2013a and makes extensive use of its optimization toolbox. In order to ensure the algorithm's correctness and robustness, we conducted several tests on benchmark optimization functions and also on benchmark equilibrium problems. The latter were extracted from Bonilla-Petricolet and collaborators (2011). We now move on to discussing the statement of the chemical and phase equilibrium optimization problem, the exact formulation of these benchmark problems, the algorithm and its architecture, the data structures and classes defined therein and, lastly, we comment briefly about the unit tests conducted in the code's development.

4.1 BENCHMARK EQUILIBRIUM PROBLEMS

Table 1, adapted from (Bonilla-Petricolet, et al., 2011), summarizes the benchmark chemical and phase equilibrium problems used to evaluate the code's performance.

Table 1 - Benchmark chemical and phase equilibrium problems. Table adapted from (Bonilla-Petricolet, et al., 2011).

No.	System	Feed	Thermodynamic models
1	$A_1 + A_2 \rightleftharpoons A_3 + A_4$ (1) Ethanol (2) Acetic acid (3) Ethyl acetate (4) Water	$\boldsymbol{n_F} = [0.5, 0.5, 0, 0]$ T = 355 K P = 101.325 kPa	NRTL model and ideal gas. $K_{eq,1} = 18.670951$

2	$\boldsymbol{A_1 + A_2 \rightleftharpoons A_3}$, and $\boldsymbol{A_4}$ as an inert. (1) Isobutene (2) Methanol (3) Methyl tert-butyl ether (4) n-Butane	$\boldsymbol{n_F = [0.3, 0.3, 0, 0.4]}$ T = 373.15 K P = 1013.25 kPa	Wilson model and ideal gas. $\boldsymbol{\ln K_{eq,1} = -\frac{\Delta G^0_{rxs}}{RT}}$, T in K $\boldsymbol{\frac{\Delta G^0_{rxs}}{R} = -4205.05 + 10.0982T - 0.2667\, T \ln T}$
3	$A_1 + A_2 + 2A_3 \rightleftharpoons +2A_4$ (1) 2-Methyl-1-butene (2) 2-Methyl-2-butene (3) Methanol (4) Tert-amyl methyl ether	$\boldsymbol{n_F} = [0.354, 0.183, 0.463, 0]$ T = 335 K P = 151.95 kPa	Wilson model and ideal gas. $K_{eq,1} = 1.057.10^{-4} \exp\frac{4273.5}{T}$ Where T is in K.
4	$A_1 + A_2 \rightleftharpoons A_3 + A_4$ (1) Acetic acid (2) n-Butanol (3) Water (4) n-Butyl acetate	$\boldsymbol{n_F} = [0.3, 0.4, 0.3, 0]$ T = 298.15 K P = 101.325 kPa	UNIQUAC model. $\ln K_{eq,1} = \frac{450}{T} + 0.8$ Where T is in K.
5	$A_1 + A_2 \rightleftharpoons A_3$	$\boldsymbol{n_F} = [0.6, 0.4, 0]$	Margules solution model. $\frac{G^E}{RT} = 3.6x_1x_2 + 2.4x_1x_3 + 2.3x_2x_3$ $K_{eq,1} = 0.9825$
6	$A_1 + A_2 \rightleftharpoons A_3 + A_4$, and A_5 as an inert. (1) 2-Methyl-1-butene (2) 2-Methyl-2-butene (3) Methanol (4) Tert-amyl methyl ether (5) n-Pentane	$\boldsymbol{n_F} = [0.1, 0.15, 0.7, 0, 0.05]$ T = 335 K P = 151.9875 kPa	Wilson model and ideal gas. $K_{eq,1} = 1.057.10^{-4} \exp\frac{4273.5}{T}$ Where T is in K.
7	$A_1 + A_2 \rightleftharpoons A_3$	$\boldsymbol{n_F} = [0.52, 0.48, 0]$ T = 323.15 K P = 101.325 kPa	Margules solution model. $K_{eq,1} = 3.5$
8	$A_1 + A_2 \rightleftharpoons A_3 + A_4$	$\boldsymbol{n_F} = [0.048, 0.5, 0.452, 0]$ T = 360 K P = 101.325 kPa	NRTL model. $K_{eq,1} = 4.0$

The parameters needed for the activity coefficient models are listed in Appendix C.

4.2 COST FUNCTION AND CONSTRAINTS

To each problem, there corresponds an objective function (Bonilla-Petricolet, et al., 2011). Table 2 displays the objective function corresponding to each problem.

Table 2 - Objective functions for each benchmark equilibrium problem.

No.	Objective function	Number of variables
1, 4, 8	$F = \Delta g - \left(n_{4,1} + n_{4,2}\right) \ln K_{eq,1}$	9
2	$F = \Delta g - \left(n_{3,1} + n_{3,2}\right) \ln K_{eq,1}$	7
3	$F = \Delta g - 0.5\left(n_{4,1} + n_{4,2}\right) \ln K_{eq,1}$	7
5, 7	$F = \Delta g - \left(n_{3,1} + n_{3,2}\right) \ln K_{eq,1}$	7
6	$F = \Delta g - 0.5\left(n_{4,1} + n_{4,2}\right) \ln K_{eq,1}$	11

The variable Δg depends on the nature of the equilibrium problem under consideration. In the case of vapor-liquid equilibrium (VLE) problems, namely, problems 1, 2, 3 and 6:

$$\Delta g = \sum_{i=1}^{c} n_{i,1} \ln\left(x_{i,1}\gamma_{i,1}\right) + \sum_{i=1}^{c} n_{i,2} \ln\left(x_{i,2}P/P_i^{sat}\right) \tag{88}$$

For liquid-liquid equilibrium (LLE) problems, namely, problems 4, 5, 7 and 8:

$$\Delta g = \sum_{i=1}^{c} n_{i,1} \ln\left(x_{i,1}\gamma_{i,1}\right) + \sum_{i=1}^{c} n_{i,2} \ln\left(x_{i,2}\gamma_{i,2}\right) \tag{89}$$

Here *c* is the total number of components in the system under consideration and the subscripts 1 and 2 refer to different phases.

To each problem, there corresponds a set of mass-balance constraints. These are shown in Table 3. Notice that in all problems it is required that $\boldsymbol{n} \geq 10^{-6}$, instead of $\boldsymbol{n} \geq 0$. That is due to the fact that having $n_i = 0$ for some species, would imply that $x_i = 0$. This cannot be the case, because that would lead to the evaluation of $\ln 0$, which is undefined.

Table 3 – Mass balance constraints for each benchmark equilibrium problem.

No.	Constraints	
1	$\begin{cases} n_{1,1} + n_{1,2} + \xi = 0.5 \\ n_{2,1} + n_{2,2} + \xi = 0.5 \\ n_{3,1} + n_{3,2} - \xi = 0 \\ n_{4,1} + n_{4,2} - \xi = 0 \end{cases}$	$10^{-6} \leq n_{i,j} \leq 1.0$ $10^{-6} \leq \xi \leq 1.0$
2	$\begin{cases} n_{1,1} + n_{1,2} + \xi = 0.3 \\ n_{2,1} + n_{2,2} + \xi = 0.3 \\ n_{3,1} + n_{3,2} - \xi = 0 \\ n_{4,1} + n_{4,2} = 0.4 \end{cases}$	$10^{-6} \leq n_{i,j} \leq 1.0$ $10^{-6} \leq \xi \leq 1.0$
3	$\begin{cases} n_{1,1} + n_{1,2} + \xi = 0.354 \\ n_{2,1} + n_{2,2} + \xi = 0.183 \\ n_{3,1} + n_{3,2} + 2\xi = 0.463 \\ n_{4,1} + n_{4,2} - 2\xi = 0 \end{cases}$	$10^{-6} \leq n_{i,j} \leq 1.0$ $10^{-6} \leq \xi \leq 1.0$
4	$\begin{cases} n_{1,1} + n_{1,2} + \xi = 0.3 \\ n_{2,1} + n_{2,2} + \xi = 0.4 \\ n_{3,1} + n_{3,2} - \xi = 0.3 \\ n_{4,1} + n_{4,2} - \xi = 0 \end{cases}$	$10^{-6} \leq n_{i,j} \leq 1.0$ $10^{-6} \leq \xi \leq 1.0$
5	$\begin{cases} n_{1,1} + n_{1,2} + \xi = 0.6 \\ n_{2,1} + n_{2,2} + \xi = 0.4 \\ n_{3,1} + n_{3,2} - \xi = 0 \end{cases}$	$10^{-6} \leq n_{i,j} \leq 1.0$ $10^{-6} \leq \xi \leq 1.0$
6	$\begin{cases} n_{1,1} + n_{1,2} + \xi = 0.1 \\ n_{2,1} + n_{2,2} + \xi = 0.15 \\ n_{3,1} + n_{3,2} + 2\xi = 0.7 \\ n_{4,1} + n_{4,2} - 2\xi = 0 \\ n_{5,1} + n_{5,2} = 0.05 \end{cases}$	$10^{-6} \leq n_{i,j} \leq 1.0$ $10^{-6} \leq \xi \leq 1.0$
7	$\begin{cases} n_{1,1} + n_{1,2} + \xi = 0.52 \\ n_{2,1} + n_{2,2} + \xi = 0.48 \\ n_{3,1} + n_{3,2} - \xi = 0 \end{cases}$	$10^{-6} \leq n_{i,j} \leq 1.0$ $10^{-6} \leq \xi \leq 1.0$
8	$\begin{cases} n_{1,1} + n_{1,2} + \xi = 0.048 \\ n_{2,1} + n_{2,2} + \xi = 0.5 \\ n_{3,1} + n_{3,2} - \xi = 0.452 \\ n_{4,1} + n_{4,2} - \xi = 0 \end{cases}$	$10^{-6} \leq n_{i,j} \leq 1.0$ $10^{-6} \leq \xi \leq 1.0$

4.3 ALGORITHMS AND SOFTWARE ARCHITECTURE

We now turn to the basic structure of the implemented code. Figure 8 is a simplified UML (Unified Modeling Language) Class Diagram. It describes the code in terms of its classes and structures and also shows how these elements relate to one another. In these diagrams, boxes represent classes (or structures).

The name of the class appears in the first field, from top to bottom. It is followed, in the second field, by a list of attributes pertaining to that particular class. The notation for attributes is (+/-) attributeName: attributeType. When preceded by a (+) sign, an attribute is said to be public, i.e., it can be accessed by objects or methods external to the class. When preceded by a (–) sign, it is said to be private, which is to say that it can only be accessed by the very object that owns it. For example, in the *Interval* class, there is an attribute named *a*, which is private and takes a double-precision numerical value. The next field comprises the methods (or functions) implemented by that class. The notation is (+/-) methodName(arg_1_Type, arg_2_Type ,…): returnType. The (+/-) has the same meaning as before. If the method returns a value, its type is represented by returnType. The types of each argument of the function are represented by arg_1_Type, arg_2_Type, and so on.

UML diagrams indicate relations between classes with connectors. Simple line connectors indicate general *associations*. Black diamond-shaped connectors denote *composition*: the class whose box is closest to the diamond head of the connector implements the class on its tail. Moreover, if an object of the former class is destroyed, all objects belonging to the latter class that were implemented by that same object are also destroyed. Numbers over the connectors and close to the classes indicate how many objects of each class are involved in that relation. An asterisk (*) indicates that zero or more objects may be involved. For example, the composition connector between Solver and PriorityQueue has two 1's above it and close to each class. It should be read as: one Solver object implements one PriorityQueue object and the latter is destroyed when the former is. A much more thorough discussion on UML can be found in standard software engineering references, such as (Sommerville, 2011).

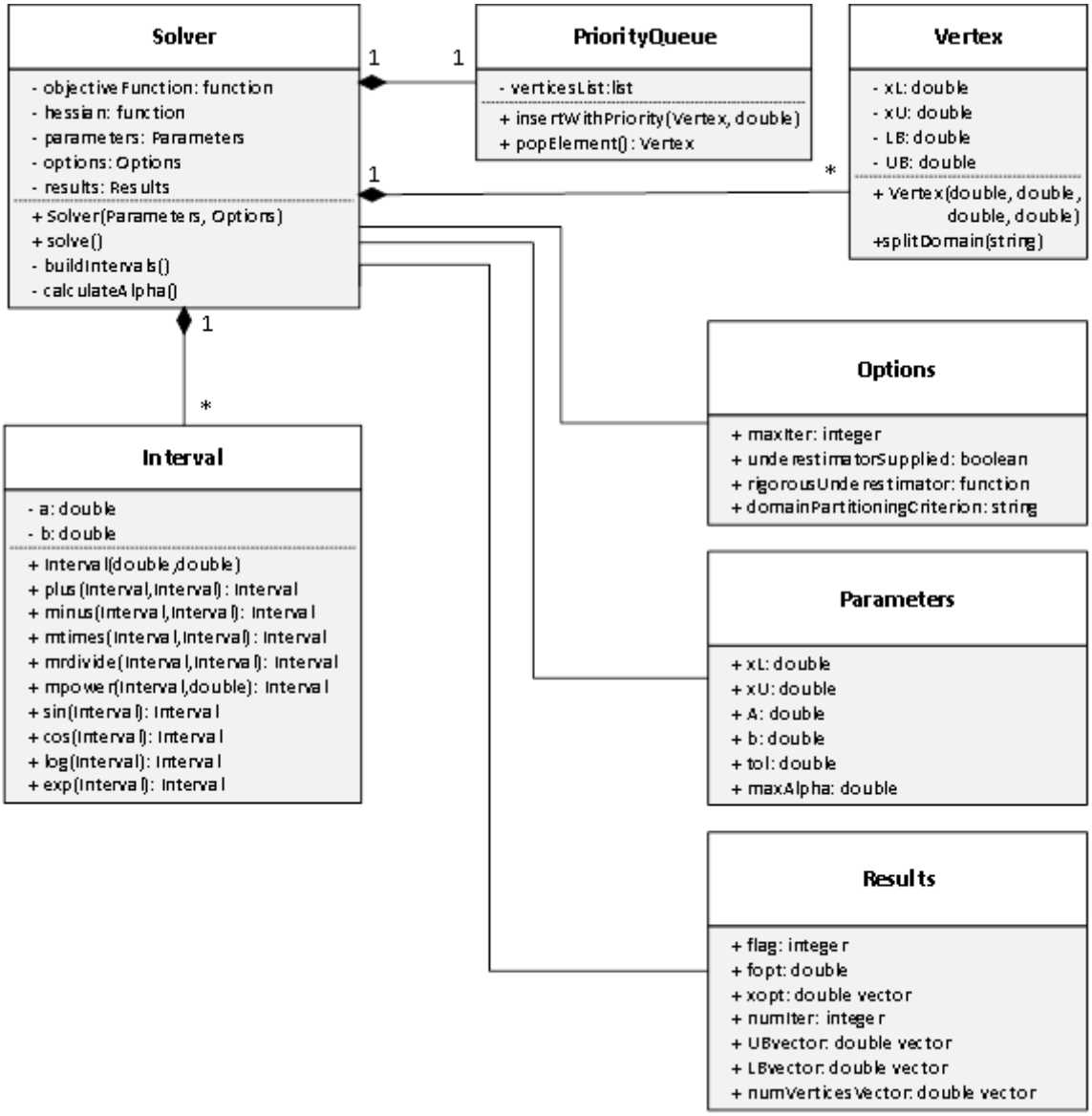


Figure 8 - Simplified UML class diagram for the implemented optimization routine.

We can now proceed to describing each class.

4.3.1 Solver class

The Solver class is where most of the computation takes place. It is responsible for reading the inputs describing an optimization problem. The Parameters structure carries information such as:

Table 4 - Meaning of the attributes carried by the Parameters structure to the Solver class.

Attribute name	Meaning
objectiveFunction	Objective function
hessian	Hessian of the objective function
xL	Lower bounds vector
xU	Upper bounds vector
A	Equality constraints matrix
B	Equality constraints vector
tol	Error tolerance

The Options structure allows the user to tune how the algorithm runs. The following table describes each of its attributes:

Table 5 - Meaning of the attributes carried by the Options structure to the Solver class.

Attribute name	Meaning
maxIter	Maximum number of iterations before the algorithm halts. *Default = 1000.*
underestimatorSupplied	Boolean attribute indicating whether a rigorous convex underestimator has been supplied. If this attribute is true, the algorithm skips the routines corresponding to the calculation of α and uses the supplied underestimator instead. *Default = False*.
rigorousUnderestimator	Rigorous convex underestimator (if available)
domainPartitioningCriterion	A string attribute that indicates which is the criterion for deciding around which variable the domain will be split. The options are: • MaximumAbsolute (*Default*): The variable with the greatest distance between its lower and upper bounds is chosen.

	• MaximumRelative: The variable with the greatest normalized distance between its lower and upper bounds is chosen. The normalization term is the initial distance between lower and upper bounds.

Once the calculations are finished, the output data is stored in a public structure named Results. This structure collects a flag that informs whether the algorithm has converged or not and collects output data for plotting:

Table 6 - Meaning of the attributes carried by the Options structure to the Solver class.

Attribute name	Meaning
flag	Maximum number of iterations before the algorithm halts. If the algorithm converges, flag = 1. If the maximum number of iterations is exceeded, flag = 0. In any other case, flag = -1.
fopt	Objective function's optimum value found by the algorithm.
xopt	Choice variables vector corresponding to the optimum value.
numIter	Number of iterations performed by the algorithm.
LBVector	Vector of lower bounds at each iteration.
UBVector	Vector of upper bounds at each iteration.
numVerticesVector	Vector containing the total number of vertices at each iteration.

The actual implementation of the solve method will be presented in a later section.

4.3.2 Interval class

The Interval class implements basic interval arithmetic functionality necessary for the α's to be calculated by Gershgorin's theorem for interval matrices (in case no rigorous convex underestimators are provided). We achieved this result by overloading basic arithmetic operators and defining basic functions on Interval objects. We defined the overloaded operations in such a way that they mimic the corresponding definitions of interval arithmetics, as shown earlier in the section titled *Interval analysis* of this work.

We have further overloaded the following operators to satisfy the following equalities, which greatly simplifies calculations. If k is a real number (or a double precision floating-point number, if we think in terms of the actual implementation):

$$k[a,b] = [ka, kb]$$

$$k + [a,b] = [a + k, b + k] \tag{90}$$

Even though trigonometric functions are not present in any of the activity coefficient models considered, we also had to implement the sine and cosine functions because some of the benchmark functions used to test the correctness of our code (to be described later) required them. We do that by noticing that the sine function is monotonic in the intervals $[-\pi/2, \pi/2]$. If we also notice that: a) whenever the points of the form $\frac{\pi}{2} + 2k\pi$, the interval's upper value is set to 1; and b) whenever the points of the form $\frac{3\pi}{2} + 2k\pi$, the interval's lower value is set to -1, we can naturally extend the sine function for any interval. A parallel argument can be made for the cosine function.

Once the interval functionality was implemented, it was possible to supply interval objects instead of floating numbers to the analytic Hessian matrices – see Appendix D – and, in so doing, obtain interval matrices. These interval matrices could then be used in conjunction with Gershgorin's theorem to find the required α values, as explained in the *αBB (α-Branch-and-Bound)* section.

4.3.3 Vertex class

The Vertex class represent the vertices in the binary tree generated during the execution of the αBB algorithm. Each vertex represents a subdomain to be searched by the code. The following table summarizes and explains its attributes:

Table 7 - Meaning of the attributes of Vertex class.

Attribute name	Meaning
xL	Vector of lower bounds on the subdomain corresponding to the current vertex.
xU	Vector of upper bounds on the subdomain corresponding to the current vertex.
LB	Lower bound found by solving the convex optimization subproblem at the subdomain corresponding to the vertex.
UB	Upper bound found by solving the nonconvex optimization subproblem at the subdomain corresponding to the vertex.

The vertex class implements the method splitDomain, which takes a string. This string represents the criterion chosen for deciding about which variable the domain should be split. This string is provided by the Solver object, which delivers its attribute domainPartitioningCriterion as an input to the vertices that it creates. The methods creates two new vertices, each one corresponding to one section of the split domain, and returns them.

4.3.4 PriorityQueue class

As the name suggests, this class implements (naively) the functionality of the Priority Queue data structure. It is essentially a container that allows new elements to be placed in it alongside with a number that is its priority, and for its elements to be withdrawn from it, one at a time. The order in which the elements are withdrawn is dictated by the priorities of the elements:

element set to come out is always the one with highest (or lowest) priority. In our work, our priority queues set the elements with lower priorities to come out first and the elements to be stored are Vertex objects.

We opted for a naïve implementation, which stores the elements in a list, verticesList. In this implementation, the elements are added to the list in $\Theta(1)$ time by the method insertWthPriority. In order to withdraw, or pop, an element, the class performs a linear search on the list until it finds the vertex with lowest priority, which takes $\Theta(n)$ time, where n is the number of elements in the list. The method responsible for this action is pop_element. We chose this implementation due to its simplicity and to the fact that this list does not usually become too large.

4.3.5 The solve method

We are now in conditions to describe how the solve method, implemented by the Solver class, was coded. The pseudo code is presented below.

Read the objective function, its hessian and its convex underestimator, if the latter is provided: $objectiveFunction$, $hessian$ and $convexUnderestimator$.
If a rigorous convex underestimator is provided, read the maximum permissible value for α: $alphaMax$
Read the matrix and vector associated to the linear constraints - A and b, respectively
Read the specified tolerance: tol
Read the domain partitioning criterion: $criterion$
Read the specified upper and lower bounds for the decision variables: xL and xU.
Initialize upper and lower bounds for the optimum objective function value: $LB = -\infty$, $UB = +\infty$.
Initialize iterations counter and flag: $numIter = 0$, $flag = -1$
Create a vertex $v0$ whose xL, xU, LB and UB are the ones just defined.
Create a PriorityQueue $priorityQueue$ and add $v0$ to it.
While $UB - LB > tol$
    If $numIter > iterMax$
        Set $flag = 0$
        Return

$currentVertex = priorityQueue.pop()$

$LB = currentVertex.LB$

$[v1, v2] = currentVertex.splitDomain(criterion)$

Run a feasibility program on the region determined by $v1$ and the constraint matrices A and b.

If the region determined by $v1$ is feasible

If a convex underestimator has been provided:

Use it as an underestimator;

Else

$intervals1 = buildIntervals(v1.xL, x1.xU)$

$alpha1 = calculateAlpha(intervals1, hessian)$

If $alpha1 > alphaMax$

$alpha1 = alphaMax$

Set the underestimator to $objectiveFunction + alpha1\ (v1.xL - x)^T(v1.xU - x)$;

Solve the convex underestimating problem with initial guess set to $\frac{1}{2}(v1.xL + v1.xU)$;

Run a feasibility program on the region determined by $v2$ and the constraint matrices A and b.

If the region determined by $v2$ is feasible

If a convex underestimator has been provided:

Use it as an underestimator;

Else

$intervals2 = buildIntervals(v2.xL, x2.xU)$

$alpha2 = calculateAlpha(intervals2, hessian)$

If $alpha2 > alphaMax$

$alpha2 = alphaMax$

Set the underestimator to $objectiveFunction + alpha2\ (v2.xL - x)^T(v2.xU - x)$;

Solve the convex underestimating problem with initial guess set to $\frac{1}{2}(v2.xL + v2.xU)$;

Set $flag = 1$

4.4 TESTING

During the development of the code, we followed the TDD (test-driven development) methodology, which means that we wrote tests before the actual code. Once the programmer writes a sufficient number of tests corresponding to certain desired functionality, he or she can then write the actual code and make sure that it meets the requirements. One of the advantages of following this approach is that whenever the programmer needs to refactor the code, he or she can easily check whether any functionality was lost in the process. We have made use of Matlab's Unit Testing Framework to implement and run all tests.

We can divide our tests in three categories, according to their goals:

1. Validate the implementation of the data structures.
2. Check the correctness and robustness of the code (benchmark functions)
3. Check the code's ability to solve actual equilibrium problems (benchmark equilibrium problems).

We will not detail the tests pertaining to the first category, as they are numerous and straightforward.

In order to ensure the implementation's correctness and robustness, we ran the code through a series of benchmark functions, whose optimum values were known in advance, and could be therefore be used to gauge the behavior of the algorithm. Table 8 displays the benchmark functions used, along with their definitions, constraints, global optimum values and a brief description of them.

Table 8 - Test functions used to evaluate code correctness.

Function (number of variables)	**Definition**	**Constraints**	**Optimum value**	**Description**
Simple quadratic (1)	$f(x) = x^2$	$x \in [-1, 1]$	$f(x^*) = 0$ $x^* = 0$	Simple convex function

Edge quadratic (1)	$f(x) = x^2$	$x \in [1, 3]$	$f(x^*) = 1$ $x^* = 1$	Minimum at the edge of the domain.
Linear-senoid (1)	$f(x) = x \sin x$	$x \in [0, 10]$	$f(x^*) = -5.44$ $x^* = 10$	Multiple local minima.
Multivariate quadratic (2)	$f(x) = x_1^2 + x_2^2$	$x_i \in [-1, 1]$	$f(x^*) = 0$ $x^* = (0,0)$	Multivariate convex function.
Rastrigin (2)	$f(x) =$ $20 + x_1^2 - 10\cos(2\pi x_1) +$ $x_2^2 - 10\cos(2\pi x_2)$	$x_i \in$ $[-5.12, 5.12]$	$f(x^*) = 0$ $x^* = 0$	Several local minima.
Rosenbrock function (2)	$f(x) = (1 - x_1)^2 +$ $100(x_2 - x_1^2)^2$	$x_i \in [0, 2]$	$f(x^*) = 0$ $x^* = (1,1)$	Global minimum in a narrow valley
Beale's function (2)	$f(x) = (1.5 - x_1 + x_1 x_2)^2$ $+$ $(2.25 - x_1 + x_1 x_2^2)^2 +$ $(2.625 - x_1 + x_1 x_2^3)^2$	$x_i \in$ $[-4.5, 4.5]$	$f(x^*) = 0$ $x^* = (3,0.5)$	Sharp peaks around the minimum.

Once the algorithm was able to pass all the tests displayed in Table 8, we moved on to the actual chemical and phase equilibrium problems discussed in the previous section.

5 RESULTS AND DISCUSSION

In this chapter, the results obtained when running the algorithm for the benchmark equilibrium problems are presented.

For each equilibrium problem, the following results were obtained and will displayed as graphs:

- Number of iterations required for the algorithm to halt for a given tolerance (ε);
- Moving average of the number of vertices being examined by the algorithm as a function of the iterations count;
- The gap between the upper bound found at the current iteration and the best lower bound found thus far.
- A box plot of the logarithm of the values of the parameter α calculated by the algorithm.

The graphs are shown in the figures ranging from Figure 9 to Figure 40.

5.1 CONVERGENCE ANALYSIS

As it would be expected, the number of iterations necessary for the algorithm to halt increases as the tolerance, ε, diminishes. That is because in order to attain ϵ -convergence for smaller and smaller values of ϵ, the algorithm must further subdivide the domain into smaller subdomains. The convex underestimators become tighter as the subdomains become smaller, which implies a smaller difference between the optimal value found for the convex underestimator problem and the actual optimal value. As the subdivision process has been carried out for a sufficient number of iterations, ϵ -optimality can then be reached.

It is interesting to notice that the number of iterations required by the algorithm seems to remain constant as ϵ diminishes for some cases, a fact which is very noticeable in Problem 4 (UNIQUAC model) and in Problem 3 (Wilson model). In Problem 4, in the curve associated to $\alpha_{max} = 10$, as $\log_{10} \epsilon$ transitioned from -2 to -3.5, the number of iterations hasn't changed. That is because the number of iterations necessary to attain ϵ-optimality corresponding to $\log_{10} \epsilon = -2$ was also sufficient to attain ϵ -optimality corresponding to $\log_{10} \epsilon = -3.5$. The

same happens in Problem 3, in the curve associated to $\alpha_{max} = 10$, as $\log_{10} \epsilon$ transitioned from -1.5 to -3.5.

Another feature of these graphs is that the curves corresponding to $\alpha_{max} = 10^3$ and $\alpha_{max} = 10^4$ overlapped in problems 1, 3 and 6. That happened because $\alpha_{max} = 10^3$ was high enough that whenever the algorithm replaced the calculated α with α_{max}, the same happened to $\alpha_{max} = 10^4$. That had the effect of essentially making the algorithm perform the same search path along the subdomains graph.

Upon comparing the iterations curves, it is possible to see that in problems 1 (NRTL), 2 (Wilson), 4 (UNIQUAC) and 7 (Margules) the maximum number of iterations necessary for the algorithm to converge corresponds to $\alpha_{max} = 10^4$ and $\log_{10} \epsilon = -3.5$. This result is not surprising, as it would be expected that larger values of α_{max} would yield looser underestimators, which would cause the algorithm to need to run more iterations for a fixed ϵ. Also, smaller values of ϵ would demand a more thorough partitioning and, thus, more iterations. Surprisingly, in problem 3 (Wilson), the maximum number of iterations corresponds to $\alpha_{max} = 10$ and $\log_{10} \epsilon = -3.5$. The explanation for this anomaly may be that, due to the topology of this particular problem, looser underestimators, corresponding to higher values of α_{max}, steered the algorithm towards the global optimum and strongly biased the search process to search in that particular region. In problems 5 (Margules) and 6 (Wilson), the maximum number of iterations also corresponds to a relatively low α_{max}, more precisely, $\alpha_{max} = 10^2$ and $\log_{10} \epsilon = -3.5$.. In problem 8 (NRTL), it corresponds to $\alpha_{max} = 10^3$and $\log_{10} \epsilon = -3.5$. The reason for these is probably the same stated for problem 3. Despite these differences, the maximum number of iterations always corresponds to $\log_{10} \epsilon = -3.5$, which shows that the tolerance ϵ plays a major role in determining its number.

In problems 3 (Wilson), 4 (UNIQUAC), 5 (Marugles), 6 (Wilson) and 8 (NRTL), the logarithm of the maximum number of iterations taken for the algorithm to converge (all corresponding to $\log_{10} \epsilon = -3.5$, but not necessarily corresponding to different values of α_{max}) ranged from 2.2 to 2.6. Problem 1 (NRTL) took the greatest number of iterations, corresponding to a logarithm that was greater than 3. Problem 7 (Margules) also took a high number of iterations, corresponding to a logarithm of more than 2.6 and problem 2 (Wilson) was converged in the smallest number of iterations, corresponding to a logarithm of less than 2.2. From these data, it is possible to see that the overall complexity of the model alone is not necessarily an indicator of the maximum number of iterations to be taken by the algorithm.

The graphs showing the gap between upper and best lower bounds provide insight into how the algorithm converges to the optimum. All such graphs display similar features. It is straightforward to check that the curves represented therein are monotonically decreasing, that is, they either decrease or remain constant.

Two factors are responsible for the drop in the gap between the upper bound and the best lower bound. The first one is the effect associated to subdividing the domain further and further. As the subdomains become smaller and smaller, the underestimating function over that subdomain also approaches the objective function, as can be seen from the relation $L(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{n} \alpha\left(x_j^L - x_j\right)\left(x_j^U - x_j\right)$. Small subdomains amount to smaller values of $\left(x_j^L - x_j\right)\left(x_j^U - x_j\right)$ for every j, which leads to tighter underestimators. The second factor is associated to how close a given subdomain is to the global optimum. Small subdomains that contain the global optimum are much more likely to yield tighter lower bounds, which contributes to diminishing the gap.

During the first iterations, the greater the maximum value of α_{max}, the greater the gap, as would be expected. As the algorithm proceeds, at around 10 to 100 iterations, a sharp drop can be seen in all graphs. This drop reflects mostly the effect of subdividing the domain over the gap. From that point on, the gap diminishes at a much lower rate. That is due to the fact that, as all subdomains have become relatively small, the effect of subdividing further has become secondary when compared to the actual position of the subdomains. In this region, not only does the gap diminishing rate drop, but it actually becomes zero at times, which leads to the gap remaining constant. It then only diminishes again when a subdomain is particularly well located. This also explains why the relative ordering of the curves changes as the iterations progress: once the subdomains become small enough, the influence of α_{max} over the quality of the lower bounds decreases in contrast to the positioning of the subdomain. From that point on, there is no straightforward way of telling which run of the algorithm will first reach the best subdomains first.

Table 9 – Number of iterations necessary for the upper bound to converge to the global minimum.

Problem	$\log_{10} \alpha_{max}$			
	1	**2**	**3**	**4**
1	8	8	8	8
2	7	7	7	7
3	2	2	2	2
4	3	3	2	2
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	7	7	7	7

It is very insightful to compare how many iterations are necessary for the upper bounds to reach the global optimum, as shown in Table 9, and to contrast them to the total number of iterations taken. Notice that it takes no more than 8 iterations for the upper bound to converge. Of course, it is the comparison between the upper and lower bounds that allows for the αBB to be able to guarantee global optimality, and merely keeping track of the upper bound would be no better than simply using a local solver such as Newton's method. From a practical standpoint, however, it is the case that the systematic scanning of the domain, with subsequent use of local solvers, was enough to handle the nonconvexities of the problems under study. That also explains why the adapted αBB algorithm under study, despite not being globally convergent, has been successful in reaching global optima.

It is also interesting to confront the numbers of iterations shown in Table 9 with the activity models corresponding to each problem. For instance, problems 5 and 7, which are associated to the Margules model – which is relatively simple – required only one iteration for the upper bound to reach the global optimum. Contrast it with problems 1 and 8, which are associated to the considerably more complex NRTL model. It took them, respectively, 8 and 7 iterations for the upper bound to reach the global optimum, which is considerably more than it took the last two. Problems 2, 3 and 6, associated to the Wilson model, displayed values which are between those of Margules and NRTL, with the remark that it took problem 2 the same number of iterations as problem 1. This might be due to the fact that the local underestimators generated by interval analysis had lower calculated minima in the subdomains that actually contained the global optimum. That would have lead the algorithm to steer away from the global optimum

during its first iterations, which would explain the high number of iterations. Interestingly, problem 4, associated to the UNIQUAC model, converged in only 3 iterations for the two lowest values of $\log_{10} \alpha_{max}$ and in 2 iterations for the two highest. That is surprising because the UNIQUAC model is quite complex and, intuitively, one might have expected that many iterations would have been necessary until the global optimum was found. However, as conjectured before, the calculated underestimators might have been such that the algorithm was steered towards the global optimum. The same factor that hindered problem 2 may have helped problem 4.

It seems reasonable to point out that models that are more complex tend to be associated, although not always, to smaller numbers of iterations necessary for the upper bound to converge to the global optimum.

The fact that the number of iterations did not change with $\log_{10} \alpha_{max}$ is also noteworthy. The first iterations of the algorithm yield very high values of α, which means that they were mostly replaced by α_{max}. If that is the case, the underestimators associated to each subdomain will differ in magnitude, but the order in which that they are selected will be preserved. For this reason, the iteration values remain constant as $\log_{10} \alpha_{max}$ varies.

5.2 GRAPH SEARCH

For clarity's sake, instead of merely plotting the number of vertices being analyzed by the algorithm versus iterations count (the graphs would be far too cluttered to read), a plot of its moving averages was constructed. The lags (the number of consecutive values being averaged at each point) corresponding to these moving averages are shown in figures 10, 14, 18, 22, 26, 30, 34 and 38. It is very interesting to notice that the number of vertices being examined remains relatively low, considering that the search tree could grow exponentially large. That shows that many vertices are being fathomed by the algorithm, as would be desired. This property is already present in the original αBB algorithm and, as shown, has not been compromised in its current adaptation.

Upon analyzing the graphs, it can be seen that, as a general rule, lower values of α_{max} yield lower mean number of vertices. That is explained by the fact that lower values of α_{max} lead to

tighter underestimators, which in turn leads to higher values of the best lower bounds, thus contributing to the fathoming of more nodes.

In problems 1 and 8, which made use of the NRTL model, the average number of vertices reached values between 12 and 15, when considering $\alpha_{max} = 10^4$,which is higher than the corresponding values of the other problems. That may be because the inherent complexity of the model yielded not very tight underestimators, which forced the algorithm to keep several vertices under examination. That seems to be a plausible explanation, as the same happened to problem 4, corresponding to the UNIQUAC model, which is also quite complex. The average number of vertices rose up to values between 10 and 12 as the iterations progressed. Problems 5 and 7, corresponding to the simpler Margules model, had an average number of vertices that rose up to the range between 5 and 10. Problems 2, 3 and 6, corresponding to the Wilson model displayed results comparable to those of the Margules model. It seems, therefore, plausible to state that the mean number of vertices produced by the algorithm as the iterations progress is associated to the complexity of the activity model: simpler models yield lower numbers of vertices, whereas more complex models yield higher numbers of vertices.

It is also interesting to notice that these differences are far more well-defined for higher values of α_{max} than for lower values, even though they are still observable.

5.3 UNDERESTIMATORS AND CALCULATED α VALUES

The box plots displaying the logarithms of the calculated α values provide important insights into how the algorithm behaves. In these graphs, the red crosses represent outliers. A very interesting property is that the values of α calculated by the algorithm can be extremely large, reaching values as high as 10^{90} as is the case of problem 4 (UNIQUAC model). Such unreasonably high values justify our approach, which is to limit these values to some α_{max}. In fact, the algorithm cannot proceed with such high values of α, as that would lead to very slow convergence times and low precision, due to numerical underflow / overflow.

These values get so high because of two factors: a) the analytic evaluation of the hessians leads to many operations, which contributes to widen the intervals; b) as can be seen in Appendix D, the calculations of the second derivatives involve molar numbers raised to the 4th power in the

denominator. That is particularly problematic when small lower bounds for the mole numbers (such as, in our case, 10^{-6}) are considered because that yields a high multiplicative factor.

Upon analyzing figures 12, 16, 20, 24, 28, 32, 36 and 40, a few facts become noticeable. In problems 1 (NRTL), 2 (Wilson), 3 (Wilson), 6 (Wilson) and 8 (NRTL) the median of $\log_{10}\alpha$ was lesser than 10 and the box plots had relatively narrow interquartile ranges. That shows that, as expected, the algorithm tends to focus on subdomains that yield higher values of lower bounds which generally correspond to lower values of α. Problem 4 (UNIQUAC) displayed an interesting behavior: its median $\log_{10}\alpha_{max}$ sharply dropped as $\log_{10}\alpha$ increased. That is probably due to the fact that as $\log_{10}\alpha_{max}$ increases, the looser the underestimators will be, and, adding that to the fact that as the domains are subdivided, the tighter the underestimators are, it follows that as $\log_{10}\alpha_{max}$ grows bigger, the greater the priority is given to small subdomains, which leads the algorithm to focus on them. That, in turn, steers the median value of $\log_{10}\alpha$ towards lower values. Interestingly, problems 5 (Margules) and 7 (Margules) displayed relatively wide interquartile ranges, especially for lower values of $\log_{10}\alpha_{max}$. That may be explained by an argument parallel to that of problem 4: as the Margules underestimators tend not to be as loose, the algorithm does not focus as much in small particular subdomains, and a wider spread of the values of $\log_{10}\alpha$ is therefore observed.

5.3.1 Problem 1

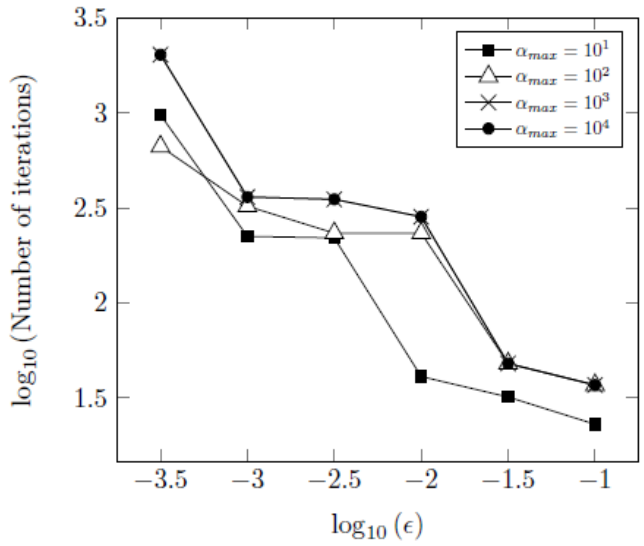


Figure 9 – Logarithm of the number of iterations versus logarithm of the error tolerance for benchmark equilibrium problem 1.

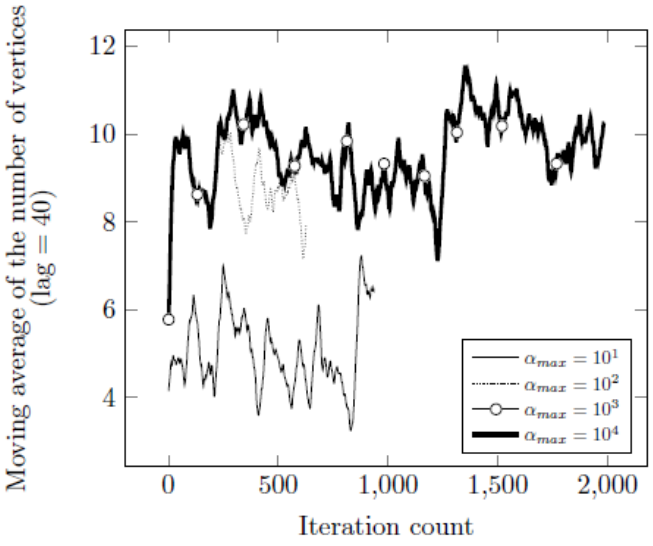


Figure 10 – Moving average of the number of vertices under consideration as the algorithm progresses for benchmark equilibrium problem 1.

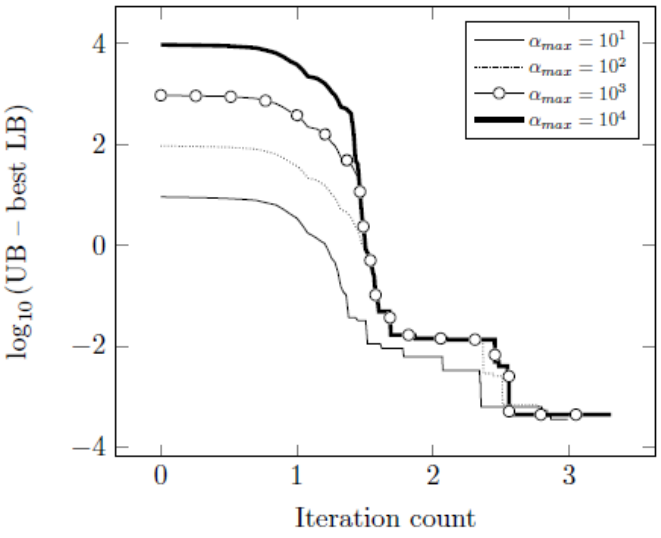


Figure 11 - Upper bound / best lower bound gap versus number of iterations for benchmark equilibrium problem 1.

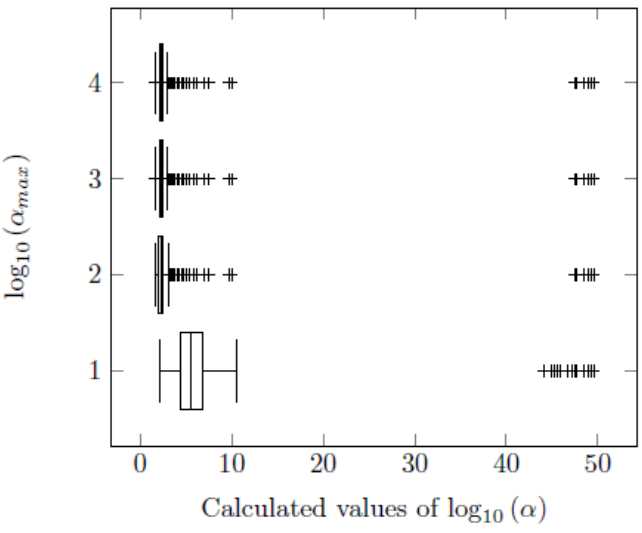


Figure 12 – Box plot of the logarithm of all the values of α calculated through interval analysis for benchmark equilibrium problem 1.

5.3.2 Problem 2

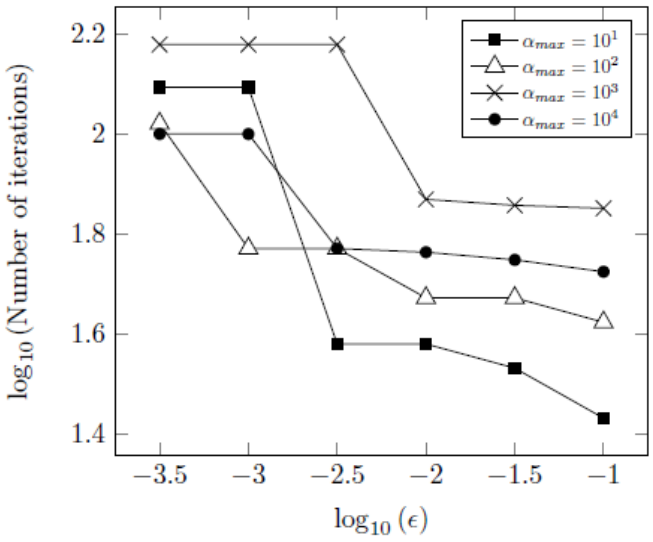


Figure 13 – Logarithm of the number of iterations versus logarithm of the error tolerance for benchmark equilibrium problem 2.

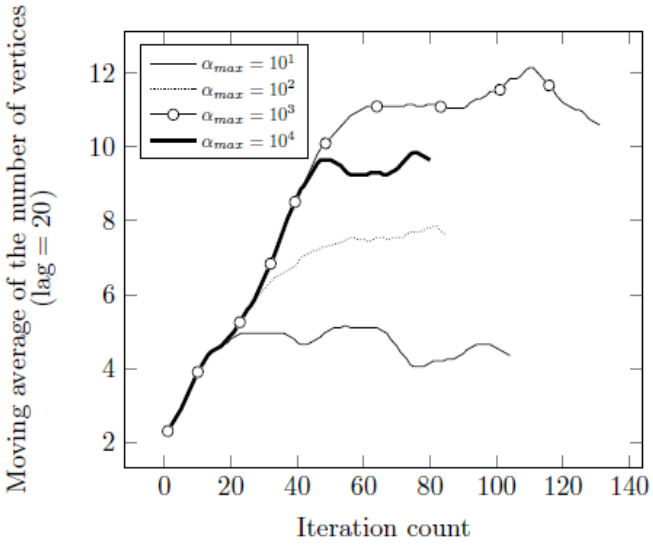


Figure 14 – Moving average of the number of vertices under consideration as the algorithm progresses for benchmark equilibrium problem 2.

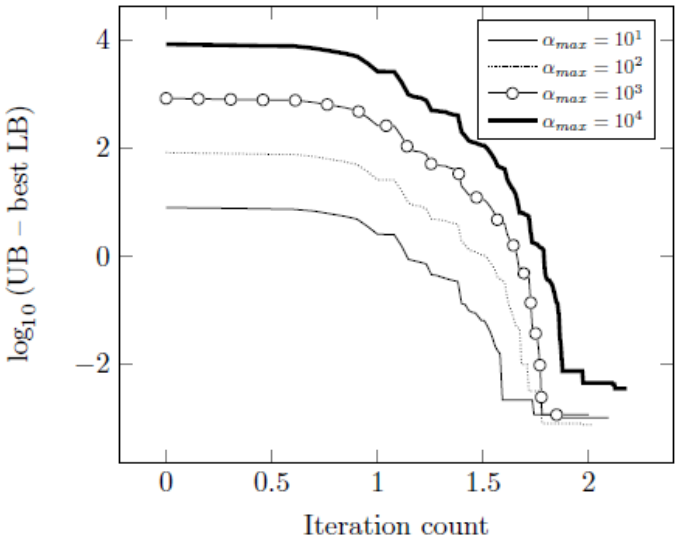


Figure 15 - Upper bound / best lower bound gap versus number of iterations for benchmark equilibrium problem 2.

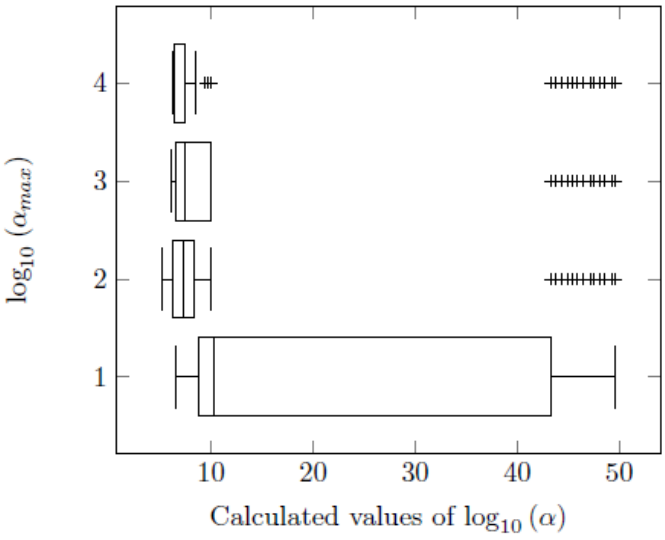


Figure 16 – Box plot of the logarithm of all the values of α calculated through interval analysis for benchmark equilibrium problem 2.

5.3.3 Problem 3

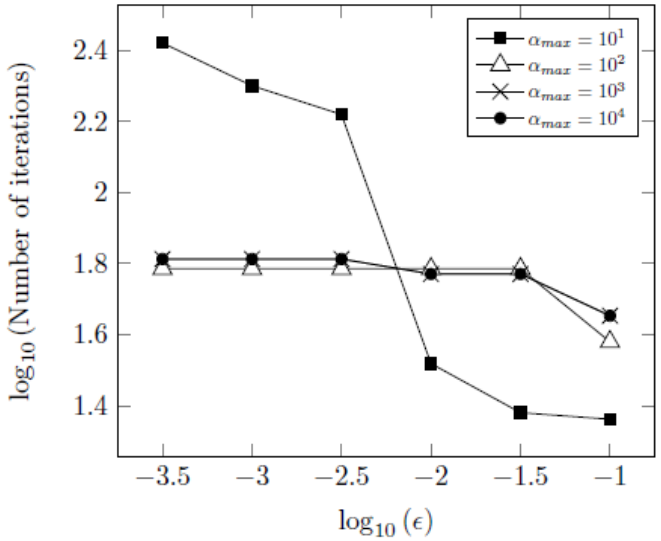


Figure 17 – Logarithm of the number of iterations versus logarithm of the error tolerance for benchmark equilibrium problem 3.

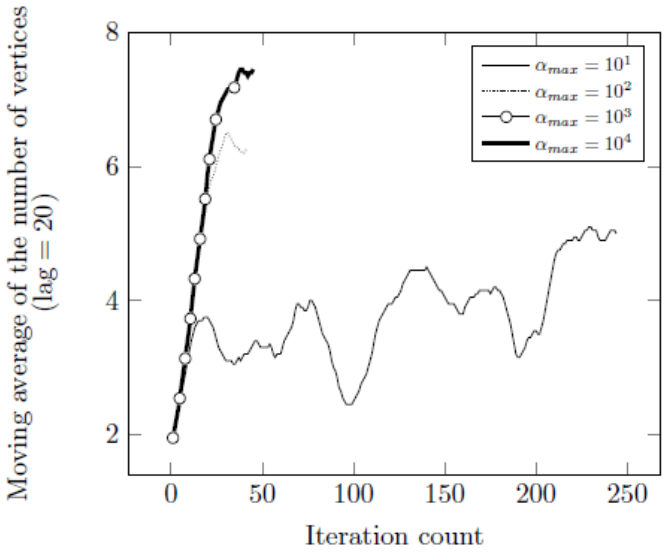


Figure 18 – Moving average of the number of vertices under consideration as the algorithm progresses for benchmark equilibrium problem 3.

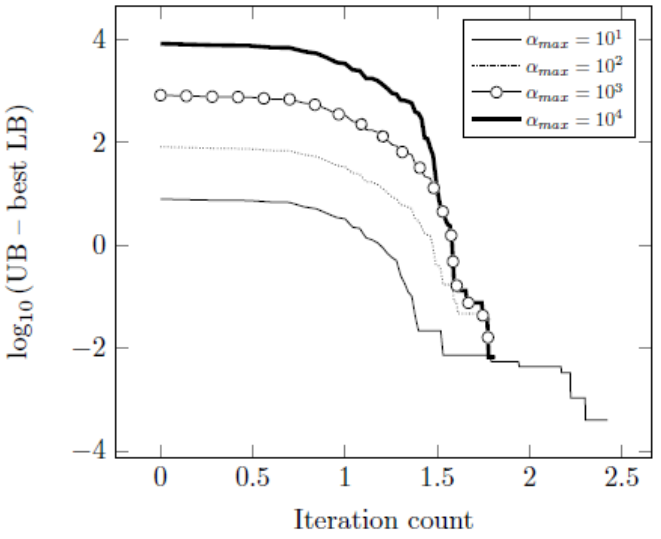


Figure 19 - Upper bound / best lower bound gap versus number of iterations for benchmark equilibrium problem 3.

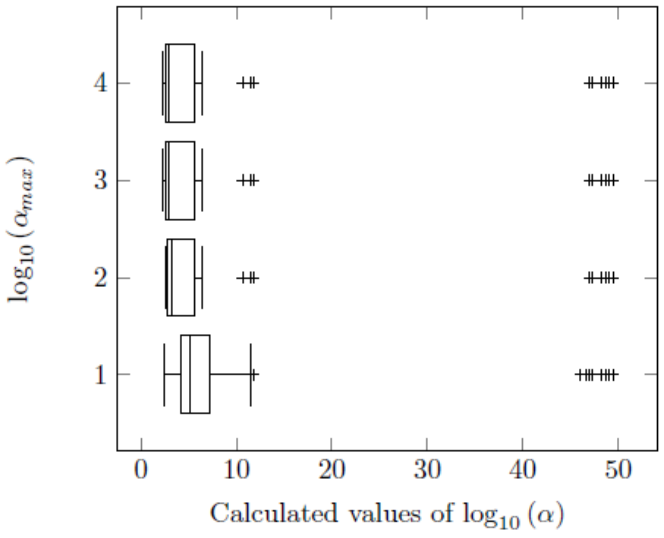


Figure 20 – Box plot of the logarithm of all the values of α calculated through interval analysis for benchmark equilibrium problem 3.

5.3.4 Problem 4

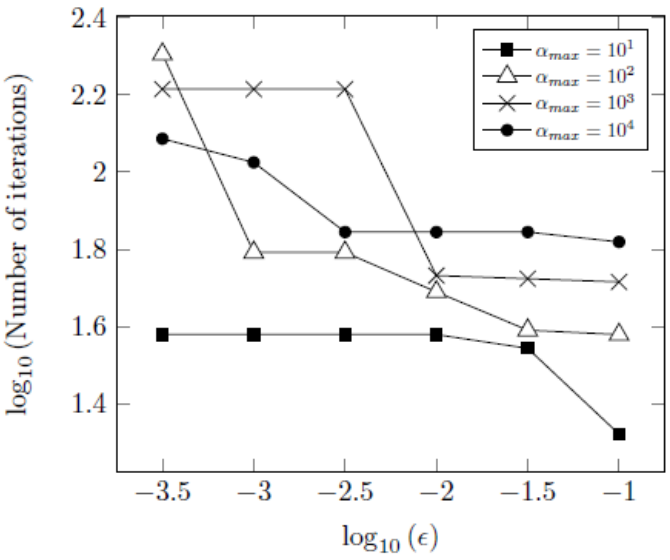


Figure 21 – Logarithm of the number of iterations versus logarithm of the error tolerance for benchmark equilibrium problem 4.

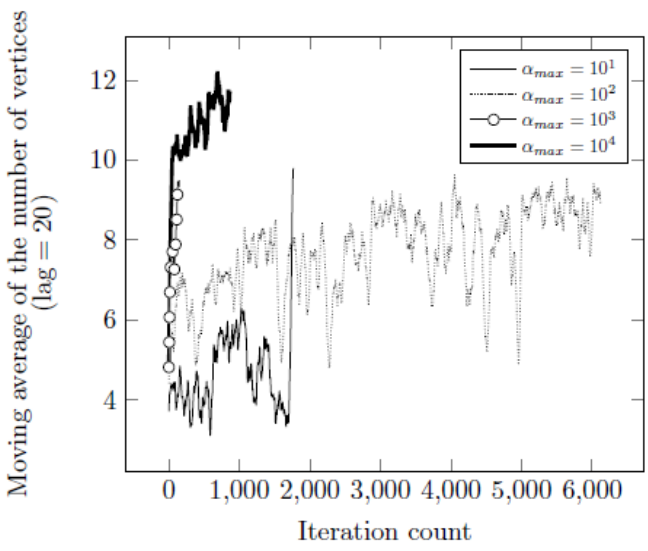


Figure 22 – Moving average of the number of vertices under consideration as the algorithm progresses for benchmark equilibrium problem 4.

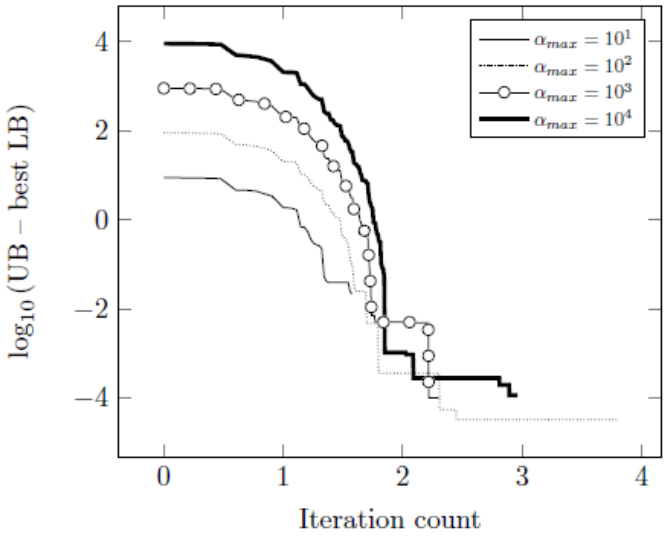


Figure 23 - Upper bound / best lower bound gap versus number of iterations for benchmark equilibrium problem 4.

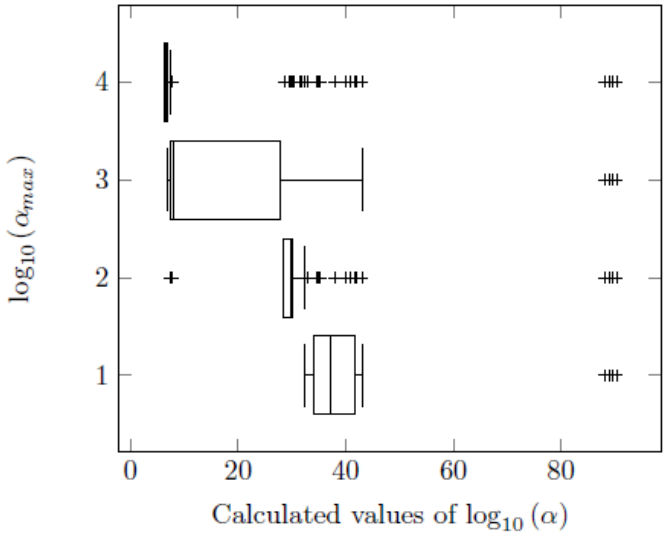


Figure 24 – Box plot of the logarithm of all the values of α calculated through interval analysis for benchmark equilibrium problem 4.

5.3.5 Problem 5

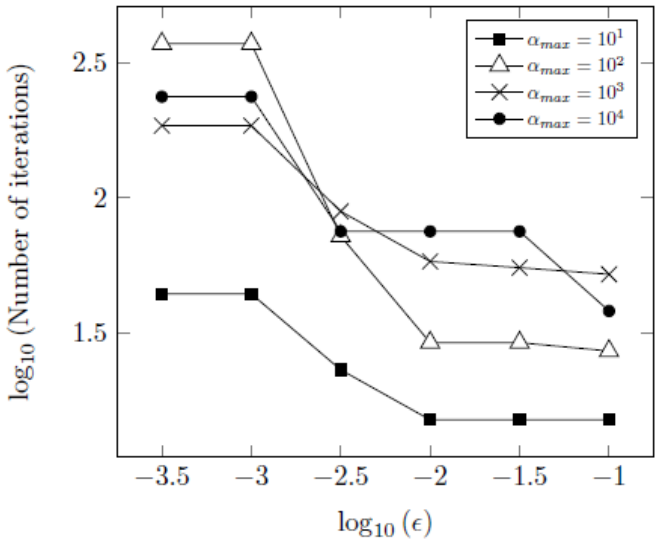


Figure 25 – Logarithm of the number of iterations versus logarithm of the error tolerance for benchmark equilibrium problem 5.

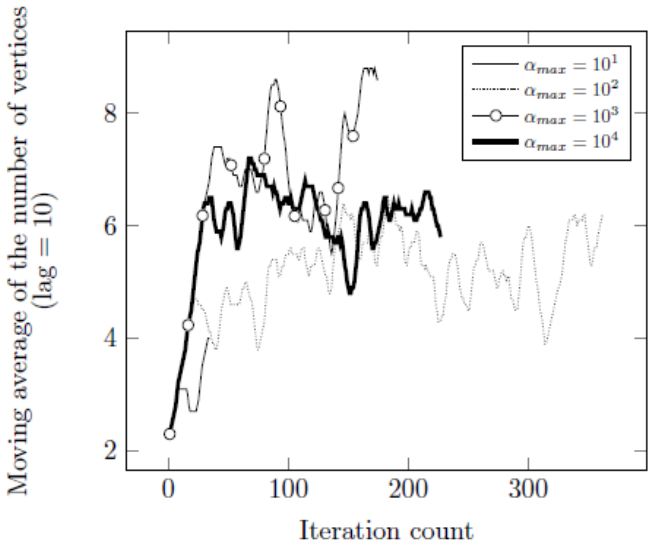


Figure 26 – Moving average of the number of vertices under consideration as the algorithm progresses for benchmark equilibrium problem 5.

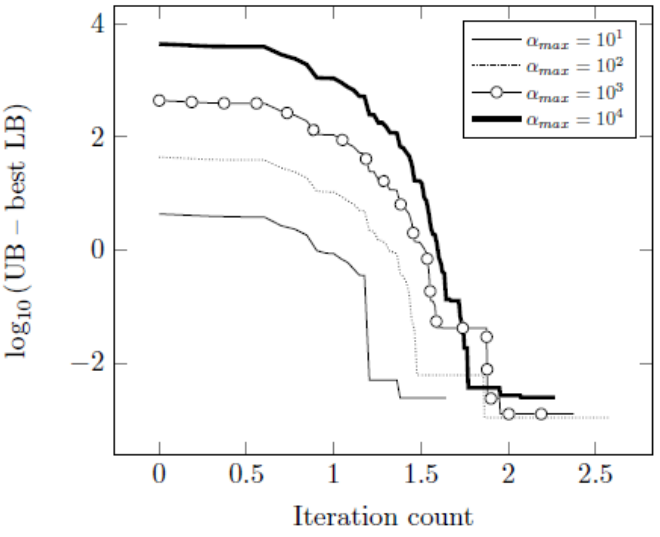


Figure 27 - Upper bound / best lower bound gap versus number of iterations for benchmark equilibrium problem 5.

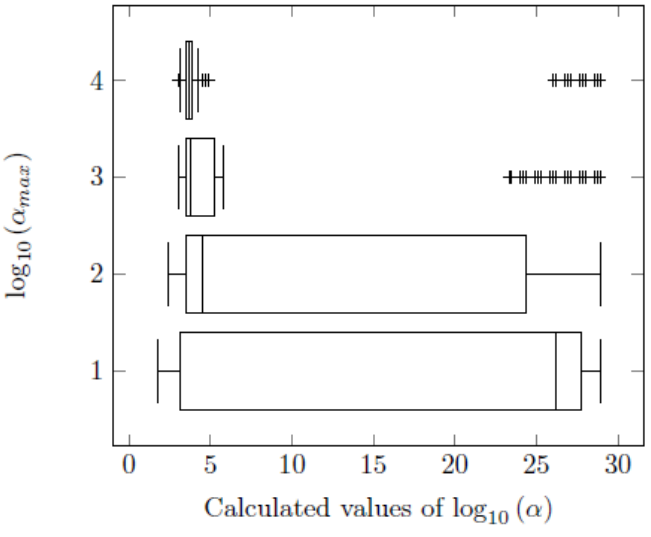


Figure 28 – Box plot of the logarithm of all the values of α calculated through interval analysis for benchmark equilibrium problem 5.

5.3.6 Problem 6

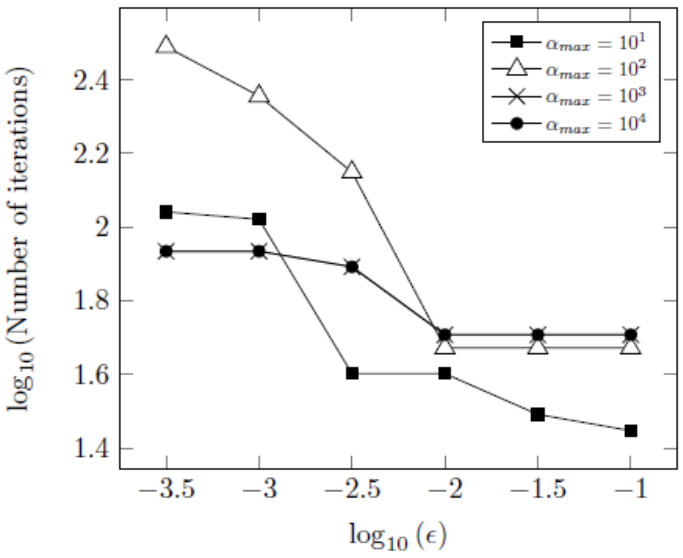


Figure 29 – Logarithm of the number of iterations versus logarithm of the error tolerance for benchmark equilibrium problem 6.

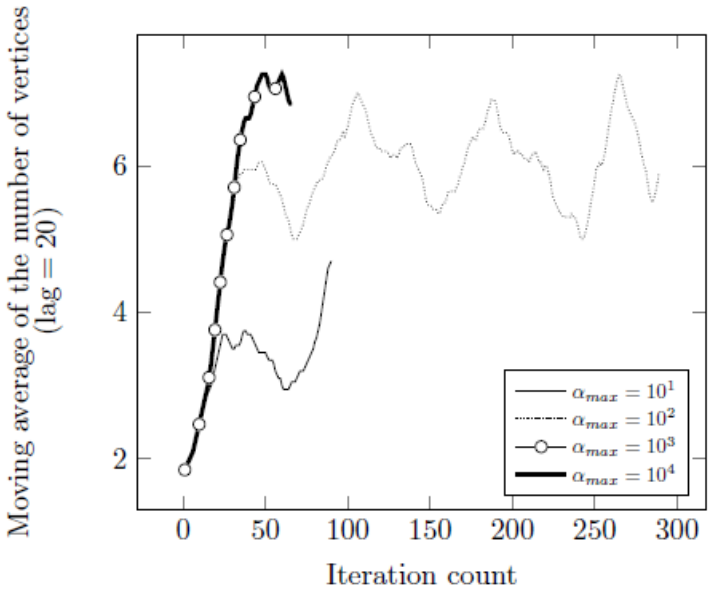


Figure 30 – Moving average of the number of vertices under consideration as the algorithm progresses for benchmark equilibrium problem 6.

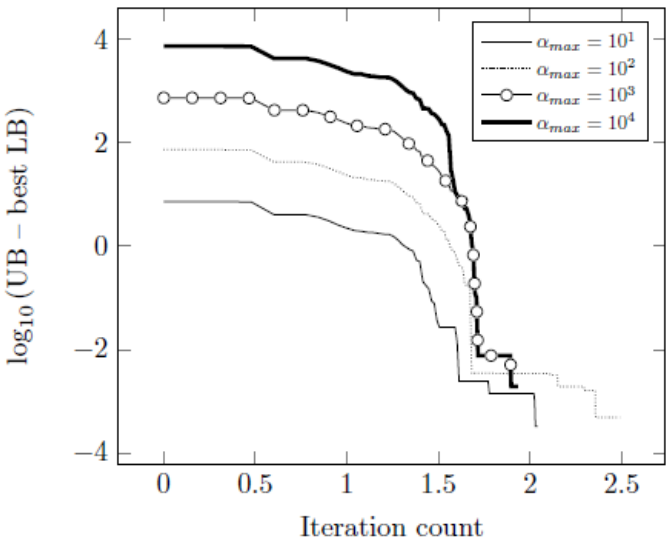


Figure 31 - Upper bound / best lower bound gap versus number of iterations for benchmark equilibrium problem 6.

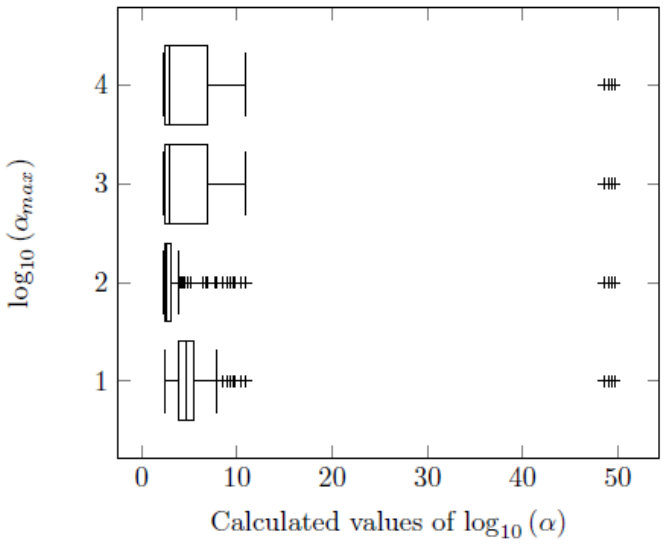


Figure 32 – Box plot of the logarithm of all the values of α calculated through interval analysis for benchmark equilibrium problem 6.

5.3.7 Problem 7

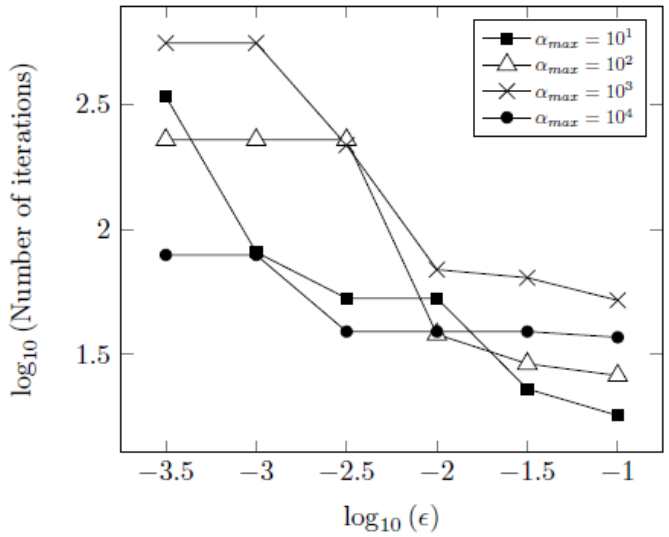


Figure 33 – Logarithm of the number of iterations versus logarithm of the error tolerance for benchmark equilibrium problem 7.

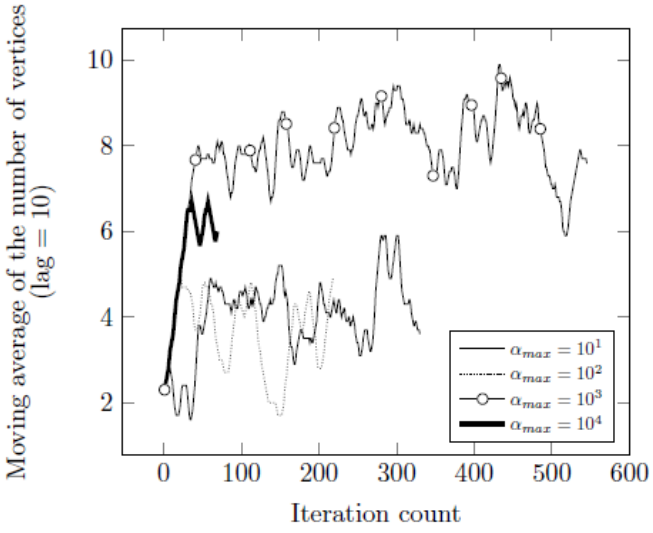


Figure 34 – Moving average of the number of vertices under consideration as the algorithm progresses for benchmark equilibrium problem 7.

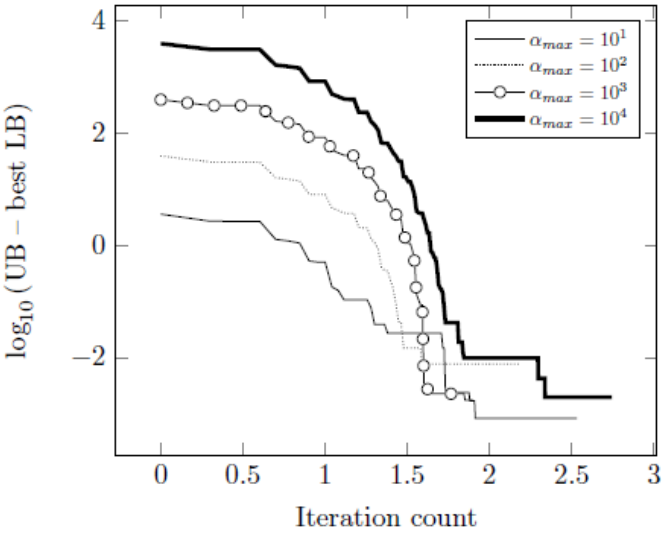


Figure 35 - Upper bound / best lower bound gap versus number of iterations for benchmark equilibrium problem 7.

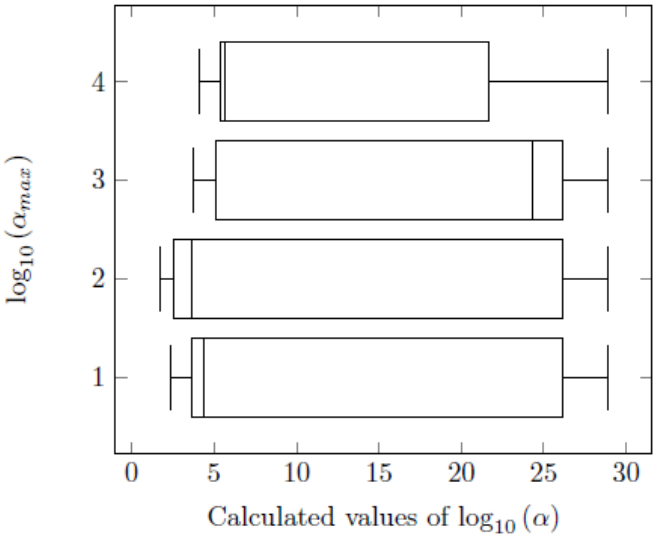


Figure 36 – Box plot of the logarithm of all the values of α calculated through interval analysis for benchmark equilibrium problem 8.

5.3.8 Problem 8

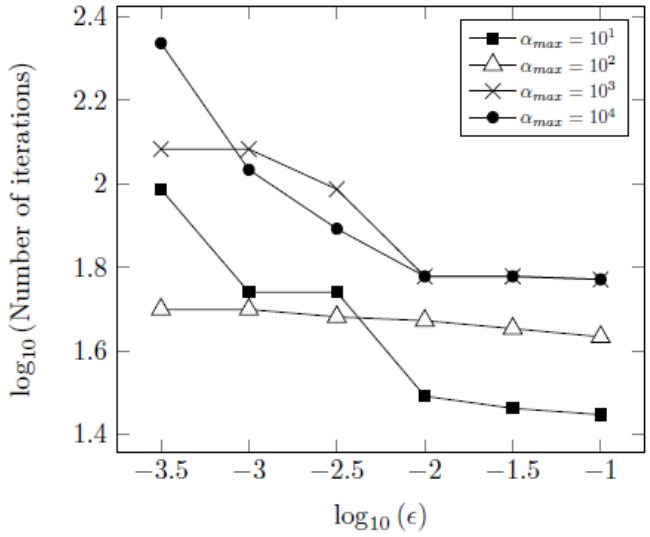


Figure 37 – Logarithm of the number of iterations versus logarithm of the error tolerance for benchmark equilibrium problem 8.

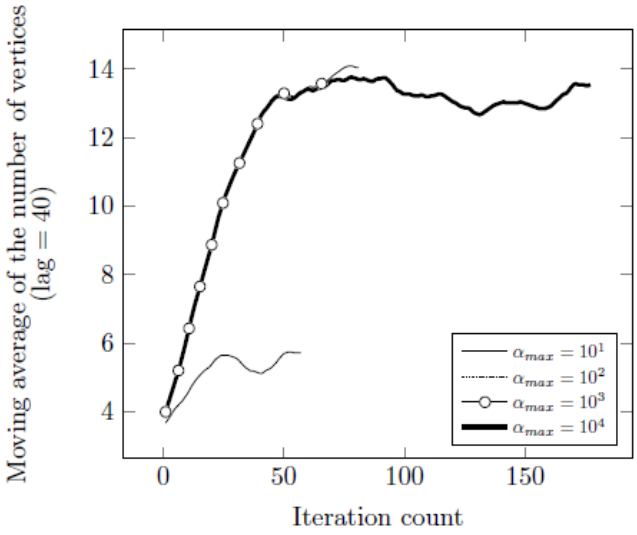


Figure 38 – Moving average of the number of vertices under consideration as the algorithm progresses for benchmark equilibrium problem 8.

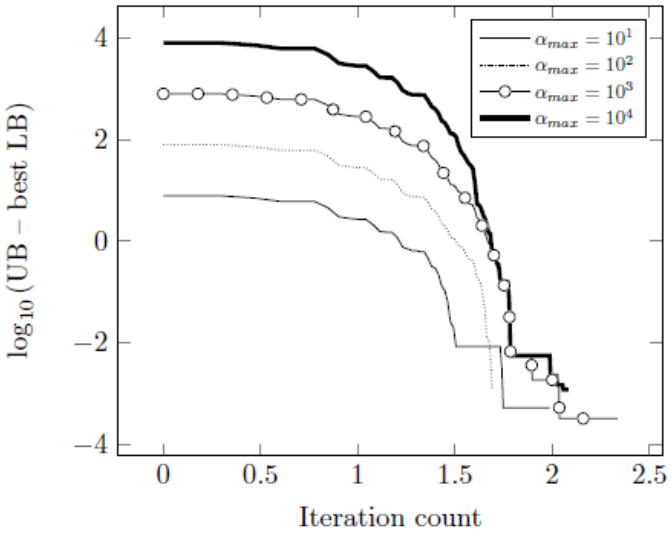


Figure 39 - Upper bound / best lower bound gap versus number of iterations for benchmark equilibrium problem 8.

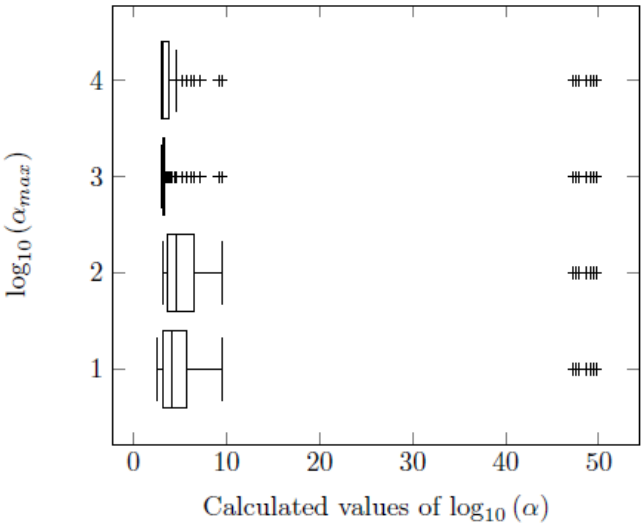


Figure 40 – Box plot of the logarithm of all the values of α calculated through interval analysis for benchmark equilibrium problem 8.

5.4 CALCULATED OPTIMUM VALUES

Table 10 displays the optimum value found for each problem when the algorithm was run with different values of α_{max} and ϵ. It also displays the true global optimum for each problem, as reported in the literature (Bonilla-Petricolet et al, 2011). Notice that the algorithm manage to successfully compute the global optimum even for low values of both α_{max} and relatively high error tolerances. Once again, this reinforces the remark that, in practice, the systematic scanning of the domain may be enough to surpass the nonconvexities in this kind of problem.

Table 11 shows the calculated distribution of species between phases in equilibrium for each problem, the total number of moles of all components in each phase and the degree of advancement of the reactions involved therein. The physical interpretation of these results is discussed next.

In problem 1, a vapor-liquid chemical equilibrium problem, phase 1 corresponds to the liquid phase whereas phase 2 corresponds to the vapor phase. It is clear that the mole fraction of ethyl acetate (component 3) is greater in the vapor phase than in the liquid phase. Conversely, the mole fraction of water (component 4) is greater in the liquid phase than in the vapor phase. That is consistent with the fact that ethyl acetate displays considerably weaker molecular interactions than those of water. For that reason, ethyl acetate tends to be dispersed in the vapor phase whereas water tends to accumulate in the condensed phase. The relatively low fractions of ethanol (component 1) and acetic acid (component 2) are due to the fact that these compounds react and are consumed to a considerable extent as is indicated by a relatively high equilibrium constant. The reaction reaches equilibrium at an advancement of 0.4236.

In problem 2, a vapor-liquid chemical equilibrium problem, phase 1 corresponds to the liquid phase whereas phase 2 corresponds to the vapor phase. The mole fraction of methyl tert-butyl ether (component 3), often abbreviated MTBE, is greater in the liquid phase than in the vapor phase. Upon evaluating and comparing the vapor pressures of MTBE and n-butane calculated through Antoine equation at T = 373.15 K, the values of 3.6 bar and 14.6 bar, respectively are obtained. The Antoine coefficients are given in Appendix C. It is possible to see that MTBE is far less volatile than n-butane. For that reason, n-butane tends to accumulate in the vapor phase, whereas MTBE accumulates in the liquid phase. The relatively low fractions of isobutene (component 1) and methanol (component 2) are due to the fact that these compounds react and are consumed to a considerable extent as is indicated by a relatively high equilibrium constant ($K_{eq} = 15.6$). The reaction reaches equilibrium at an advancement of 0.2041.

In problem 3, a vapor-liquid chemical equilibrium problem, phase 1 corresponds to the liquid phase whereas phase 2 corresponds to the vapor phase. The mole fraction of tert-amyl methyl ether (component 4) is greater in the liquid phase than in the vapor phase. The opposite happens to methanol (component 3). Once again, upon evaluating and comparing the vapor pressures of tert-amyl methyl ether and methanol calculated through Antoine equation at T = 373.15 K, the values of 0.45 bar and 0.91 bar, respectively are obtained. The Antoine coefficients are given in Appendix C. It is possible to see that tert-amyl methyl ether is far less volatile than methanol and, for that reason, methanol tends to accumulate in the vapor phase, whereas tert-amyl methyl ether accumulates in the liquid phase. The vapor pressures calculated for 2-methyl-1-butene (component 1) and for 2-methyl-2-butene (component 2) are significantly higher – 2.6 bar and 2.4 bar, respectively. The reaction reaches equilibrium at an advancement of 0.0707.

Problem 4, represents a liquid-liquid chemical equilibrium problem, phase 1 corresponds to the organic phase whereas phase 2 corresponds to the aqueous phase. In phase 1 a considerable amount of all compounds can be found, their mole fractions ranging from 16% to 36%. That is not unexpected, since all components, except water, are organic and not strongly polar, which leads them to aggregate in the same phase. Notice, however, that a considerable amount of water is also present in the organic phase. That is because water is still able, to some extent, to establish intermolecular interactions with the most polar parts of the other molecules, especially n-Butanol and n-Butyl acetate due to their particularly polar OH groups. Phase 2 contains mostly water and small amounts of n-Butyl acetate and n-Butanol, due to their polarity. The reaction reaches equilibrium at an advancement of 0.1486.

Problem 5 describes a liquid-liquid chemical equilibrium system. The components are not identified in the original article. The mole fraction of components 2 and 3 are greater in phase 1 than in phase 2, whereas the opposite happens to component 3. The reaction reaches equilibrium at an advancement of 0.2745.

In problem 6, a vapor-liquid chemical equilibrium problem, phase 1 corresponds to the liquid phase whereas phase 2 corresponds to the vapor phase. The mole fraction of methanol (component 3), is greater in the liquid phase than in the vapor phase. The opposite happens to n-pentane (component 5). Upon evaluating and comparing the vapor pressures of methanol and n-pentane calculated through Antoine equation at T = 335 K, the values of 0.91 bar and 2.28 bar, respectively are obtained. It is possible to see that methanol is far less volatile than n-pentane. For that reason, n-pentane tends to accumulate in the vapor phase, whereas methanol accumulates in the liquid phase. The reaction reaches equilibrium at an advancement of 0.0745.

Problem 7 represents a liquid-liquid chemical equilibrium system. The components are not identified in the original article. The mole fraction of components 1 and 2 are greater in phase 1 than in phase 2, whereas the opposite happens to component 3. The reaction reaches equilibrium at an advancement of 0.4405.

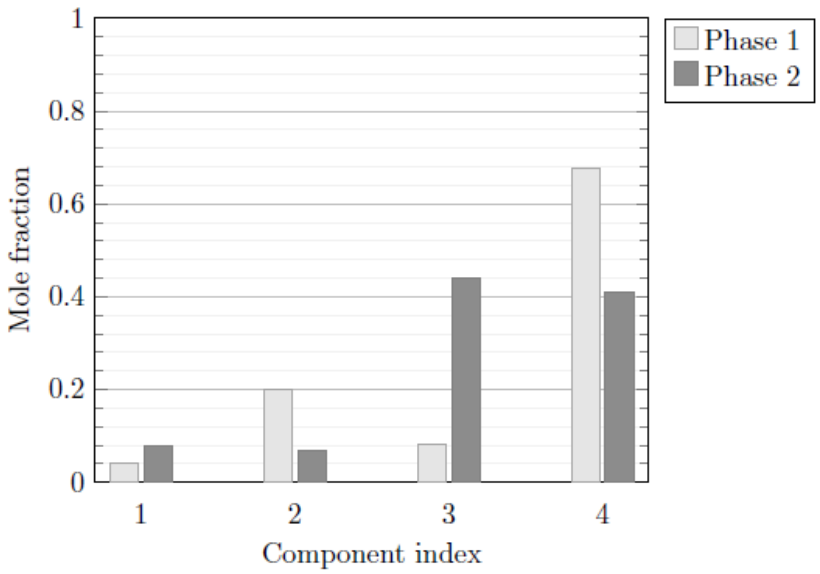
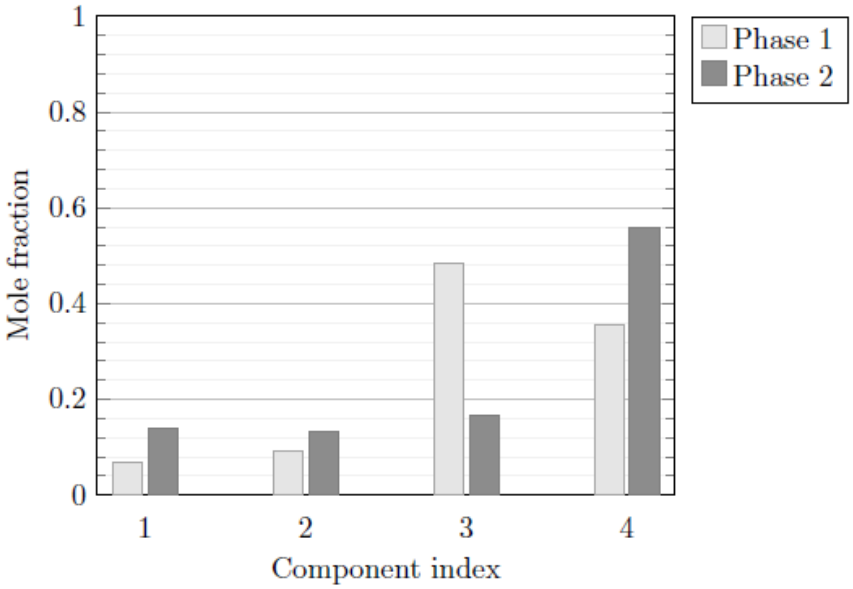
Problem 8 represents a liquid-liquid chemical equilibrium system. The components are not identified in the original article. It is quite clear that component 3 is the major constituent of phase 1, in which phase there is very little of any other component. In phase 2, the major constituent is component 2, even though smaller fractions of components 3 and 4 are also present. Component 1 is a minor compound in both phases because the total amount of it fed to the system was relatively low. The reaction reaches equilibrium at an advancement of 0.0405.

Table 10 – Optimum values found for each equilibrium problem when the algorithm was run with different combinations of α_{max} and $log_{10}\epsilon$.

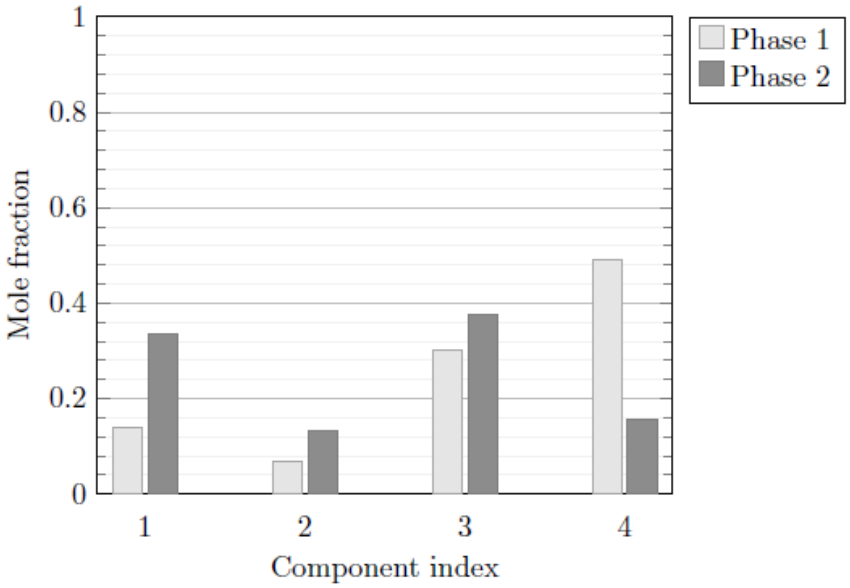
Problem	α_{max}	$\log_{10}\epsilon$						Global optimum(*)
		-4.5	**-4**	**-3.5**	**-3**	**-2.5**	**-2**	
1	10	-2.05812	-2.05812	-2.05812	-2.05812	-2.05812	-2.05812	-2.05813
	100	-2.05812	-2.05812	-2.05812	-2.05812	-2.05812	-2.05812	
	1000	-2.05812	-2.05812	-2.05812	-2.05812	-2.05812	-2.05812	
	10000	-2.05812	-2.05812	-2.05812	-2.05812	-2.05812	-2.05812	
2	10	-1.43493	-1.43493	-1.43493	-1.43493	-1.43493	-1.43493	-1.43427
	100	-1.43493	-1.43493	-1.43493	-1.43493	-1.43493	-1.43493	
	1000	-1.43493	-1.43493	-1.43493	-1.43493	-1.43493	-1.43493	
	10000	-1.43493	-1.43493	-1.43493	-1.43493	-1.43493	-1.43493	
3	10	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637
	100	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637	
	1000	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637	
	10000	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637	-1.22637	
4	10	-1.10628	-1.10628	-1.10628	-1.10628	-1.10628	-1.10628	-1.10630
	100	-1.10628	-1.10628	-1.10628	-1.10628	-1.10628	-1.10628	
	1000	-1.10628	-1.10628	-1.10628	-1.10628	-1.10628	-1.10628	
	10000	-1.10628	-1.10628	-1.10628	-1.10628	-1.10628	-1.10628	
5	10	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451
	100	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451	
	1000	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451	
	10000	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451	-0.14451	
6	10	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258
	100	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258	
	1000	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258	
	10000	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258	-0.87258	
7	10	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376
	100	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376	
	1000	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376	
	10000	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376	-0.65376	
8	10	-0.31198	-0.31198	-0.31198	-0.31198	-0.31198	-0.31198	-0.31192
	100	-0.31198	-0.31198	-0.31198	-0.31198	-0.31198	-0.31198	
	1000	-0.31198	-0.31198	-0.31198	-0.31198	-0.31198	-0.31198	
	10000	-0.31198	-0.31198	-0.31198	-0.31198	-0.31198	-0.31198	

(*) extracted from (Bonilla-Petricolet et al, 2011).

Table 11 – Calculated component distribution between the phases in equilibrium corresponding to the global optimum.

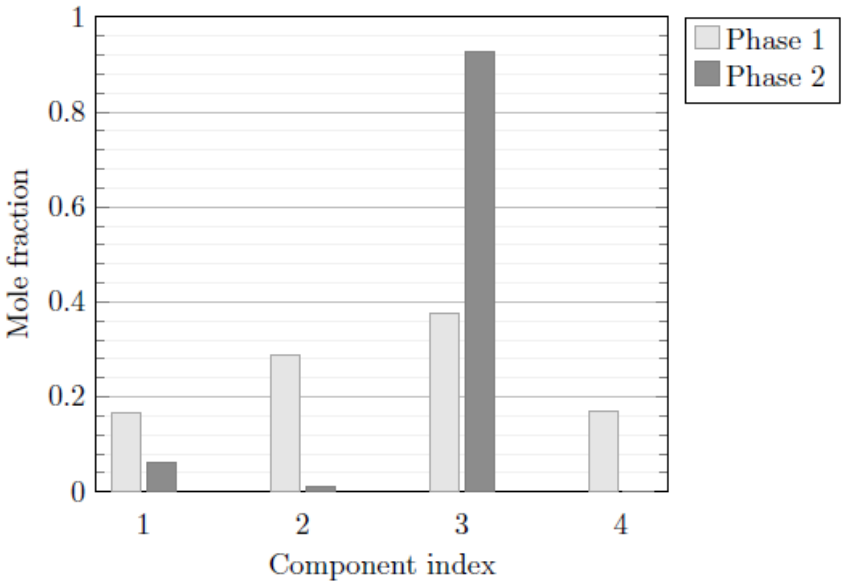
Problem	Component distribution between the phases	Total moles and advancement
1		$n_1^{total} = 0.0492$ $n_2^{total} = 0.9509$ $\xi = 0.4236$
2		$n_1^{total} = 0.2224$ $n_2^{total} = 0.5736$ $\xi = 0.2041$

3



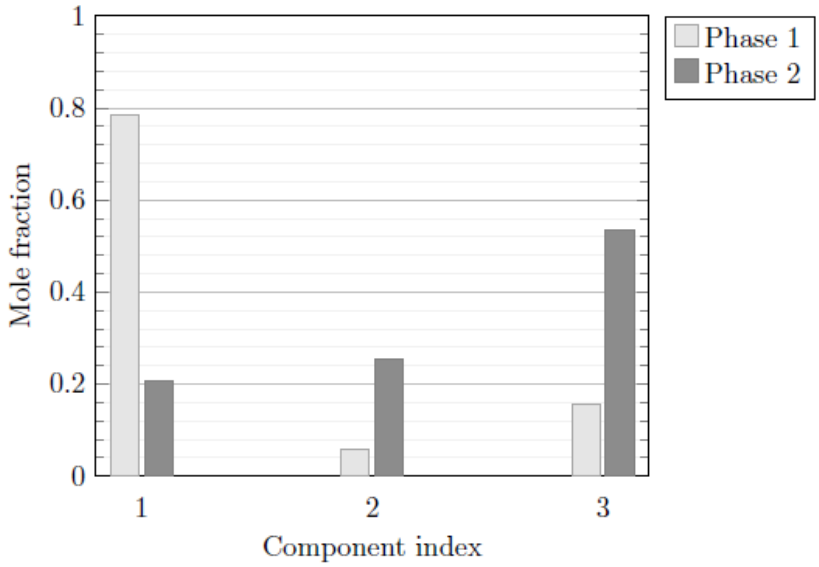
$n_1^{total} = 0.0252$
$n_2^{total} = 0.8333$
$\xi = 0.0707$

4



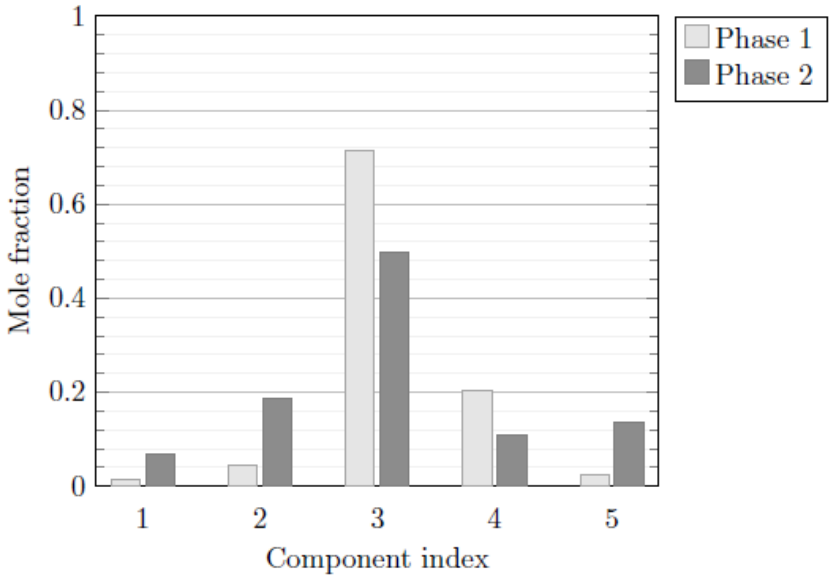
$n_1^{total} = 0.1302$
$n_2^{total} = 0.8698$
$\xi = 0.1486$

5



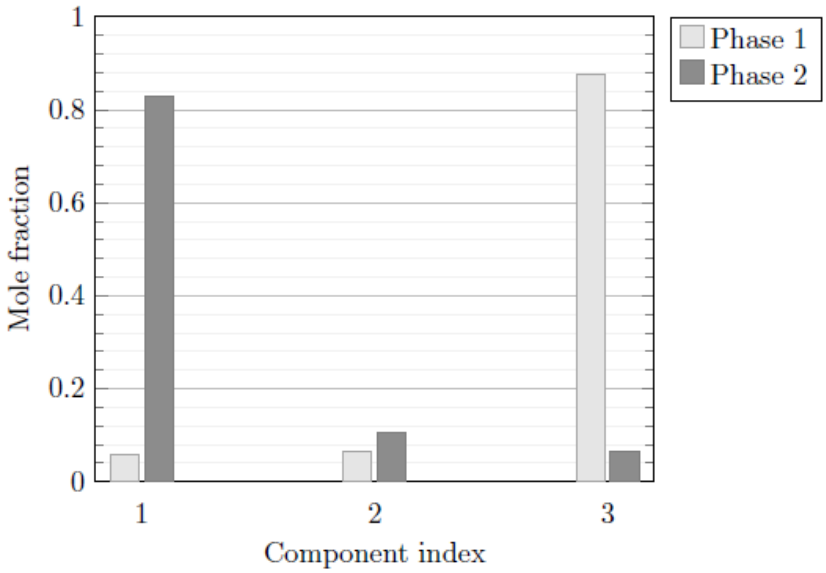
$n_1^{total} = 0.4236$
$n_2^{total} = 0.3018$
$\xi = 0.2745$

6



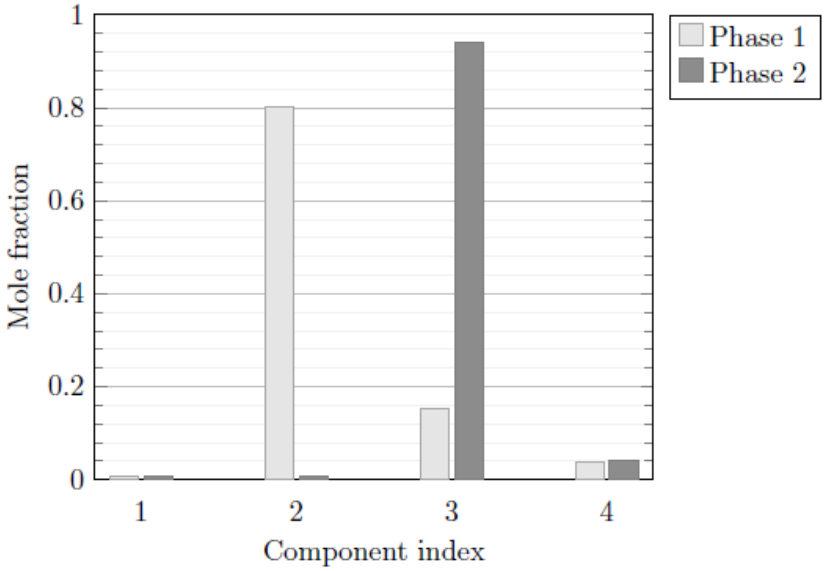
$n_1^{total} = 0.5902$
$n_2^{total} = 0.2609$
$\xi = 0.0745$

7



$n_1^{total} = 0.4979$
$n_2^{total} = 0.0616$
$\xi = 0.4405$

8



$n_1^{total} = 0.5691$
$n_2^{total} = 0.4308$
$\xi = 0.0405$

5.5 COMPARISON WITH THE RESULTS OBTAINED BY BONILLA-PETRICOLET AND COLLABORATORS

Bonilla-Petricolet and collaborators (2011) studied the performance of three different stochastic optimization algorithms, namely, simulated annealing (SA), genetic algorithms (GA) and differential evolution with tabu list (DETL) when solving the eight benchmark problems presented.

The authors have run all three algorithms in each benchmark equilibrium problem 100 times using random initial guesses. The benchmark problems were formulated both as constrained optimization problems (which is the formulation that employed in this work) and unconstrained optimization problems (which is the equivalent formulation described in the section titled *Reduction of dimensionality*). The success rate of the i-th problem, SR_i, was defined as the percentage of those trials that successfully reached the global optimum. For each such method, then, a global success rate GSR was defined as the average of the success rates of all problems using that particular method (Bonilla-Petricolet et al., 2011).

The authors have also considered two stopping criteria for the algorithms: a) maximum number of iterations divided by the maximum number of generations (SC1) and b) maximum number of iterations without improvement of the best function value (SC2).

The authors then moved on to examine how GSR varied as a function of both SC1 and SC2. The authors have also considered how well these algorithms would behave when the solution found was improved using a quasi-Newton local solver.

For the constrained optimization formulation, both with and without improving the solution through the quasi-Newton method, none of the algorithms displayed a GSR that exceeded 90% for any stopping criteria. It is also clear that the quasi-Newton improving step significantly improved GSR for both criteria (Bonilla-Petricolet et al., 2011), most noticeably for SC2.

It should be pointed out that the fact that these algorithms have failed to attain the global maximum at least 10% of the time is in contrast with the adapted algorithm presented in this work. This algorithm, being based on a global deterministic method, has reached the global optima of all problems in one run.

6 CONCLUSIONS AND SUGGESTIONS

From what has been exposed, it is possible to reach the following conclusions:

- If it is desired to employ the αBB algorithm with interval analysis for solving chemical equilibria problems through direct Gibbs energy minimization, it is important to realize that the α values so calculated may be exceedingly large, which, may cause local solvers to fail. This difficulty must be circumvented somehow.
- The strategy chosen here to keep α values from growing too large proved to be effective for solving the benchmark equilibrium problems under study, and the adapted algorithm was able to reach the global optimum.
- The adapted algorithm, due to the fact that it does not mathematically guarantee perfect underestimation, can no longer guarantee global optimality. However, in practice, it was capable of correctly attaining the global optimum even for relatively low values of α_{max} and relatively low values of ϵ.
- The upper bounds found by the algorithm converge to the global optimum in relatively few iterations, which suggests that the systematic scanning of the domain by the algorithm may be enough, in practice, to reliably find the global optimum.
- This also suggests that the αBB algorithm with interval analysis provides a powerful framework that may be used as a base for other algorithms.

In light of these conclusions, the following questions may be answered in future studies:

- How well does this algorithm behave when nonideal gaseous / vapor phases are considered?

- Does the algorithm behave equally well in highly nonideal systems, such as electrolyte solutions?

- Is it possible to extend the algorithm to effectively account for an arbitrary number of phases?

- Can the problem of having too large values of α be circumvented in any other way which, preferably, preserves the mathematical guarantee of optimality?

- How can the underestimating functions be made tighter?

- Can the αBB framework be used in conjunction with stochastic algorithms to improve its performance?

- Is it possible to somehow randomize the algorithm to achieve a better mean performance?

7 REFERENCES

Abbott, M. M., Smith, J. M. & Van Ness, H. C., 2001. *Chemical engineering thermodynamics.* Boston: McGraw-Hill.

Androulakis, I. & Maranas, C. F. C., 1995. αBB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization,* Issue 7, pp. 337-363.

Barber, C. B., Dobkin, D. P. & Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software,* Issue 22, pp. 469-483.

Bhargava, V., Fateen, S. & Bonilla-Petricolet, A., 2013. Cuckoo Search: A new nature-inspired optimization method for phase equilibrium calculations. *Fluid Phase Equilibria,* Issue 337, p. 191– 200.

Bonilla-Petricolet, A., Rangaiah, G. P. & Segovia-Hernández, J. G., 2011. Constrained and unconstrained Gibbs energy. *Fluid phase equilibria,* Volume 300, pp. 120-134.

Boyd, S. & Vandenberghe, L., 2004. *Convex optimization.* Cambridge: Cambridge University Press.

Brezonik, P. & Arnold, W., 2011. *Water chemistry: an introduction to the chemistry of natural and engineered aquatic systems.* New York: Oxford University Press.

Denbigh, K., 1971. *Principles of chemical equilibrium.* Cambridge: Cambridge University Press.

Floudas, C. A., 2000. *Deterministic global optimization: theory, methods and applications.* New York: Springer-Science+Business Media.

Floudas, C. A. & Paradalos, P. M., 2009. *Encyclopedia of optimization.* New York: Springer Science+Business.

Gopal, V. & Biegler, L. T., 1997. Nonsmooth dynamic simulation with linear programming based methods. *Computers and Chemical Engineering,* 21(7), pp. 675-689.

Greiner, H., 1988. Computing complex chemical equilibria by generalized linear programming. *Mathematical and Computer modelling,* 10(7), pp. 529-550.

Greiner, H., 1988. *The chemical equilibrium problem for a multiphase system formulated as a convex program.* Irsee Bavaria, Pergamon Press, pp. 155-170.

Karpov, I., Chudnenko, K. & Kulik, D., 1997. Modeling chemical mass transfer in geochemical processes: thermodynamic relations, conditions of equilibria, and numerical algorithms. *American Journal of Science,* Volume 297, pp. 767-806.

Kulik, D. e. a., 2012. GEM-Selektor geochemical modeling package: revised algorithm and GEMS3K numerical kernel for coupled simulation codes. *Computers & Geosciences,* Issue 17, pp. 1-24.

Lax, P., 2007. *Linear algebra and its applications.* 2nd ed. New York: Wiley-Intersciences.

Luenberger, D. G. & Ye, Y., 2010. *Linear and nonlinear programming.* 3rd ed. New York: Springer Science + Business Media.

Maranas, C. D. F. C. A., 1994. Global minimum potential energy conformations of small molecules. *Journal of Global Optimization,* 4(2), pp. 135 -170.

Mathworks, 2016. *Constrained Nonlinear Optimization Algorithms.* [Online]
Available at: http://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html

McDonald, C. & Floudas, C., 1995. Global optimization for the phase and chemical equilibrium problem: application to the NRTL equation. *Computers & Chemical Engineering,* 19(11), p. 1111–1139.

McDonald, C. & Floudas, C. A., 1995. Global optimisation and analysis for the Gibbs free energy function using the UNIFAC, Wilson and ASOG equations. *Industrial & Engineering Chemistry Research,* Issue 34, pp. 1674-1687.

Moore, R. E., Kearfott, R. B. & Cloud, M. J., 2009. *Introduction to interval analysis.* Philadelphia: Society for Industrial and Applied Mathematics.

Munkres, J. R., 2000. *Topology.* 2nd ed. Upper Saddle River: Prentice Hall Inc..

Nichita, V. D., Gomez, S. & Luna, E., 2002. Multiphase equilibria calculation by direct minimization of Gibbs free energy with a global optimization method. *Computers and Chemical Engineering,* Issue 26, pp. 1703-1724.

Perevoshchikova, N. et al., 2012. A convex hull algorithm for a grid minimzation of Gibbs energy as initial step in equilibrium calculations in two-phase multicomponent alloys. *Computational Materials Science,* Issue 61, pp. 54-66.

Reid, R. C., Prausnitz, J. M. & Poling, B. E., 1987. *The properties of gases and liquids.* 5th ed. New York: McGraw Hill Inc..

Rossi, C., Cardozo-Filho, L. & Guirardello, R., 2009. Gibbs free energy minimization for the calculation of chemical and phase equilibrium using linear programming. *Fluid Phase Equilibria,* pp. 117-128.

Rudin, W., 1976. *Principles of mathematical analysis.* New York: McGraw Hill Education.

Ryll, O., Blago, S. & Hasse, H., 2012. Convex envelope mehod for the determination of fluid phase diagrams. *Fluid Phase Equilibria,* Issue 324, pp. 108-116.

Sommerville, I., 2011. *Software Engineering.* Massachusetts: Addison-Wesley.

Spall, J. C., 2003. *Introduction to stochastic search and optimization: estimation, simulation and control.* New Jersey: Wiley-Interscience.

Tchobanoglous, G. & Schroeder, E. D., 1987. *Water quality: characteristics, modeling and modification.* Boston: Addison-Wesley.

Tester, J. & Modell, M., 1997. *Thermodynamics and its applications.* 3rd ed. Upper Saddle River: Prentice Hall PTR.

Ung, S. & Doherty, M. F., 1995. Vapor-liquid phase equilibrium in systems with multiple chemical reactions. *Chemical Engineering Science,* Volume 50, pp. 23-48.

Vanderbei, R. J., 2007. *Linear programming: foundations and extensions.* 3rd ed. New York: Springer Science + Business Media.

Walas, S., 1985. *Phase Equilibria in Chemical Engineering.* Stoneham: Massachusetts.

Wasylkiewicz, S. K. & Ung, S., 2000. Global phase stability analysis for heterogeneous reactive mixtures and calculation of reactive liquid–liquid and vapor–liquid–liquid equilibria. *Fluid Phase Equilibria,* p. 253–272.

Zhang, H., Bonilla-Petricolet, A. & Rangaiah, G. P., 2011. A review on global optimization methods for phase equilibrium modeling and calculations. *The open thermodynamics journal,* pp. 71-92.

8 APPENDIX A

8.1 ELLIPSOID METHOD IMPLEMENTED BY KARPOV AND COLLABORATORS (1997)

In this section, we describe the implementation of the IPM algorithm by Karpov and collaborators (1997). The actual IPM algorithm used is an ellipsoid method. It takes as input an initial feasible approximation $\boldsymbol{x}^0$ and a tolerance ϵ. The algorithm finds a feasible solution Δ^r at the *r*-th iteration by solving the following program:

$$\begin{aligned} &\min \quad \sum_{j\in L_s} v_j^r \, \Delta_j \\ &s.t. \\ &\qquad \boldsymbol{A}\Delta = \mathbf{0}, \\ &\qquad \sum_{j\in L_s} \frac{\Delta_j^2}{q_j^r} \le 1 \end{aligned} \tag{91}$$

Where

$$q_j^r = \begin{cases} x_j^r - \underline{x_j} & \text{if only a lower bound } \underline{x_j} \text{ is provided} \\ \overline{x_j} - x_j^r & \text{if only an upper bound } \overline{x_j} \text{ is provided} \\ \min\left(x_j^r - \underline{x_j}\,;\ \overline{x_j} - x_j^r\right) & \text{if both are provided} \end{cases} \tag{92}$$

L_s denotes the set of dependent components. With the aid of Lagrange multipliers – u_i^r –, the optimization problem (87) can be converted to solving a system of linear equations:

$$\sum_{i=1}^{n} r_{ik}^r u_i^r = \sum_{j\in L_s} v_i^r a_{kj} q_j^r \qquad i,k \in N \tag{93}$$

$$r_{ik}^{r} = \sum_{j \in L_s} a_{kj} a_{ij} q_j^r \qquad i,k \in N$$

N represents the set of indices of the independent components of the system and $n = |N|$. From that, one obtains u_i^r. The calculation of Δ^r is done s follows (Karpov, et al., 1997):

$$\Delta_j^r = q_j^r \left(\sum_{i=1}^{n} a_{ij} u_i^r - v_j^r \right) \qquad i \in L_s \tag{94}$$

In possession of the descent direction Δ^r, the variable values for the next iteration can be computed as usual:

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^{r+1} + \lambda_r \Delta^r \tag{95}$$

The step size is represented by λ_r and should be small. The stopping criterion used was Dikin's criterion (Karpov, et al., 1997):

$$CD = \sqrt{\sum_{j \in L_s} \left[q_j^r\right]^2 \left(\sum_{i=1}^{n} a_{ij} u_i^r - v_j^r \right)^2} \leq \epsilon \tag{96}$$

Where $\epsilon = 10^{-5}$. Through this approach, the authors have successfully performed Helmholtz energy minimization on a 4-phase, 39-component system (Karpov, et al., 1997).

8.2 MASS BALANCE REFINEMENT (MBR) ROUTINE PROPOSED BY KULIK AND COLLABORATORS (2012)

This procedure aims to minimize, within a pre-specified tolerance, the mass balance residuals $\varsigma_i^{(r)}$, where:

$$\varsigma_i^{(r)} = n_i^{(b)} - \sum_{j \in L} a_{ij} n_j^{(y,r)} \qquad i \in N \tag{97}$$

The index *r* denotes the iteration number, and is present in the above formula due to the fact that this procedure can be used multiple times, in each iteration, in order to assure that mass balances will not be violated within some tolerance. The authors mention that this tolerance generally ranges from 10^{-6} mol/kg to 10^{-8} mol/kg, except for trace components, in which case it might be required that it be smaller than 10^{-9} mol/kg.

The step size of the main IPM is calculated as follows:

$$\lambda_r = \underset{0 \le \lambda \le \mu_r}{\operatorname{argmin}} \left[\sum_{j \in L} \left(n_j^{(x,r)} + \lambda_j \right) v_j^{(r)} \right] \tag{98}$$

Which is a one-dimensional optimization problem. The vector of number of moles being processed by the IPM at the *r*-th iteration is designated by $\boldsymbol{n}^{(x,r)}$. The upper bound μ_r must be such that $\boldsymbol{n}^{(x,r)} + \mu_r \Delta^r$ remains within the feasible region.

In highly non-ideal systems, the Gibbs energy function may display severe oscillations, which, if not treated appropriately, may compromise the convergence of the proposed IPM algorithm. For that reason, a so-called smoothing factor α_γ is introduced. The authors found that the amount of smoothing required would depend on the ration of the Dikin's criterion and the prescribed convergence tolerance ϵ_D:

$$\alpha_\gamma = \epsilon_\delta + \exp[\ln(1 - \epsilon_\alpha) + \ln CD] + \frac{\epsilon_\alpha - \epsilon_\delta}{1 + \frac{\exp(\ln \epsilon_D - \ln CD)}{\epsilon_\delta}} \tag{99}$$

Aside from ϵ_D, two other empirical constants are needed, namely ϵ_α and ϵ_δ, both of them lying within the interval $(0,1)$. ϵ_α sets the position of the α_γ plateau during the first iterations, when

CD is high and severe corrections are necessary. ϵ_δ sets the minimum value for α_γ, which occurs when CD is small.

9 APPENDIX B

The GOP algorithm is applicable to the NRTL model due to its following property (McDonald & Floudas, 1995):

$$\sum_{i\in C} n_i^k \left\{\frac{\sum_{j\in C}\tau_{j,i}G_{j,i}n_j^k}{\sum_{j\in C}G_{j,i}n_j^k}\right\} - \sum_{i\in C} n_i^k \left\{\sum_{j\in C}\frac{G_{i,j}n_j^k}{\sum_{l\in C}G_{i,l}n_l^k}\left(\frac{\sum_{l\in C}\tau_{l,j}G_{l,j}n_l^k}{\sum_{l\in C}G_{l,j}n_l^k}\right)\right\} = 0 \qquad (100)$$

This property is extremely important, for it allows us to rewrite the Gibbs free energy in a much more convenient way (McDonald & Floudas, 1995):

$$G(\boldsymbol{n}) = \sum_{k\in\Pi} C^k + \sum_{k\in\Pi_L}\sum_{i\in C} n_i^k \left\{\sum_{j\in C}\frac{G_{i,j}\tau_{l,j}n_j^k}{\sum_{l\in C}G_{i,l}n_l^k}\right\} \qquad (101)$$

Where Π_L are the liquid phases to which NRTL model applies and C^k denotes:

$$C^k = \sum_{i\in C} n_i^k \left\{\frac{\Delta G_i^{k,f}}{RT} + \ln\frac{n_i^k}{\sum_{j\in C} n_j^k}\right\} \qquad (102)$$

It has already been established that the terms C^k are convex. Therefore, all non-convexities on the Gibbs function are due to the terms on right side of $\sum_{k\in\Pi} C^k$ in equation (101).

We can now define new variables (101) in such a way that the original problem is transformed into an equivalent biconvex problem. The new variables are:

$$\Psi_i^k = \frac{n_i^k}{\sum_{j\in C} G_{j,i} n_j^k} \quad \Leftrightarrow \quad \Psi_i^k \left\{ \sum_{j\in C} G_{j,i} n_j^k \right\} - n_i^k = 0 \qquad i \in C, \;\; k \in \Pi_L \tag{103}$$

These definitions are then incorporated in the transformed problem as equality constraints. Upon examination of the new constraints it possible to check that for every fixed n_i^k, the constraint is affine in $\Psi_i^k$and vice versa. Upon partitioning the variables as:

$$\boldsymbol{\Psi} \leftarrow \{\Psi_i^k\} \qquad \boldsymbol{n} \leftarrow \{n_i^k\} \tag{104}$$

The transformed optimization problem can now be stated as our primal problem:

$$\min_{\boldsymbol{n},\boldsymbol{\Psi}} G(\boldsymbol{\Psi}, \boldsymbol{n}) = \sum_{k\in\Pi} C^k + \sum_{k\in\Pi_L} \sum_{i\in C} n_i^k \left\{ \sum_{j\in C} G_{i,j} \tau_{l,j} \Psi_j^k \right\}$$

$$s.t.$$

$$\Psi_i^k \left\{ \sum_{j\in C} G_{j,i} n_j^k \right\} - n_i^k = 0 \qquad i \in C, \;\; k \in \Pi_L, \tag{105}$$

The primal problem is always feasible provided that the mole balance constraints are not violated (McDonald & Floudas, 1995). The mass balances were not included as constraints in the primal problem in order not to compromise its feasibility – in the actual algorithm, they will be included in the relaxed dual subproblems, which will be stated and explained later.

As mentioned, it will be need to solve relaxed dual subproblems and it will be also necessary to evaluated KKT conditions. These necessities lead us to the calculation of the Lagrangian functions associated to the primal problem (McDonald & Floudas, 1995). For a fixed number of moles $\bar{\boldsymbol{n}}$:

$$L(\boldsymbol{\Psi}, \bar{\boldsymbol{n}}, \boldsymbol{\lambda}) = \sum_{k \in \Pi} C^k + \sum_{k \in \Pi_L} \sum_{i \in C} \bar{n}_i^k \left\{ \sum_{j \in C} G_{i,j} \tau_{i,j} \Psi_j^k \right\} + \sum_{k \in \Pi_L} \sum_{i \in C} \lambda_{\Psi_i^k} \left\{ \Psi_i^k \sum_{j \in C} G_{j,i} \bar{n}_j^k - \bar{n}_i^k \right\} \quad (106)$$

Evaluating the KKT conditions for the primal, i.e., solving for $\nabla_{\Psi_i^k} L(\boldsymbol{\Psi}, \bar{\boldsymbol{n}}, \boldsymbol{\lambda}) = 0$ we obtain an explicit and compact formula for the Lagrange multipliers (McDonald & Floudas, 1995):

$$\lambda_{\Psi_i^k} = -\frac{\sum_{j \in C} G_{j,i} \tau_{j,i} \bar{n}_j^k}{\sum_{j \in C} G_{j,i} \bar{n}_j^k} \quad (107)$$

The authors point out that if a phase disappears, the denominator of the last expression will be 0. In this case, by making $\lambda_{\Psi_i^k} = 0 \ \ \forall i$ will make sure that the KKT conditions will be satisfied despite of that. Having established the preliminary properties of the NRTL-based Gibbs energy minimization problem, we now move to the actual GOP algorithm.

From a high-level perspective, the algorithm makes use of the primal problem, which upon evaluation or optimization always yields upper bounds for the global optimum, and of (relaxed) dual subproblems, which provide lower bounds. As Slater's condition is satisfied, it follows that the duality gap is zero. This suggests an iterative procedure, whereby primal and dual subproblems are solved and the distance between the upper and lower bounds obtained through them be used to gauge how precise the solution turns. A very detailed explanation of the algorithm, including proofs and mathematical background is provided by Floudas (2000).

We begin by noticing that problem (74) can be also written as:

$$
\begin{aligned}
&\min_{\boldsymbol{y}} v(\boldsymbol{y}) \\
&s.t. \\
&\quad v(\boldsymbol{y}) = \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}), \\
&\quad \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0}, \\
&\quad \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{0}, \\
&\quad \boldsymbol{y} \in Y \cap V \\
&V = \{\boldsymbol{y} \mid h(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0}, g(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{0} \text{ for some } \boldsymbol{x} \in X\}
\end{aligned} \tag{108}
$$

The problem has thus been rewritten as the combination of an internal and an external optimization problem. If we fix a value of $\boldsymbol{y} \in Y \cap V$, namely $\boldsymbol{y}^k$, from the strong duality theorem (which follows from Slater's condition), it follows that solving the internal optimization problem is equivalent to maximizing its dual. Mathematically (McDonald & Floudas, 1995):

$$
\left\{\begin{matrix} \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}^k) \\ s.t. \\ \quad \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) = \boldsymbol{0} \\ \quad \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k) \leq \boldsymbol{0} \end{matrix}\right\} = \sup_{\substack{\boldsymbol{\mu} \geq \boldsymbol{0} \\ \boldsymbol{\lambda}}} \inf_{\boldsymbol{x}} \{f(\boldsymbol{x}, \boldsymbol{y}^k) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) + \boldsymbol{\mu}^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k)\} \tag{109}
$$

In other words:

$$
v(\boldsymbol{y}) = \sup_{\substack{\boldsymbol{\mu} \geq \boldsymbol{0} \\ \boldsymbol{\lambda}}} inf_{\boldsymbol{x}} \{f(\boldsymbol{x}, \boldsymbol{y}^k) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) + \boldsymbol{\mu}^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k)\} \quad \forall \boldsymbol{y} \in Y \cap V \tag{110}
$$

This optimization problem may be just as hard to solve as the original problem. However, we may relax it in order to make it simpler. For that, we drop the set constraint $\boldsymbol{y} \in Y \cap V$ and instead of demanding equality in Equation (110), we relax it to an inequality:

$$
v(\boldsymbol{y}) \geq inf_{\boldsymbol{x}} \{f(\boldsymbol{x}, \boldsymbol{y}^k) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}^k) + \boldsymbol{\mu}^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}^k)\} \tag{111}
$$

Merging these results, we can finally write the so-called relaxed dual subproblem, which will provide lower bounds for the algorithm:

$$\begin{aligned} &\min_{\substack{\boldsymbol{y}\in\boldsymbol{Y}\\ \mu_B}} \mu_B \\ &s.t. \\ &\quad \mu_B \geq \min_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\mu}) \qquad \forall \boldsymbol{\mu} > \mathbf{0}, \boldsymbol{\lambda} \end{aligned} \tag{112}$$

Where $L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\mu}) = f(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) + \boldsymbol{\mu}^T\, \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y})$. We will refer to the optimization problem $\min_{\boldsymbol{x}} L\left(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda}^k,\boldsymbol{\mu}^k\right)$ as the inner relaxed dual subproblem (IRD). $\boldsymbol{\lambda}^k$ and $\boldsymbol{\mu}^k$ are the Lagrange multipliers obtained from the *k*-th primal problem. As this problem can still be hard to solve, we will replace it with a set of simpler problems that validly underestimate it. We now turn to the construction of such simpler problems. In the case of the NRTL model, by fixing the Lagrange multipliers $\bar{\boldsymbol{\lambda}}$, and recollecting terms, the Lagrangian can be written as:

$$\begin{aligned} L(\boldsymbol{\Psi},\boldsymbol{n},\bar{\boldsymbol{\lambda}}) = &\sum_{k\in\Pi} C^k + \sum_{k\in\Pi_L}\sum_{i\in C} \Psi_i^k \left\{ \sum_{j\in C} G_{i,j}\tau_{i,j} n_j^k + \bar{\lambda}_{\Psi_i^k} \sum_{j\in C} G_{j,i} n_j^k \right\} \\ &- \sum_{k\in\Pi_L}\sum_{i\in C} n_i^k \bar{\lambda}_{\Psi_i^k} \end{aligned} \tag{113}$$

The values of $\bar{\lambda}_{\Psi_i^k}$ are calculated as shown in Equation (107). By subtracting this last expression from equation (106) we obtain:

$$\begin{aligned} L(\boldsymbol{\Psi},\boldsymbol{n},\bar{\boldsymbol{\lambda}}) = &\sum_{k\in\Pi_L}\sum_{i\in C} \Psi_i^k \left\{ \sum_{j\in C} G_{j,i} \left[\tau_{j,i} + \bar{\lambda}_{\Psi_i^k}\right]\left[n_j^k - \bar{n}_j^k\right] \right\} + \sum_{k\in\Pi} C^k \\ &- \sum_{k\in\Pi_L}\sum_{i\in C} n_i^k \bar{\lambda}_{\Psi_i^k} \end{aligned} \tag{114}$$

It is now possible to obtain a compact expression for the gradients of the Lagrangian, the so-called qualifying constraints, which describe the nature of the interaction of the two variable sets (McDonald & Floudas, 1995).

$$g_i^k(\boldsymbol{y}) = \nabla_{\Psi_i^k} L\left(\boldsymbol{\Psi}, \boldsymbol{n}, \bar{\boldsymbol{\lambda}}\right) = \sum_{j \in C} G_{j,i} \left[\tau_{j,i} + \bar{\lambda}_{\Psi_i^k}\right] \left[n_j^k - \bar{n}_j^k\right] \qquad \forall i \in C, k \in \Pi_L \tag{115}$$

It can be seen that these constraints form hyperplanes that partition the $\boldsymbol{y}$ variable space. Each $\boldsymbol{x}$ variable, Ψ_i^k interacts with a summation of $\boldsymbol{y}$ variables, n_j^k. It turns out that we can improve this partitioning by making sure that each $\boldsymbol{y}$ variable interacts with only a single $\boldsymbol{x}$ variable. The set of partitioning hyperplanes will be, therefore, orthogonal. In order to achieve that, we will augment the set of $\boldsymbol{x}$ variables (McDonald & Floudas, 1995):

$$\hat{\Psi}_{ij}^k = \Psi_i^k \qquad \forall j \in C \tag{116}$$

We have, in a way introduced redundancy into the algorithm, but the orthogonality that we will be able to achieve justifies it. Reevaluating the gradient of the Lagrangian with respect to this new set of variables, it can be shown that:

$$\hat{g}_{ij}^k(\boldsymbol{y}) = \nabla_{\hat{\Psi}_{ij}^k} L\left(\boldsymbol{\Psi}, \boldsymbol{n}, \bar{\boldsymbol{\lambda}}\right) = G_{j,i} \left[\tau_{j,i} + \bar{\lambda}_{\hat{\Psi}_{ij}^k}\right] \left[n_j^k - \bar{n}_j^k\right] \qquad \forall i \in C, j \in C, k \in \Pi_L \tag{117}$$

Each of these hyperplanes divides the space in two regions – one where $n_j^k - \bar{n}_j^k \geq 0$ and other where $n_j^k - \bar{n}_j^k < 0$. From that it can be shown (McDonald & Floudas, 1995) that they partition the space in $2^{N_{CV}}$ *n*-rectangles, where $N_{CV} = |C||\Pi_L|$. In each one of those rectangles the sign of $n_j^k - \bar{n}_j^k$ will remain constant for all phases and components. We shall denote the box bounds

of each rectangle by $B\{L^B, U^B\}$, where L^B, U^B are the lower and upper bounds for each variable. The authors have also introduced the following notation (McDonald & Floudas, 1995):

$$\left.\begin{array}{lll} s_{ik}^{B_l} = +1 & \Longrightarrow & n_j^k - \bar{n}_j^k \geq 0 \\ s_{ik}^{B_l} = -1 & \Longrightarrow & n_j^k - \bar{n}_j^k < 0 \end{array}\right\} \quad \forall i \in C, k \in \Pi_L \tag{118}$$

Where each rectangle is designated by B_l. From the definition of $\widehat{\Psi}_{ij}^k$ and the lower and upper bounds of n_i^k, it is possible to find corresponding bounds for $\widehat{\Psi}_{ij}^k$:

$$L_{\widehat{\Psi}_{ij}^k} = \frac{L_{n_i^k}^B}{L_{n_i^k}^B + \sum_{j \neq i} G_{ji} U_{n_j^k}^B} \qquad U_{\widehat{\Psi}_{ij}^k} = \frac{U_{n_i^k}^B}{U_{n_i^k}^B + \sum_{j \neq i} G_{ji} L_{n_j^k}^B} \tag{119}$$

The last critical step in this derivation consists on deciding whether the $\boldsymbol{x}$ variables should take their lower or upper bounds at each iteration. If we collect terms it is possible to rewrite the Lagrangian as (McDonald & Floudas, 1995):

$$L(\widehat{\boldsymbol{\Psi}}, \boldsymbol{n}, \bar{\boldsymbol{\lambda}}) = \sum_{k \in \Pi_L} \sum_{i \in C} \left\{ \left[n_j^k - \bar{n}_j^k \right] \sum_{j \in C} \widehat{\Psi}_{ji}^k \left[G_{ij} \left\{ \tau_{ij} + \bar{\lambda}_{\psi_i^k} \right\} \right] \right\} + \sum_{k \in \Pi} C^k - \sum_{k \in \Pi_L} \sum_{i \in C} n_i^k \bar{\lambda}_{\psi_i^k} \tag{120}$$

In order to make a choice, it is necessary to check the sign of $n_j^k - \bar{n}_j^k$ and also check the sign of the multiplying term $G_{ij} \left\{ \tau_{ij} + \bar{\lambda}_{\psi_i^k} \right\}$. Much like before, we define the following variables to guide us in this decision process:

$$\bar{s}_{ji}^{k} = +1 \quad \text{if} \quad G_{ij}\left\{\tau_{ij} + \bar{\lambda}_{\Psi_i^k}\right\} \geq 0$$
$$\bar{s}_{ji}^{k} = -1 \quad \text{if} \quad G_{ij}\left\{\tau_{ij} + \bar{\lambda}_{\Psi_i^k}\right\} < 0 \qquad (121)$$

The decision is then made according to the following criterion. For the current iteration K:

$$\left(\bar{s}_{ji}^{k}\right)^{K} s_{ik}^{B_l} = +1 \quad \Longrightarrow \quad \left(\widehat{\Psi}_{ji}^{k}\right)^{B_l^K} = L_{\widehat{\Psi}_{ij}^{k}}$$
$$\left(\bar{s}_{ji}^{k}\right)^{K} s_{ik}^{B_l} = -1 \quad \Longrightarrow \quad \left(\widehat{\Psi}_{ji}^{k}\right)^{B_l^K} = U_{\widehat{\Psi}_{ij}^{k}} \qquad (122)$$

For all the previous iterations K_P:

$$\left(\bar{s}_{ji}^{k}\right)^{K_P}\left[\left(\bar{n}_{i}^{k}\right)^{K} - \left(\bar{n}_{i}^{k}\right)^{K_P}\right] = +1 \quad \Longrightarrow \quad \left(\widehat{\Psi}_{ji}^{k}\right)^{B_l^{K_P}} = L_{\widehat{\Psi}_{ij}^{k}}$$
$$\left(\bar{s}_{ji}^{k}\right)^{K_P}\left[\left(\bar{n}_{i}^{k}\right)^{K} - \left(\bar{n}_{i}^{k}\right)^{K_P}\right] = -1 \quad \Longrightarrow \quad \left(\widehat{\Psi}_{ji}^{k}\right)^{B_l^{K_P}} = U_{\widehat{\Psi}_{ij}^{k}} \qquad (123)$$

We can finally outline the algorithm. The details of implementation can be found in the original article (McDonald & Floudas, 1995).

First an initial guess is supplied and the upper and lower bounds for the mole numbers are set. The primal problem is then solved by any local optimization solver and its Lagrange multipliers, $\bar{\boldsymbol{\lambda}}^{K}$, are stored for that iteration. The values of $\left(\bar{s}_{ji}^{k}\right)^{K}$ are also calculated and stored. A combination of qualifying constraints $B_l$is chosen, its lower and upper bounds are calculated, as well as $s_{ik}^{B_l}$. We now solve the relaxed dual within these constraints:

$$
\begin{aligned}
&\min \mu_B \\
&s.t. \\
&\quad \mu_B \geq L_C\left(\widehat{\boldsymbol{\Psi}}^{B_l^K}, \boldsymbol{n}, \overline{\boldsymbol{\lambda}}^K\right) \\
&\quad \mu_B \geq L\left(\widehat{\boldsymbol{\Psi}}^{B_l^{K_P}}, \boldsymbol{n}, \overline{\boldsymbol{\lambda}}^{K_P}\right) \qquad \forall K_P \\
&\quad L^B \leq \boldsymbol{n} \leq U^B \\
&\quad 0 = \boldsymbol{A}\boldsymbol{n} - \boldsymbol{b}
\end{aligned}
\tag{124}
$$

Notice that the mass balance, which was dropped in the beginning of this derivation is now being enforced. L_C is evaluated by using equation (120). If the optimal value is greater that the smallest primal solution, it should be fathomed. If not, bounds are updated and it is stored and another B_l is chosen and the procedure is repeated. Once there are no more B_l available, the one that yielded the smallest solution is selected to be the mole number vector for the next iteration. If the difference between the smallest primal solution and the greatest relaxed dual solution is lesser than a certain pre-specified tolerance, the algorithm halts and provides an answer.

10 APPENDIX C

10.1 PARAMETERS FOR THE BENCHMARK CHEMICAL EQUILIBRIUM PROBLEMS

The following data were extracted from (Bonilla-Petricolet, et al., 2011).

10.1.1 Problem 1

The saturation pressure of each component in the vapor phase are given by $\log_{10} P_i^{sat} = A_i - \frac{B_i}{T+C_i}$, where P_i^{sat} is in Pa and T is in K. The coefficients A, B and C for each component is given in Table 12.

Table 12 - Saturation pressure coefficients for the components of Problem 1.

Component(*)	Parameters of pure component		
	A	B	C
1	9.95614	1440.52	-60.44
2	9.6845	1644.05	-39.63
3	9.22298	1238.71	-56.15
4	10.09171	1668.21	-45.14

(*) (1) Ethanol; (2) Acetic acid; (3) Ethyl acetate; (4) Water

The NRTL model parameters are shown in Table 13. We also assumed that $\alpha_{ij} = 0.3 \; \forall \; i, j$.

Table 13 - NRTL parameters for Problem 1.

Component(*)	τ_{ij} in the NRTL model			
	1	2	3	4
1	0	1.3941	0.6731	-0.2019
2	-1.0182	0	0.007	-0.4735
3	0.1652	0.5817	0	1.7002
4	2.1715	1.6363	1.9257	0

(*) (1) Ethanol; (2) Acetic acid; (3) Ethyl acetate; (4) Water

10.1.2 Problem 2

The saturation pressure of each component in the vapor phase is given by $\log_{10} P_i^{sat} = A_i - \frac{B_i}{T+C_i}$, where P_i^{sat} is in mmHg and T is in °C. The coefficients A, B and C for each component are given in Table 14.

Table 14 - Saturation pressure coefficients for the components of Problem 2.

Component(*)	Parameters of pure component		
	A	B	C
1	6.84132	923.201	239.99
2	8.07372	1578.23	239.382
3	6.87201	1116.825	224.744
4	6.80896	935.86	238.73

(*) (1) Isobutene; (2) Methanol; (3) MTBE; (4) Butane

The Wilson's model parameters are shown in Table 15.

Table 15 - Wilson parameters for Problem 2.

Component(*)	V_i	u_{ij} in the Wilson model (cal/mol)			
		1	2	3	4
1	93.33	-	169.9953	-60.1022	-
2	44.44	2576.8532	-	1483.2478	2283.8726
3	118.8	271.5669	-406.3902	-	-
4	100.39	-	382.3429	-	-

(*) (1) Isobutene; (2) Methanol; (3) MTBE; (4) Butane

10.1.3 Problems 3 and 6

The saturation pressure of each component in the vapor phase is given by:

$$\ln P_i^{sat} = A_i + \frac{B_i}{T} + C_i \ln T + D_i T^2, \quad i = 1,2,5$$

$$\ln P_i^{sat} = A_i + \frac{B_i}{T+C_i}, \quad i = 3,4$$

where P_i^{sat} is in Pa and T is in K. The coefficients A, B, C and D for each component are given in Table 16.

Table 16 - Saturation pressure coefficients for the components of Problems 3 and 6.

Component(*)	Parameters of pure component			
	A	B	C	D
1	74.527	-5232.2	-8.1482	8.474E-06
2	82.614	-5586.1	-9.4429	1.0858E-05
3	23.5347	-3661.468	-32.77	-
4	20.9441	-2936.223	-47.70385	-
5	81.624	-5578.5	-9.2354	9.4522E-06

(*) (1) 2-Methyl-1-Butene; (2) 2-Methyl-2-Butene; (3) Methanol; (4) TAME; (5) n-Pentane

The Wilson's model parameters are shown in Table 17.

Table 17 - Wilson parameters for Problems 3 and 6.

Component(*)	V_i	u_{ij} in the Wilson model (J/mol)				
		1	2	3	4	5
1	0.10868	-	478.8	1376.5	-611.75	326.74
2	0.10671	-477.94	-	968.81	-386.04	362.28
3	0.04069	9772.3	10147	-	4826.3	11749
4	0.13345	951.33	712.33	-177	-	1143.9
5	0.11613	-194.18	-265.49	1946.7	-447.84	-

(*) (1) 2-Methyl-1-Butene; (2) 2-Methyl-2-Butene; (3) Methanol; (4) TAME; (5) n-Pentane

10.1.4 Problem 4

The UNIQUAC model parameters are shown in Table 15.

Table 18 - UNIQUAC parameters for Problem 4.

Component(*)	Q	R_u	u_{ij} in the UNIQUAC model (cal/mol)			
			1	2	3	4
1	2.072	2.2024	-	-131.7686	-343.593	-298.4344
2	3.052	3.4543	148.2833	-	68.0083	82.5336

3	1.4	0.92	527.9269	581.1471	-	394.2396
4	4.196	4.8724	712.2349	24.6386	756.4163	-

(*) (1) Acetic acid; (2) n-Butanol; (3) Water; (4) n-Butyl acetate

10.1.5 Problem 7

The Margules model parameters are shown in Table 19.

Table 19 - Margules parameters for Problem 7.

Component	A_{ij} in the Margules solution model (K)		
	1	2	3
1	0	478.6	1074.484
2	478.6	0	626.9
3	1074.484	626.9	0

10.1.6 Problem 8

The NRTL model parameters are shown in Table 20 and in Table 21.

Table 20 - NRTL parameters u_{ij} for Problem 8.

Component(*)	u_{ij} in the NRTL model			
	1	2	3	4
1	0	1850.2001	79.4397	-327.5173
2	-80.4396	0	667.4489	-219.7238
3	369.0624	3280.604	0	-484.8901
4	256.8999	842.6079	1126.4792	0

Table 21 - NRTL parameters α_{ij} for Problem 8.

Component(*)	α_{ij} in the NRTL model			
	1	2	3	4
1	-	0.3	0.3006	0.3044
2	0.3	-	0.2564	0.2997
3	0.3006	0.2564	-	0.3
4	0.3044	0.2997	0.3	-

11 APPENDIX D

11.1 ANALYTIC GRADIENTS AND HESSIANS FOR GIBBS ENERGY.

As mentioned earlier, the Gibbs free energy functions that underwent constrained optimization took one of the following forms (Bonilla-Petricolet, et al., 2011):

$$F = \Delta g - \left(n_{i,1} + n_{i,2}\right) \ln K_{eq,i} \tag{125}$$

In the case of vapor-liquid equilibrium (VLE) problems:

$$\Delta g = \sum_{i=1}^{c} n_{i,1} \ln\left(x_{i,1}\gamma_{i,1}\right) + \sum_{i=1}^{c} n_{i,2} \ln\left(x_{i,2}P/P_i^{sat}\right) \tag{126}$$

For liquid-liquid equilibrium (LLE) problems:

$$\Delta g = \sum_{i=1}^{c} n_{i,1} \ln\left(x_{i,1}\gamma_{i,1}\right) + \sum_{i=1}^{c} n_{i,2} \ln\left(x_{i,2}\gamma_{i,2}\right) \tag{127}$$

Here *c* is the total number of components in the system under consideration and the subscripts 1 and 2 refer to different phases. The above formulae are sums of terms which belong to one of the following three types:

$$\begin{aligned} C_{i,p}^{1} &= n_{i,p} \ln x_{i,p} \\ C_{i,p}^{2} &= n_{i,p} \ln x_{i,p}P/P_i^{sat} \\ NC_{i,p} &= n_{i,p} \ln \gamma_{i,p} \end{aligned} \tag{128}$$

The terms of the forms $C_{i,p}^{1}$ and $C_{i,p}^{2}$ are convex, whereas those that take the form $NC_{i,p}$ are potentially nonconvex. That observation enables us to simplify the notation for our objective functions:

For VLE:

$$F = \sum_{i=1}^{c} \left(C_{i,1}^{1} + NC_{i,1}\right) + \sum_{i=1}^{c} C_{i,2}^{2} - \left(n_{i,1} + n_{i,2}\right) \ln K_{eq,i} \tag{129}$$

For LLE:

$$F = \sum_{i=1}^{c} \left(C_{i,1}^{1} + NC_{i,1}\right) + \sum_{i=1}^{c} \left(C_{i,2}^{1} + NC_{i,2}\right) - \left(n_{i,1} + n_{i,2}\right) \ln K_{eq,i} \tag{130}$$

It is now clear that in order to calculate the first and second derivatives of the objective function, it suffices to evaluate the first and second derivatives of the terms $C_{i,p}^{1}$, $C_{i,p}^{2}$ and $NC_{i,p}$.

11.1.1 Convex terms

We begin by evaluating the first derivatives of $C_{i,j}^{1}$ and $C_{i,j}^{2}$.

$$\frac{\partial C_{i,p}^{1}}{\partial n_{j,p}} = \frac{\partial \left(n_{i,p} \ln x_{i,p}\right)}{\partial n_{j,p}} = \begin{cases} \dfrac{n_{i,p}}{x_{i,p}} \left(\dfrac{\partial x_{i,p}}{\partial n_{j,p}}\right) + \ln x_{i,p} & if\ i = j \\ \dfrac{n_{i,p}}{x_{i,p}} \left(\dfrac{\partial x_{i,p}}{\partial n_{j,p}}\right) & if\ i \neq j \end{cases} \tag{131}$$

$$\frac{\partial C_{i,p}^{2}}{\partial n_{j,p}} = \frac{\partial \left[n_{i,p} \ln \left(\dfrac{x_{i,p} P}{P_i^{sat}}\right)\right]}{\partial n_{j,p}} = \begin{cases} \dfrac{n_{i,p}}{x_{i,p}} \left(\dfrac{\partial x_{i,p}}{\partial n_{j,p}}\right) + \ln x_{i,p} + \ln \dfrac{P}{P_i^{sat}} & if\ i = j \\ \dfrac{n_{i,p}}{x_{i,p}} \left(\dfrac{\partial x_{i,p}}{\partial n_{j,p}}\right) & if\ i \neq j \end{cases} \tag{132}$$

It follows that the second derivatives are given by:

$$\frac{\partial^2 C_{i,p}^{1}}{\partial n_{j,p} \partial n_{k,p}} = \frac{\partial^2 C_{i,p}^{2}}{\partial n_{j,p} \partial n_{k,p}} = \tag{133}$$

$$\begin{cases} \frac{\partial\left(\frac{n_{i,p}}{x_{i,p}}\right)}{\partial n_{k,p}}\left(\frac{\partial x_{i,p}}{\partial n_{j,p}}\right)+\frac{n_{i,p}}{x_{i,p}}\frac{\partial^2 x_{i,p}}{\partial n_{j,p}\partial n_{k,p}}+\frac{1}{x_{i,p}}\left(\frac{\partial x_{i,p}}{\partial n_{k,p}}\right) & if\ i=j \\ \frac{\partial\left(\frac{n_{i,p}}{x_{i,p}}\right)}{\partial n_{k,p}}\left(\frac{\partial x_{i,p}}{\partial n_{j,p}}\right)+\frac{n_{i,p}}{x_{i,p}}\frac{\partial^2 x_{i,p}}{\partial n_{j,p}\partial n_{k,p}} & if\ i\neq j \end{cases}$$

Where

$$\frac{\partial\left(\frac{n_{i,p}}{x_{i,p}}\right)}{\partial n_{k,p}}=\frac{\left(\frac{\partial n_{i,p}}{\partial n_{k,p}}\right)x_{i,p}-\left(\frac{\partial x_{i,p}}{\partial n_{k,p}}\right)n_{i,p}}{x_{i,p}^2} \tag{134}$$

And, of course:

$$\left(\frac{\partial n_{i,p}}{\partial n_{k,p}}\right)=\begin{cases}1 & if\ i=k\\ 0 & if\ i\neq k\end{cases} \tag{135}$$

All derivatives evaluated so far have been left in terms of mole fractions' derivatives. Let us now proceed to evaluating them. Upon defining the molar fractions as:

$$x_{i,p}=\frac{n_{i,p}}{\sum_{l=1}^{c} n_{l,p}} \tag{136}$$

It follows that:

$$\frac{\partial x_{i,p}}{\partial n_{j,p}}=\begin{cases}\frac{\sum_{l=1}^{c} n_{l,p}-n_{i,p}}{\left(\sum_{l=1}^{c} n_{l,p}\right)^2} & \text{if } i=j\\ \frac{-n_{i,p}}{\left(\sum_{l=1}^{c} n_{l,p}\right)^2} & \text{if } i\neq j\end{cases} \tag{137}$$

$$\frac{\partial^2 x_{i,p}}{\partial n_{j,p}\partial n_{k,p}}=$$
$$\begin{cases}\frac{\left[1-\left(\frac{\partial n_{i,p}}{\partial n_{k,p}}\right)\right]\left(\sum_{l=1}^{c} n_{l,p}\right)^2-2\left(\sum_{l=1}^{c} n_{l,p}\right)\left(\sum_{l=1}^{c} n_{l,p}-n_{i,p}\right)}{\left(\sum_{l=1}^{c} n_{l,p}\right)^4} & \text{if } i=j\\ \frac{-\left(\frac{\partial n_{i,p}}{\partial n_{k,p}}\right)\left(\sum_{l=1}^{c} n_{l,p}\right)^2-2\left(\sum_{l=1}^{c} n_{l,p}\right)\left(\sum_{l=1}^{c} n_{l,p}-n_{i,p}\right)}{\left(\sum_{l=1}^{c} n_{l,p}\right)^4} & \text{if } i\neq j\end{cases} \tag{138}$$

By plugging the formulae described in equations 137 and 138 in equations 131, 132 and 133, we can calculate analytically the gradient and hessian of all convex terms. Let us move on to the non-convex terms.

11.1.2 Non-convex terms

From the definition of the non-convex terms:

$$\frac{\partial NC_{i,p}}{\partial n_{j,p}} = \frac{\partial\left(n_{i,p}\ln\gamma_{i,p}\right)}{\partial n_{j,p}} = \frac{\partial n_{i,p}}{\partial n_{j,p}} + \ln\gamma_{i,p} + n_{i,p}\frac{\partial\left(\ln\gamma_{i,p}\right)}{\partial n_{j,p}} \tag{139}$$

For some activity models, it is more natural and convenient to express the derivatives as a function of $\ln\gamma_{i,p}$, as opposed to expanding them. For other models, further expansion yields:

$$\frac{\partial NC_{i,p}}{\partial n_{j,p}} = \frac{\partial n_{i,p}}{\partial n_{j,p}} + \ln\gamma_{i,p} + \frac{n_{i,p}}{\gamma_{i,p}}\left(\frac{\partial\gamma_{i,p}}{\partial n_{j,p}}\right) \tag{140}$$

Now, for the second derivatives:

$$\frac{\partial^2 NC_{i,p}}{\partial n_{j,p}\partial n_{k,p}} = \frac{\partial\left(\ln\gamma_{i,p}\right)}{\partial n_{k,p}} + \left(\frac{\partial n_{i,p}}{\partial n_{k,p}}\right)\frac{\partial\left(\ln\gamma_{i,p}\right)}{\partial n_{j,p}} + n_{i,p}\frac{\partial^2\left(\ln\gamma_{i,p}\right)}{\partial n_{j,p}\partial n_{k,p}} \tag{141}$$

Expansion yields:

$$\frac{\partial^2 NC_{i,p}}{\partial n_{j,p}\partial n_{k,p}} = \frac{1}{\gamma_{i,p}}\frac{\partial\gamma_{i,p}}{\partial n_{k,p}} + \frac{\partial\left(\frac{n_{i,p}}{\gamma_{i,p}}\right)}{\partial n_{k,p}}\frac{\partial\gamma_{i,p}}{\partial n_{j,p}} + \frac{n_{i,p}}{\gamma_{i,p}}\frac{\partial^2\gamma_{i,p}}{\partial n_{j,p}\partial n_{k,p}}$$

Where (142)

$$\frac{\partial\left(\frac{n_{i,p}}{\gamma_{i,p}}\right)}{\partial n_{k,p}} = \frac{\frac{\partial n_{i,p}}{\partial n_{k,p}}\gamma_{i,p} - \frac{\partial \gamma_{i,p}}{\partial n_{k,p}} n_{i,p}}{\gamma_{i,p}^2}$$

The derivatives that depend on $\gamma_{i,p}$ depend, therefore, on the model used. We now move on to describe this dependency.

11.1.2.1 Margules model

We've already described the Margules model and, for convenience, we display it here again with slight modifications, in order to keep the notation consistent.

$$\ln \gamma_{i,p} = \frac{1}{2T}\sum_{a=1}^{c}\sum_{b=1}^{c}\left(A_{a,i} + A_{b,i} - A_{a,b}\right)x_{a,p}x_{b,p} \tag{143}$$

The model is written in terms of the logarithm of the activity coefficient. For that reason, we will be concerned with finding derivatives that are expressed in terms of $\ln \gamma_{i,p}$.

$$\frac{\partial\left(\ln \gamma_{i,p}\right)}{\partial n_{j,p}} = \frac{1}{2T}\sum_{a=1}^{c}\sum_{b=1}^{c}\left[\left(A_{a,i} + A_{b,i} - A_{a,b}\right)\frac{\partial\left(x_{a,p}x_{b,p}\right)}{\partial n_{j,p}}\right]$$

Where

$$\frac{\partial\left(x_{a,p}x_{b,p}\right)}{\partial n_{j,p}} = x_{a,p}\frac{\partial x_{b,p}}{\partial n_{j,p}} + x_{b,p}\frac{\partial x_{a,p}}{\partial n_{j,p}} \tag{144}$$

It follows that the second derivatives are given by:

$$\frac{\partial^2\left(\ln \gamma_{i,p}\right)}{\partial n_{j,p}\partial n_{k,p}} = \frac{1}{2T}\sum_{a=1}^{c}\sum_{b=1}^{c}\left[\left(A_{a,i} + A_{b,i} - A_{a,b}\right)\frac{\partial^2\left(x_{a,p}x_{b,p}\right)}{\partial n_{j,p}\partial n_{k,p}}\right]$$

Where

$$\frac{\partial^2\left(x_{a,p}x_{b,p}\right)}{\partial n_{j,p}\partial n_{k,p}} = \frac{\partial x_{a,p}}{\partial n_{k,p}}\frac{\partial x_{b,p}}{\partial n_{j,p}} + x_{a,p}\frac{\partial^2 x_{b,p}}{\partial n_{j,p}\partial n_{k,p}} + \frac{\partial x_{b,p}}{\partial n_{k,p}}\frac{\partial x_{a,p}}{\partial n_{j,p}} + x_{b,p}\frac{\partial^2 x_{a,p}}{\partial n_{j,p}\partial n_{k,p}} \tag{145}$$

11.1.2.2 Wilson model

As before, we begin by readapting our previous definition in order to better fit our current notations.

$$\ln \gamma_{i,p} = 1 - \ln\left(\sum_{a=1}^{c} x_{a,p}\Lambda_{i,a}\right) - \sum_{b=1}^{c}\left(\frac{x_{b,p}\Lambda_{b,i}}{\sum_{a=1}^{c} x_{a,p}\Lambda_{b,a}}\right) \tag{146}$$

In order to simplify the next derivations, we shall define:

$$\begin{aligned} X_i &= \sum_{a=1}^{c} x_{a,p}\Lambda_{i,a} \\ Y_{i,b} &= x_{b,p}\Lambda_{b,i} \\ Z_b &= \sum_{a=1}^{c} x_{a,p}\Lambda_{b,a} \end{aligned} \tag{147}$$

Wilson model, then, becomes:

$$\ln \gamma_{i,p} = 1 - \ln X_i - \sum_{b=1}^{c}\left(\frac{Y_{i,b}}{Z_b}\right) \tag{148}$$

The first derivatives are, in terms of the newly defined variables:

$$\frac{\partial\left(\ln \gamma_{i,p}\right)}{\partial n_{j,p}} = -\frac{1}{X_i}\frac{\partial X_i}{\partial n_{j,p}} - \sum_{b=1}^{c}\left(\frac{\frac{\partial Y_{i,b}}{\partial n_{j,p}}Z_b - \frac{\partial Z_b}{\partial n_{j,p}}Y_{i,b}}{Z_b^2}\right) \tag{149}$$

And the first derivatives of the new variables are:

$$\begin{aligned} \frac{\partial X_i}{\partial n_{j,p}} &= \sum_{a=1}^{c}\frac{\partial x_{a,p}}{\partial n_{j,p}}\Lambda_{i,a} \\ \frac{\partial Y_{i,b}}{\partial n_{j,p}} &= \frac{\partial x_{b,p}}{\partial n_{j,p}}\Lambda_{b,i} \\ \frac{\partial Z_b}{\partial n_{j,p}} &= \sum_{a=1}^{c}\frac{\partial x_{a,p}}{\partial n_{j,p}}\Lambda_{b,a} \end{aligned} \tag{150}$$

As for the second derivatives:

$$\frac{\partial^2(\ln\gamma_{i,p})}{\partial n_{j,p}\partial n_{k,p}} = \frac{1}{X_i^2}\frac{\partial X_i}{\partial n_{k,p}}\frac{\partial X_i}{\partial n_{j,p}} - \frac{\partial^2 X_i}{\partial n_{j,p}\partial n_{k,p}}\frac{1}{X_i}$$
$$-\sum_{b=1}^{c}\left(\frac{\frac{\partial^2 Y_{i,b}}{\partial n_{j,p}\partial n_{k,p}}Z_b + \frac{\partial Y_{i,b}}{\partial n_{j,p}}\frac{\partial Z_b}{\partial n_{k,p}} - \frac{\partial^2 Z_b}{\partial n_{j,p}\partial n_{k,p}}Y_{i,b} - \frac{\partial Y_{i,b}}{\partial n_{k,p}}\frac{\partial Z_b}{\partial n_{j,p}}}{Z_b^4}\right) \quad (151)$$

And

$$\frac{\partial^2 X_i}{\partial n_{j,p}\partial n_{k,p}} = \sum_{a=1}^{c}\frac{\partial^2 x_{a,p}}{\partial n_{j,p}\partial n_{k,p}}\Lambda_{i,a}$$
$$\frac{\partial^2 Y_{i,b}}{\partial n_{j,p}\partial n_{k,p}} = \frac{\partial^2 x_{b,p}}{\partial n_{j,p}\partial n_{k,p}}\Lambda_{b,i}$$
$$\frac{\partial^2 Z_b}{\partial n_{j,p}\partial n_{k,p}} = \sum_{a=1}^{c}\frac{\partial^2 x_{a,p}}{\partial n_{j,p}\partial n_{k,p}}\Lambda_{b,a} \quad (152)$$

11.1.2.3 NRTL model

We proceed exactly as before:

$$\ln\gamma_{i,p} = \frac{\sum_{a=1}^{c}\tau_{a,i}G_{a,i}x_{a,p}}{\sum_{a=1}^{c}G_{a,i}x_{a,p}} + \sum_{a=1}^{c}\frac{G_{i,a}x_{a,p}}{\sum_{b=1}^{c}G_{i,b}x_{b,p}}\left(\tau_{i,a} - \frac{\sum_{b=1}^{c}\tau_{b,a}G_{b,a}x_{b,p}}{\sum_{b=1}^{c}G_{b,a}x_{b,p}}\right) \quad (153)$$

Again, we introduce new support variables:

$$X_i = \frac{\sum_{a=1}^{c}\tau_{a,i}G_{a,i}x_{a,p}}{\sum_{a=1}^{c}G_{a,i}x_{a,p}}$$
$$Y_{i,a} = \frac{G_{i,a}x_{a,p}}{\sum_{b=1}^{c}G_{i,b}x_{b,p}}$$
$$Z_{i,a} = \tau_{i,a} - \frac{\sum_{b=1}^{c}\tau_{b,a}G_{b,a}x_{b,p}}{\sum_{b=1}^{c}G_{b,a}x_{b,p}} \quad (154)$$

The model, then, becomes:

$$\ln \gamma_{i,p} = X_i + \sum_{a=1}^{c} Y_{i,a} Z_{i,a} \tag{155}$$

First derivatives:

$$\frac{\partial(\ln \gamma_{i,p})}{\partial n_{j,p}} = \frac{\partial X_i}{\partial n_{j,p}} + \sum_{a=1}^{c} \frac{\partial Y_{i,a}}{\partial n_{j,p}} Z_{i,a} + \frac{\partial Z_{i,a}}{\partial n_{j,p}} Y_{i,a}$$

Where

$$\frac{\partial X_i}{\partial n_{j,p}} = \frac{\left(\sum_{a=1}^{c} \tau_{a,i} G_{a,i} \frac{\partial x_{a,p}}{\partial n_{j,p}}\right)\left(\sum_{a=1}^{c} G_{a,i} x_{a,p}\right) - \left(\sum_{a=1}^{c} G_{a,i} \frac{\partial x_{a,p}}{\partial n_{j,p}}\right)\left(\sum_{a=1}^{c} \tau_{a,i} G_{a,i} x_{a,p}\right)}{\left(\sum_{a=1}^{c} G_{a,i} x_{a,p}\right)^2}$$

$$\frac{\partial Y_{i,a}}{\partial n_{j,p}} = \frac{G_{i,a} \frac{\partial x_{a,p}}{\partial n_{j,p}}\left(\sum_{b=1}^{c} G_{i,b} x_{b,p}\right) - \left(\sum_{b=1}^{c} G_{i,b} \frac{\partial x_{b,p}}{\partial n_{j,p}}\right)\left(G_{i,a} x_{a,p}\right)}{\left(\sum_{b=1}^{c} G_{i,b} x_{b,p}\right)^2}$$

$$\frac{\partial Z_{i,a}}{\partial n_{j,p}} = \frac{\left(\sum_{b=1}^{c} \tau_{b,a} G_{b,a} \frac{\partial x_{b,p}}{\partial n_{j,p}}\right)\left(\sum_{b=1}^{c} G_{b,a} x_{b,p}\right) - \left(\sum_{b=1}^{c} G_{b,a} \frac{\partial x_{b,p}}{\partial n_{j,p}}\right)\left(\sum_{b=1}^{c} \tau_{b,a} G_{b,a} x_{b,p}\right)}{\left(\sum_{b=1}^{c} G_{b,a} x_{b,p}\right)^2} \tag{156}$$

Second derivatives:

$$\frac{\partial^2(\ln \gamma_{i,p})}{\partial \quad \partial} = \tag{157}$$

$$\frac{\partial^2 X_i}{\partial n_{j,p}\partial n_{k,p}} + \sum_{a=1}^{c}\left[\frac{\partial^2 Y_{i,a}}{\partial n_{j,p}\partial n_{k,p}} Z_{i,a} + \frac{\partial Y_{i,a}}{\partial n_{j,p}}\frac{\partial Z_{i,a}}{\partial n_{k,p}} + \frac{\partial^2 Z_{i,a}}{\partial n_{j,p}\partial n_{k,p}} Y_{i,a} + \frac{\partial Z_{i,a}}{\partial n_{j,p}}\frac{\partial Y_{i,a}}{\partial n_{k,p}}\right]$$

By further introducing new variables:

$$\frac{\partial X_i}{\partial n_{j,p}} = \frac{XA - XB}{XC^2}$$

$$\frac{\partial Y_{i,a}}{\partial n_{j,p}} = \frac{YA - YB}{YC^2}$$

$$\frac{\partial Z_{i,a}}{\partial n_{j,p}} = \frac{ZA - ZB}{ZC^2} \qquad (158)$$

Which leads to:

$$\frac{\partial^2 X_i}{\partial n_{j,p}\partial n_{k,p}} = \frac{\left(\frac{\partial XA}{\partial n_{k,p}} - \frac{\partial XB}{\partial n_{k,p}}\right)XC^2 - 2XC\frac{\partial XC}{\partial n_{k,p}}(XA - XB)}{XC^4}$$

$$\frac{\partial^2 Y_{i,a}}{\partial n_{j,p}\partial n_{k,p}} = \frac{\left(\frac{\partial YA}{\partial n_{k,p}} - \frac{\partial YB}{\partial n_{k,p}}\right)YC^2 - 2YC\frac{\partial YC}{\partial n_{k,p}}(YA - YB)}{YC^4}$$

$$\frac{\partial^2 Z_{i,a}}{\partial n_{j,p}\partial n_{k,p}} = \frac{\left(\frac{\partial ZA}{\partial n_{k,p}} - \frac{\partial ZB}{\partial n_{k,p}}\right)ZC^2 - 2ZC\frac{\partial ZC}{\partial n_{k,p}}(ZA - ZB)}{ZC^4} \qquad (159)$$

Finally:

$$\frac{\partial XA}{\partial n_{k,p}} = \left(\sum_{a=1}^{c} \tau_{a,i} G_{a,i} \frac{\partial^2 x_{a,p}}{\partial n_{j,p} \partial n_{k,p}}\right)\left(\sum_{a=1}^{c} G_{a,i} x_{a,p}\right) + \left(\sum_{a=1}^{c} G_{a,i} \frac{\partial x_{a,p}}{\partial n_{k,p}}\right)\left(\sum_{a=1}^{c} \tau_{a,i} G_{a,i} \frac{\partial x_{a,p}}{\partial n_{j,p}}\right)$$

$$\frac{\partial XB}{\partial n_{k,p}} = \left(\sum_{a=1}^{c} G_{a,i} \frac{\partial x_{a,p}}{\partial n_{k,p}}\right)\left(\sum_{a=1}^{c} \tau_{a,i} G_{a,i} x_{a,p}\right) + \left(\sum_{a=1}^{c} G_{a,i} x_{a,p}\right)\left(\sum_{a=1}^{c} \tau_{a,i} G_{a,i} \frac{\partial x_{a,p}}{\partial n_{k,p}}\right)$$

$$\frac{\partial XC}{\partial n_{k,p}} = \sum_{a=1}^{c} G_{a,i} \frac{\partial x_{a,p}}{\partial n_{k,p}}$$

$$\frac{\partial YA}{\partial n_{k,p}} = G_{i,a} \frac{\partial^2 x_{a,p}}{\partial n_{j,p} \partial n_{k,p}}\left(\sum_{b=1}^{c} G_{i,b} x_{b,p}\right) + G_{i,a} \frac{\partial x_{a,p}}{\partial n_{j,p}}\left(\sum_{b=1}^{c} G_{i,b} \frac{\partial x_{b,p}}{\partial n_{k,p}}\right)$$

$$\frac{\partial YB}{\partial n_{k,p}} = \left(\sum_{b=1}^{c} G_{i,b} \frac{\partial^2 x_{b,p}}{\partial n_{j,p} \partial n_{k,p}}\right)\left(G_{i,a} x_{a,p}\right) + \left(\sum_{b=1}^{c} G_{i,b} \frac{\partial x_{b,p}}{\partial n_{j,p}}\right)\left(G_{i,a} \frac{\partial x_{a,p}}{\partial n_{k,p}}\right)$$

$$\frac{\partial YC}{\partial n_{k,p}} = \sum_{b=1}^{c} G_{i,b} \frac{\partial x_{b,p}}{\partial n_{k,p}}$$

$$\frac{\partial ZA}{\partial n_{k,p}} = \left(\sum_{b=1}^{c} \tau_{b,a} G_{b,a} \frac{\partial^2 x_{b,p}}{\partial n_{j,p} \partial n_{k,p}}\right)\left(\sum_{b=1}^{c} G_{b,a} x_{b,p}\right) + \left(\sum_{b=1}^{c} \tau_{b,a} G_{b,a} \frac{\partial x_{b,p}}{\partial n_{j,p}}\right)\left(\sum_{b=1}^{c} G_{b,a} \frac{\partial x_{b,p}}{\partial n_{k,p}}\right)$$

$$\frac{\partial ZB}{\partial n_{k,p}} = \left(\sum_{b=1}^{c} G_{b,a} \frac{\partial^2 x_{b,p}}{\partial n_{j,p} \partial n_{k,p}}\right)\left(\sum_{b=1}^{c} \tau_{b,a} G_{b,a} x_{b,p}\right) + \left(\sum_{b=1}^{c} G_{b,a} \frac{\partial x_{b,p}}{\partial n_{j,p}}\right)\left(\sum_{b=1}^{c} \tau_{b,a} G_{b,a} \frac{\partial x_{b,p}}{\partial n_{k,p}}\right)$$

$$\frac{\partial ZC}{\partial n_{k,p}} = \sum_{b=1}^{c} G_{b,a} \frac{\partial x_{b,p}}{\partial n_{k,p}} \tag{160}$$

11.1.2.4 UNIQUAC model

Lastly, we move on to the UNIQUAC model and proceed exactly as before.

$$\ln \gamma_{i,p} = \ln \gamma_{i,p}^{E} + \ln \gamma_{i,p}^{R}$$

$$\ln \gamma_{i,p}^{E} = \ln \frac{\phi_{i,p}}{x_{i,p}} + 5Q_i \ln \frac{\phi_{i,p}}{x_{i,p}} + l_i - \frac{\phi_{i,p}}{x_{i,p}} \sum_{a=1}^{c} x_{a,p} l_a \tag{161}$$

$$\ln \gamma_{i,k}^{R} = Q_i \left[1 - \ln\left(\sum_{a=1}^{c} \theta_a \tau_{ai} \right) - \sum_{a=1}^{c} \left(\frac{\theta_a \tau_{ai}}{\sum_{b=1}^{c} \theta_b \tau_{bi}} \right) \right]$$

$$\theta_{i,p} = \frac{Q_i x_{i,p}}{\sum_{l=1}^{c} Q_l x_{l,p}}$$

$$\phi_{i,p} = \frac{R_{u,i} x_{i,p}}{\sum_{l=1}^{c} R_{u,l} x_{l,p}}$$

First derivatives:

$$\frac{\partial(\ln \gamma_{i,p})}{\partial n_{j,p}} = \frac{\partial(\ln \gamma_{i,p}^{E})}{\partial n_{j,p}} + \frac{\partial(\ln \gamma_{i,p}^{R})}{\partial n_{j,p}}$$

$$\frac{\partial(\ln \gamma_{i,p}^{E})}{\partial n_{j,p}} = \frac{x_{i,p}}{\phi_{i,p}} \left(\frac{\frac{\partial \phi_{i,p}}{\partial n_{j,p}} x_{i,p} - \frac{\partial x_{i,p}}{\partial n_{j,p}} \phi_{i,p}}{x_{i,p}^{2}} \right) (1 + 5Q_i)$$

$$- \left(\frac{\frac{\partial \phi_{i,p}}{\partial n_{j,p}} x_{i,p} - \frac{\partial x_{i,p}}{\partial n_{j,p}} \phi_{i,p}}{x_{i,p}^{2}} \right) \sum_{a=1}^{c} x_{a,p} l_a - \frac{\phi_{i,p}}{x_{i,p}} \sum_{a=1}^{c} \frac{\partial x_{a,p}}{\partial n_{j,p}} l_a$$

(162)

It is again helpful to introduce support variables:

$$X_{i,a} = \theta_a \tau_{ai}$$

$$Y_i = \sum_{b=1}^{c} \theta_b \tau_{bi}$$

$$\frac{\partial X_{i,a}}{\partial n_{j,p}} = \frac{\partial \theta_{a,p}}{\partial n_{j,p}} \tau_{ai}$$

$$\frac{\partial Y_i}{\partial n_{j,p}} = \sum_{b=1}^{c} \frac{\partial \theta_{b,p}}{\partial n_{j,p}} \tau_{bi}$$

$$\frac{\partial^2 X_{i,a}}{\partial n_{j,p} \partial n_{k,p}} = \frac{\partial^2 \theta_{a,p}}{\partial n_{j,p} \partial n_{k,p}} \tau_{ai}$$

$$\frac{\partial^2 Y_i}{\partial n_{j,p} \partial n_{k,p}} = \sum_{b=1}^{c} \frac{\partial^2 \theta_{b,p}}{\partial n_{j,p} \partial n_{k,p}} \tau_{bi}$$

(163)

Which leads to:

$$\frac{\partial\left(\ln\gamma_{i,p}^{R}\right)}{\partial n_{j,p}} = Q_i\left[-\frac{1}{\sum_{a=1}^{c}\theta_{a,p}\tau_{ai}}\sum_{a=1}^{c}\frac{\partial\theta_{a,p}}{\partial n_{j,p}}\tau_{ai} - \sum_{a=1}^{c}\frac{\frac{\partial X_{i,a}}{\partial n_{j,p}}Y_i - \frac{\partial Y_i}{\partial n_{j,p}}X_{i,a}}{Y_{i,a}^2}\right] \tag{164}$$

We also have:

$$\frac{\partial\theta_{i,p}}{\partial n_{j,p}} = \frac{Q_i\frac{\partial x_{i,p}}{\partial n_{j,p}}\left(\sum_{l=1}^{c}Q_l x_{l,p}\right) - Q_i x_{i,p}\left(\sum_{l=1}^{c}Q_l\frac{\partial x_{l,p}}{\partial n_{j,p}}\right)}{\left(\sum_{l=1}^{c}Q_l x_{l,p}\right)^2}$$

$$\frac{\partial\phi_{i,p}}{\partial n_{j,p}} = \frac{R_{u,i}\frac{\partial x_{i,p}}{\partial n_{j,p}}\left(\sum_{l=1}^{c}R_{u,l}x_{l,p}\right) - R_{u,i}x_{i,p}\left(\sum_{l=1}^{c}R_{u,l}\frac{\partial x_{l,p}}{\partial n_{j,p}}\right)}{\left(\sum_{l=1}^{c}R_{u,l}x_{l,p}\right)^2} \tag{165}$$

The second derivatives are, then:

$$\frac{\partial^2\left(\ln\gamma_{i,p}\right)}{\partial n_{j,p}\partial n_{k,p}} = \frac{\partial^2\left(\ln\gamma_{i,p}^{E}\right)}{\partial n_{j,p}\partial n_{k,p}} + \frac{\partial^2\left(\ln\gamma_{i,p}^{R}\right)}{\partial n_{j,p}\partial n_{k,p}} \tag{166}$$

For clarity, let us introduce yet another variable:

$$Z = \frac{\frac{\partial\phi_{i,p}}{\partial n_{j,p}}x_{i,p} - \frac{\partial x_{i,p}}{\partial n_{j,p}}\phi_{i,p}}{x_{i,p}^2}$$

$$\frac{\partial Z}{\partial n_{k,p}} = \frac{\frac{\partial^2\phi_{i,p}}{\partial n_{j,p}\partial n_{k,p}}x_{i,p} + \frac{\partial\phi_{i,p}}{\partial n_{j,p}}\frac{\partial x_{i,p}}{\partial n_{k,p}} - \frac{\partial^2 x_{i,p}}{\partial n_{j,p}\partial n_{k,p}}\phi_{i,p} - \frac{\partial\phi_{i,p}}{\partial n_{k,p}}\frac{\partial x_{i,p}}{\partial n_{j,p}}}{x_{i,p}^4} \tag{167}$$

Let us also rewrite $\frac{\partial \phi_{i,p}}{\partial n_{j,p}}$ as follows:

$$\frac{\partial \phi_{i,p}}{\partial n_{j,p}} = \frac{A_{i,j}}{B^2} \tag{168}$$

We may now express the second derivatives of $\phi_{i,p}$ more conveniently as:

$$\frac{\partial^2 \phi_{i,p}}{\partial n_{j,p} \partial n_{k,p}} = \frac{\frac{\partial A_{i,j}}{\partial n_{k,p}} B^2 - 2A_{i,j} B \frac{\partial B}{\partial n_{k,p}}}{B^4}$$

$$\begin{aligned}\frac{\partial A_{i,j}}{\partial n_{k,p}} = {} & R_{u,i} \frac{\partial^2 x_{i,p}}{\partial n_{j,p} \partial n_{k,p}} \left(\sum_{l=1}^{c} R_{u,l} x_{l,p} \right) + R_{u,i} \frac{\partial x_{i,p}}{\partial n_{j,p}} \left(\sum_{l=1}^{c} R_{u,l} \frac{\partial x_{l,p}}{\partial n_{k,p}} \right) \\ & - R_{u,i} \frac{\partial x_{i,p}}{\partial n_{k,p}} \left(\sum_{l=1}^{c} R_{u,l} \frac{\partial x_{l,p}}{\partial n_{j,p}} \right) \\ & - R_{u,i} x_{i,p} \left(\sum_{l=1}^{c} R_{u,l} \frac{\partial^2 x_{l,p}}{\partial n_{j,p} \partial n_{k,p}} \right)\end{aligned}$$

$$\frac{\partial B}{\partial n_{k,p}} = \sum_{l=1}^{c} R_{u,l} \frac{\partial x_{l,p}}{\partial n_{k,p}} \tag{169}$$

The same reasoning applies to $\theta_{i,p}$:

$$\frac{\partial \theta_{i,p}}{\partial n_{j,p}} = \frac{C_{i,j}}{D^2}$$

$$\frac{\partial^2 \theta_{i,p}}{\partial n_{j,p} \partial n_{k,p}} = \frac{\frac{\partial C_{i,j}}{\partial n_{k,p}} D^2 - 2C_{i,j} D \frac{\partial D}{\partial n_{k,p}}}{D^4}$$

$$\begin{aligned}\frac{\partial C_{i,j}}{\partial n_{k,p}} = {} & Q_i \frac{\partial^2 x_{i,p}}{\partial n_{j,p} \partial n_{k,p}} \left(\sum_{l=1}^{c} Q_l x_{l,p} \right) + Q_i \frac{\partial x_{i,p}}{\partial n_{j,p}} \left(\sum_{l=1}^{c} Q_l \frac{\partial x_{l,p}}{\partial n_{k,p}} \right) \\ & - Q_i \frac{\partial x_{i,p}}{\partial n_{k,p}} \left(\sum_{l=1}^{c} Q_l \frac{\partial x_{l,p}}{\partial n_{j,p}} \right) - Q_i x_{i,p} \left(\sum_{l=1}^{c} Q_l \frac{\partial^2 x_{l,p}}{\partial n_{j,p} \partial n_{k,p}} \right)\end{aligned} \tag{170}$$

$$\frac{\partial D}{\partial n_{k,p}} = \sum_{l=1}^{c} Q_l \frac{\partial x_{l,p}}{\partial n_{k,p}}$$

Now:

$$\frac{\partial^2 \left(\ln \gamma_{i,p}^{E}\right)}{\partial n_{j,p} \partial n_{k,p}} = (1 + 5Q_i)\left(Z^2 + \frac{x_{i,p}}{\phi_{i,p}} \frac{\partial Z}{\partial n_{k,p}}\right)$$
$$- \frac{\partial Z}{\partial n_{k,p}} \sum_{a=1}^{c} x_{a,p} l_a - Z \sum_{a=1}^{c} \frac{\partial x_{a,p}}{\partial n_{k,p}} l_a$$
$$+ Z \frac{x_{i,p}}{\phi_{i,p}} \sum_{a=1}^{c} \frac{\partial x_{a,p}}{\partial n_{j,p}} l_a - \frac{\phi_{i,p}}{x_{i,p}} \sum_{a=1}^{c} \frac{\partial^2 x_{a,p}}{\partial n_{j,p} \partial n_{k,p}} l_a \tag{171}$$

Lastly, it is possible to write the second derivatives of $\ln \gamma_{i,p}^{R}$ as the sum of two terms:

$$\frac{\partial^2 \left(\ln \gamma_{i,p}^{R}\right)}{\partial n_{j,p} \partial n_{k,p}} = T_1 + T_2 \tag{172}$$

Where:

$$T_1 = Q_i \left\{ \left[\frac{\sum_{a=1}^{c} \frac{\partial \theta_{a,p}}{\partial n_{k,p}} \tau_{ai}}{\left(\sum_{a=1}^{c} \theta_{a,p} \tau_{ai}\right)^2} \right] \left[\sum_{a=1}^{c} \frac{\partial \theta_{a,p}}{\partial n_{j,p}} \tau_{ai} \right] \right.$$
$$\left. + \left[\sum_{a=1}^{c} \frac{\partial^2 \theta_{a,p}}{\partial n_{j,p} \partial n_{k,p}} \tau_{ai} \right] \left[- \frac{1}{\sum_{a=1}^{c} \theta_{a,p} \tau_{ai}} \right] \right\}$$

$$T_2 = -Q_i \sum_{a=1}^{c} \frac{Y_{i,a}^2 T_3 - 2 Y_{i,a} \frac{\partial Y_{i,a}}{\partial n_{k,p}} \left(\frac{\partial X_{i,a}}{\partial n_{j,p}} Y_i - \frac{\partial Y_i}{\partial n_{j,p}} X_{i,a} \right)}{Y_{i,a}^4}$$

$$T_3 = \frac{\partial^2 X_{i,a}}{\partial n_{j,p} \partial n_{k,p}} Y_i + \frac{\partial X_{i,a}}{\partial n_{j,p}} \frac{\partial Y_i}{\partial n_{k,p}} - \frac{\partial^2 Y_i}{\partial n_{j,p} \partial n_{k,p}} X_{i,a} - \frac{\partial X_{i,a}}{\partial n_{k,p}} \frac{\partial Y_i}{\partial n_{j,p}}$$

(173)