A new algorithm for the automation of phase diagram calculation

Maria Emelianenko[1], Zi-Kui Liu[2] and Qiang Du[1]*

[1]Department of Mathematics
[2]Department of Materials Science and Engineering
The Pennsylvania State University
University Park, PA 16803, USA

Abstract

We propose a new algorithm for minimizing the Gibbs energy functional and constructing phase diagrams through utilizing geometric properties of the energy surfaces together with effective sampling techniques to improve on the starting points for the minimization. The new method possesses advantages over existing methods in terms of computational complexity and can be used to automate the calculation of phase equilibria in complex systems. Numerical results for binary and ternary systems are presented. Generalizations to higher dimensions are discussed.

PACS codes: 81.30.Bx, 64.60.-i, 02.60.Pn, 02.70.-c
Keywords: Phase diagrams; Stable equilibria; Free energy minimization; Automation; Numerical algorithm; Complexity

* Corresponding author. Tel.: 1-814-865-3674 fax: 1-814-865-3735.
E-mail address: qdu@math.psu.edu (Qiang Du)

1. Introduction

Phase diagrams are visual representations of the equilibrium phases in a material as a function of temperature, pressure and concentrations of the constituent components and are frequently used as basic blueprints for materials research and development. Under typical experimental conditions of constant pressure and temperature and a closed system, calculated phase equilibria are obtained via minimization of the total Gibbs energy of a system by adjusting the compositions and amounts of all individual phases in the system. As in any minimization procedure, the starting values play an important role due to the existence of many possible metastable states.

Many existing software packages lack the ability to automatically determine system properties from initial data and can produce metastable equilibria instead of stable ones or simply diverge if the initial guess is not good enough. Several algorithms were proposed to automate the process of finding suitable starting positions, all of which carry an increased computational cost. In this paper we make an attempt to improve on the existing strategies for automating phase diagram calculations by introducing a novel reduced complexity algorithm based on adaptive critical point detection approach. The main advantage of the new scheme lies in its ability to effectively reduce the total number of trial calculations by recognizing the importance of geometry specific properties of the Gibbs energies.

We start with an introduction to the necessary theoretical background and a short overview of existing techniques, which are succeeded by the detailed description of the new algorithm as well as its generalizations in section 3. Section 4 contains the results of several numerical calculations for binary and ternary systems. Some concluding remarks are made in section 5.

2. Theoretical aspects of phase diagram calculation

2.1. Mathematical model

Let us fix both the temperature and the pressure as independent system variables with a total of one mole of components. Let f_k be the total content of the *k*-th component in the system and f_k^i the content of the k-th component in the i-th phase and $f^{(i)}$ the number of moles of the phase i (by our assumption $\sum_{k=1}^{K} f_k^i = f^{(i)}$). We also let $f = (f_k)_{k=1,\ldots,K}$. Since it is easier to work with molar quantities, we use $x^{(i)} = (x_k^i)_{k=1,\ldots,K} = (f_k^i / f^{(i)})_{k=1,\ldots,K}$ to denote the vector consisting of mole fractions of the k-th component in the phase i and $G^{(i)}$ is the corresponding molar Gibbs energy. In this case, the equilibrium analysis of a K-component system with n phases leads to the following Gibbs energy minimization problem:

$$\begin{cases} \min\limits_{(\mathrm{f},x)} \left\{ \mathrm{G} = \sum\limits_{\mathrm{i}=1}^{\mathrm{n}} f^{(i)} G^{(i)}\left(x^{(i)}\right) \right\} \\ \sum\limits_{i=1}^{n} f^{(i)} x^{(i)} = f, \\ \sum\limits_{k=1}^{K} x_k^i = 1, \quad i = 1,\ldots n, \\ f^{(i)} \geq 0, x_k^i \geq 0 \end{cases} \tag{1}$$

The equality constraints given in the above equations arise from the preservation of both component and total mass. The minimization problem is also subject to the above natural inequality constraints that guarantee nonnegativity of contents and mole fractions for each of the phases.

Suppose $x^* = \left(f^{(i)}, x^{(i)}\right)^*$ is an extremum of G. An index set A is defined to include those indices whose corresponding inequality constraint belongs to the active set of constraints at x^*.

Recall that a local constrained extremum x^* coincides with an unconstrained critical point of the following Lagrangian, where $\lambda_i, \eta_k, \gamma_i, \tau_k^i$ are the Lagrange multipliers corresponding to the equality and inequality constraints introduced above:

$$L = \sum_{i=1}^{n} f^{(i)} G^{(i)}\left(x^{(i)}\right) - \sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{K} x_k^i - 1 \right) - \sum_{k=1}^{K} \eta_k \left(\sum_{i=1}^{n} f^{(i)} x_k^i - f_k \right) - \sum_{i \in A} \gamma_i f^{(i)} - \sum_{i,k \in A} \tau_k^i x_k^i . \tag{2}$$

Here, we may use the theory on the complementarity property of the constraints and the Lagrange multipliers so that only active constraints at the extremum point need to be considered. Notice, however, that if an inequality constraint given in (1) becomes active, the total content for some phases becomes zero, which implies that there are phases that do not take part in the

equilibrium. If these phases are known at the beginning, we can discard them and therefore reduce the problem dimension. However, it is sometimes difficult to determine which phases would form an equilibrium a priori, so a generalized approach is preferred.

We can, however, make the computation more efficient by monitoring the phase contents in the process of optimization and eliminate those phases whose contents become insignificant. The issue with this approach is that a phase content may accidentally become very small in the course of the numerical procedure, so discarding a phase completely without allowing it to reappear may lead to unwanted consequences. The usual tactic is to assign a small tolerance value $\varepsilon > 0$ to all phases whose contents are lower than ε during the equilibrium calculation. This assures that all phases have equal chances of contributing to the equilibrium state, at the same time making all inequality constraints inactive. In other words, we arrive at the following problem:

$$L = \sum_{i=1}^{n} f^{(i)} G^{(i)}\left(x^{(i)}\right) - \sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{K} x_k^i - 1 \right) - \sum_{k=1}^{K} \eta_k \left(\sum_{i=1}^{n} f^{(i)} x_k^i - f_k \right) \tag{3}$$

The well known Karush-Kuhn-Tucker theorem for optimization problems [1] asserts that at the critical point, the following set of first order conditions is met:

$$\begin{cases} \dfrac{\partial L}{\partial f^{(i)}}(x^*) = G^{(i)}\left(x^{(i)}\right) - \sum\limits_{k=1}^{K} \eta_k x_k^i = 0 \\ \dfrac{\partial L}{\partial x_k^i}(x^*) = f^{(i)} \dfrac{\partial G^{(i)}}{\partial x_k^i} - \lambda_i - f^{(i)} \eta_k = 0 \end{cases} \tag{4}$$

where $\lambda_i, \eta_k \geq 0$ at x^* and constraint equations in (1) are satisfied. We can now solve for η_k from the second equation and substitute into the first one to get:

$$\begin{cases} \eta_k = \dfrac{\partial G^{(i)}}{\partial x_k^i} - \dfrac{\lambda_i}{f^{(i)}} \\ G^{(i)}\left(x^{(i)}\right) - \sum\limits_{j=1}^{K} \dfrac{\partial G^{(i)}}{\partial x_j^i} x_j^i + \dfrac{\lambda_i}{f^{(i)}} = 0 \end{cases}$$

It follows that

$$\eta_k = \frac{\partial G^{(i)}}{\partial x_k^i} + G^{(i)}\left(x^{(i)}\right) - \sum_{j=1}^{K} \frac{\partial G^{(i)}}{\partial x_j^i} x_j^i \tag{5}$$

for all $i = 1, \ldots, n$. Notice that the expression at the right hand side is the full derivative of the Gibbs energy $\widetilde{G}^{(i)}\left((f_k^i)_{k=1,\ldots,n}\right) = f^{(i)} G^{(i)}\left(x^{(i)}\right)$ with respect to contents f_k^i of the k-th component in the i-th phase. In other words, making a change of variables back from molar quantities to

f_k^i, we conclude that for given temperature, pressure and overall composition, the minimum of the objective function satisfies the following equations:

$$\begin{cases} \mu_1^1 = \mu_1^2 = \ldots = \mu_1^{\,n} = \eta_1 \\ \qquad\qquad \ldots \\ \mu_K^1 = \mu_K^2 = \ldots = \mu_K^n = \eta_K \end{cases} \tag{6}$$

where $\mu_k^i = \partial \widetilde{G}^{(i)} / \partial f_k^i$. These equations are called Gibbs equilibrium conditions, which imply that the value of the chemical potential for each component *k* is the same in all phases $i = 1, \ldots n$.

Furthermore, from the first equation in (4), we get that $G^{(i)}\left(x^{(i)}\right) - G^{(j)}\left(x^{(j)}\right) = \eta^T\left(x^{(i)} - x^{(j)}\right)$. Coupled with (6), this implies the common tangent hyper-plane property in the *G*-*x* space with the Lagrange multiplier η being the normal to the plane. Such a property is well known for the phase diagram calculation.

It should be mentioned that these equations provide only necessary conditions. In order to guarantee that the solution found at this step is indeed a minimizer, one should verify that $w^T(\nabla^2 L)w > 0$ at x^* for all limiting directions w of a feasible sequence. Notice that at x^*,

$$w^T(\nabla^2 L)w = \sum_i w^{iT} \begin{pmatrix} f^{(i)}\left(\dfrac{\partial^2 G^{(i)}}{\partial x_k^i \partial x_j^i}\right) & \left(\dfrac{\partial G^{(i)}}{\partial x_k^i} - \eta_k\right) \\ \left(\dfrac{\partial G^{(i)}}{\partial x_k^i} - \eta_k\right) & 0 \end{pmatrix} w^i = \sum_i w^{iT} \begin{pmatrix} f^{(i)}\left(\dfrac{\partial^2 G^{(i)}}{\partial x_k^i \partial x_j^i}\right) & \dfrac{\lambda_i}{f^{(i)}} e \\ \dfrac{\lambda_i}{f^{(i)}} e & 0 \end{pmatrix} w^i .$$

Here, *e* denotes the column vector with all components being equal to 1. We may write any feasible direction as $w^i = (w_1^i, w_2^i)^T$ and then make it adhere to the space of constraints (see [1]), which in this case leads to $w_1^{iT} e = 0$. Thus, we get

$$w^T(\nabla^2 L)w = \sum_i f^{(i)} w_1^{iT} \left(\frac{\partial^2 G^{(i)}}{\partial x_k^i \partial x_j^i}\right) w_1^i .$$

It follows that the solution x^* of the unconstrained problem (3) is a local minimum of the total Gibbs energy of the system provided that the Hessian of the mixing energy is positive definite at x^*, otherwise it is possible that the point at hand is a maximum or a saddle. This observation explains why the regions of positive concavity are so important for the process of finding good initial guess that we are going to discuss in the following sections.

While the set of conditions provided by the KKT theorem is capable of identifying local solutions of the problem (3), phase diagrams often require the knowledge of stable equilibria of the system and hence call for a more careful analysis of the minimizers of the total Gibbs energy.

As we have already noted, any local solution should satisfy the common tangent hyper-plane property in the *G-x* space, whereas stable solutions would have to belong to the lower convex hull determined by these points. In order to eliminate points belonging to the interior of the convex hull at any stage of the algorithm, one can perform two kinds of tests. The first one, referred to as a stability check, consists in determining whether a given phase possesses the minimal energy among all phases considered at this point. The remaining coplanarity check identifies whether there are solutions lying below the plane determined by a set of test points. Clearly any subset of points on the boundary of the convex hull satisfies both of these tests, and we're going to exploit this fact later in designing our method for stable diagram construction.

Going back to equations (6), notice that they result in a system of nonlinear equations which should be solved numerically. Hence the success of the whole task of calculating phase equilibria depends on the effectiveness of the scheme chosen to solve the nonlinear system.

Technical implementations of the nonlinear solution procedure differ from algorithm to algorithm. The two most popular ones rely on the Newton-Raphson and the simplex methods for iterative solution of the system (6). All of the CALPHAD-type software tools use methods like the two-step method of Hillert [2, 3, 4] or the one step method of Lukas et al. [5] to minimize the Gibbs energy. Typical drawbacks of these strategies include the use of prior knowledge in providing suitable starting points and the possibility of divergence or convergence to metastable minima.

Other methods were proposed that attempt to get a direct solution to the minimization problem (1) either by constructing phase field boundaries or determining the minimum free energy surface directly [6]. These algorithms do not suffer from stability issues as much as the iterative methods mentioned above, but face problems with higher computational costs and the possible loss of information due to limited resolution.

We will focus our attention on improving the iterative solution adopted in the packages of the Thermocalc family. In doing so, we are going to follow the line of direct methods by recognizing the importance of geometrical information for the design of an efficient minimization scheme.

2.2 Geometrical considerations

As shown above, the procedure of finding solution to the minimization problem described here, from the geometric perspective, is nothing but a common tangent hyper-plane construction for the equilibrium phase surfaces in the $G - x$ space. Indeed, to give a more detailed illustration, let us consider the binary 2-phase case as an example. From the conservation of phase mass condition we have

$$x_1^1 = x^{(1)}; \quad x_2^1 = 1 - x^{(1)}$$
$$x_1^2 = x^{(2)}; \quad x_2^2 = 1 - x^{(2)} \quad .$$

Hence, the minimization problem can be written as

$$\begin{cases} \min\limits_{(f,x)}\{G = f^{(1)}G^{(1)}(x^{(1)}) + f^{(2)}G^{(2)}(x^{(2)})\} \\ f^{(1)}x^{(1)} + f^{(2)}x^{(2)} = f_1 \\ f^{(1)}(1-x^{(1)}) + f^{(2)}(1-x^{(2)}) = f_2 \end{cases}.$$

In simpler form,

$$\begin{cases} \min\limits_{(f,x)}\{G = f^{(1)}G^{(1)}(x^{(1)}) + f^{(2)}G^{(2)}(x^{(2)})\} \\ f^{(1)}x^{(1)} + f^{(2)}x^{(2)} - f_1 = 0 \\ f^{(1)} + f^{(2)} - (f_1 + f_2) = 0 \end{cases}.$$

By means of the Lagrange multipliers μ and η, we can represent the above system in form of the following unconstrained minimization problem:

$$L = f^{(1)}G^{(1)}(x^{(1)}) + f^{(2)}G^{(2)}(x^{(2)}) - \mu\left(f^{(1)}x^{(1)} + f^{(2)}x^{(2)} - f_1\right) - \eta\left(f^{(1)} + f^{(2)} - \left(f_1 + f_2\right)\right)$$

At an equilibrium, the partial derivatives of the Lagrangian with respect to $(\vec{f},\vec{x})$ become zero:

$$\frac{\partial L}{\partial x^{(1)}} = f^{(1)}\left(\frac{\partial \mathrm{G}^{(1)}(x^{(1)})}{\partial x^{(1)}} - \mu\right) = 0; \frac{\partial L}{\partial x^{(2)}} = f^{(2)}\left(\frac{\partial \mathrm{G}^{(2)}(x^{(2)})}{\partial x^{(2)}} - \mu\right) = 0$$

$$\frac{\partial \mathrm{L}}{\partial f^{(1)}} = G^{(1)}(x^{(1)}) - \mu x^{(1)} - \eta = 0$$

$$\frac{\partial \mathrm{L}}{\partial f^{(2)}} = G^{(2)}(x^{(2)}) - \mu x^{(2)} - \eta = 0.$$

It follows that $\eta = G^{(1)}(x^{(1)}) - \mu x^{(1)} = G^{(2)}(x^{(2)}) - \mu x^{(2)}$ (similar derivation can be found e.g. in [7]). Hence the solution should satisfy the following equations

$$\begin{cases} \mu = \dfrac{\partial \mathrm{G}^{(1)}(x^{(1)})}{\partial x^{(1)}} = \dfrac{\partial \mathrm{G}^{(2)}(x^{(2)})}{\partial x^{(2)}} \\ \mu = \dfrac{G^{(1)}(x^{(1)}) - G^{(2)}(x^{(2)})}{x^{(1)} - x^{(2)}} \end{cases}.$$

Geometrically, it is the common tangent line of Gibbs energy curves. Similar argument can be carried out in higher dimensions.

With this observation in mind, we are now ready to discuss the problems associated with the construction of phase diagrams using the existing algorithms and some improved techniques.

2.3 Existing algorithms and motivation

As mentioned earlier, the minimization procedures adopted in existing iterative-type software have the following drawbacks:

- (stability) They either fail to converge or perhaps converge to some metastable equilibrium when a starting point is taken too far from the desired minimizer.
- (user-dependence) The computer programs cannot independently determine the existence of a miscibility gap, hence some prior knowledge of system properties is required.

Figure 1 demonstrates the problem in producing a correct phase diagram for a system with a miscibility gap. The diagram in Figure 1(a) is the correct phase diagram of the Ca-Li-Na system at $T = 900K$ produced by specifying the "`set_miscibility_gap`" option, while the wrong diagram in Figure 1(b) is the result of calculations when this option is not provided by the user.

Mathematically speaking, a miscibility gap arises when the Gibbs energy of a phase exhibits multiple minima. We need to design an algorithm capable of predicting system properties of this kind from the initial data. Ultimately this algorithm may be used as a basis to automate phase diagram calculation process as a whole.

The problem with miscibility gaps has been addressed before. A solution has been proposed by Chen et al [8], [9], [10]. Their method relies on a discretization of composition axis in order to represent solution phases by a set of stoichiometric compounds. It performs a series of tests to reveal the pairs of points which can coexist in stable equilibrium. These tests include stability checks (discarding points with higher energy values) and "coplanarity" checks, which test a pair for stable 2-phase equilibrium. As soon as candidate pairs are identified, they are taken as initial approximations for a consecutive minimization procedure, which is supposed to lead to the exact solution. In a two-phase case with N stoichiometric compounds the coplanarity (colinearity) condition holds, if

$$\frac{\begin{vmatrix} G_s & G_i & G_j \\ x_{1,s} & x_{1,i} & x_{1,j} \\ x_{2,s} & x_{2,i} & x_{2,j} \end{vmatrix}}{\begin{vmatrix} x_{1,i} & x_{1,j} \\ x_{2,i} & x_{2,j} \end{vmatrix}} > 0$$

for any of the compounds $A_{x_{1,s}} B_{x_{2,s}}$, $s = 1,...N, s \neq i, j$.(see [8])

This method generally gives a much better initial point for optimization, but at a higher computational cost. Indeed, even in 2D, the coplanarity check performed for each of the $(N(N-1))/2$ pairs of stoichiometric phases involves $(N-2)$ calculations of the determinants specified above. Since the numerator needs a total of 12 multiplications and 5 additions, while the denominator is calculated after 2 multiplications and 1 addition, the total complexity for the coplanarity check is of the order $O(N(N-1)(N-2)) \Rightarrow O(N^3)$ operations, where N is the number of points in the subdivision.

3. A new algorithm

Both of the aforementioned drawbacks have to do with the fact that Thermocalc does not possess the ability to recognize and utilize the geometric properties of Gibbs energy curves. The method of Chen et al described above takes into account system geometry, but limits its operation to function values, while it is the derivative information that seems to provide the best insight into the geometric structure of the object under study. Knowledge of the critical and inflection points of a system is a helpful tool in designing an efficient minimization algorithm. This is the key idea behind the new numerical scheme we are about to propose next, which overcomes the drawbacks

of the previously mentioned algorithms, at the same time giving comparable accuracy to the solution.

3.1 Description of the algorithm

Since in general the Gibbs energy is represented by a nonlinear functional, the task of finding precise locations of its critical points may be too arduous. It makes sense to either deal with numerical derivatives instead, or to consider a reasonable approximation to the given functional. Although we are going to follow the first approach when presenting the algorithms, it is worth noting that in the binary case it is possible to use a polynomial least-squares fitting. We will return to this point later in the discussion on the numerical characteristics of the algorithm.

3.1.1 Binary case

First, notice that the procedure of finding critical points becomes much more difficult with the decrease of curvature values. Hence it is desirable to deal with functions that are not flat to begin with. In practice, many complex systems exhibit this type of behavior, thus leading to numerous problems and possible failures of computational software. Being aware of this fact, we use the following idea. Since a linear transformation of the carrying axis does not change the relative positions of minima for energy curves and does not significantly change their absolute locations, we can tilt the axis and use the modified geometry to get an initial approximation for the optimization procedure that is carried out later for the original configuration. The transformation suitable for these purposes has the form: $y_{new}(x) = M(y(x) - (y_m(1) - y_m(0))x - y_m(0))$. The constants here are chosen to make sure that the curve having the minimal value at the right end ($y_m(x)$) approaches zero on both sides. A scaling constant M is introduced to increase the curvature (we use $M = 2$ in the examples below). In Figure 2 we display such a transformation for the Ca-Na system. The transformation described here is performed only once at the initial stage of the construction so it does not increase the scheme complexity.

The main component of the algorithm to be proposed for a binary phase diagram construction is the recursive procedure of finding positions of critical points. This procedure relies on the adaptive refinement strategy and uses first and second order derivatives information to detect possible miscibility gaps and identify local minima with a prescribed accuracy ε. The other user defined parameters include the maximum number of refinements and the total number of axis subdivisions at each step. All derivatives in the algorithms described below are computed numerically by some finite difference approximation schemes.

Function *minima* = *AdaptiveSearch*(a, b, *phase*, *iter*)
Global parameters: N - the number of axis subdivisions,
ε - tolerance, *Niter* - maximum number of allowed refinements
Input parameters: *a, b* – ends of the interval, *phase* – phase index, *iter* – iteration index
Output parameters: *minima* – positions of the minima for the energy of the *phase*
while *(iter<=Niter)*
(1) Sample N points $a = x_0 < x_1 < \ldots < x_N < x_{N+1} = b$.
(2) For ($iter == 1$) % finding concavity regions

(a) Calculate $G''^{(phase)}(x_j)$ for $j = 1,\ldots,N$.

(b) Locate inflection points by finding indices $\{s \,|\, 1 \le s \le N\}$ such that $G''^{(phase)}(x_s) \cdot G''^{(phase)}(x_{s+1}) < 0$.

(c) Identify interval(s) for refinement by counting inflection points.

If no inflection points found, put k=1, $a(1) = a$, $b(1) = b$, endif.

If one inflection point x_s found and $G''^{(phase)}(x_s) > 0$, put k=1, $a(1) = a$, $b(1) = x_s$, endif

If one inflection point x_s found and $G''^{(phase)}(x_s) < 0$, put k=1, $a(1) = x_s$, $b(1) = b$, endif

If two inflection points $x_{s\cdot}, x_{s_2}$ found, put k=2, $a(1)=a$, $b(1) = x_{s_1}$, $a(2)= x_{s_2}$, $b(2)=b$, endif

(d) Perform recursive search on each of the identified intervals $(a(j), b(j))$:

$minima(j) = AdaptiveSearch(a(j), b(j), phase, 2)$

(3) For $(iter > 1)$ % recursive search procedure

(a) Calculate $G'^{(phase)}(x_j)$, $j = 1,\ldots,N$.

(b) Find $s = \arg\min_{j=1,\ldots,N} G'^{(phase)}(x_j)$

(c) If ($G'^{(phase)}(x_s) < \varepsilon$) or $(iter == Niter)$ % met stopping criteria

$minima = x_s$, return *minima*

else %recursive refinement

For $\delta = (b-a)/(2N)$ do

$minima = AdaptiveSearch(x_s - \delta, x_s + \delta, phase, iter + 1)$

end if

end while

return *minima*

In simple words, the method attempts to find approximate locations of all possible minima of the energy functional and take them as starting points for the subsequent minimization procedure. Note that since there are at least one and at most two minima for any unordered phase under consideration, the algorithm will refine the grid as long as it cannot detect any of them. As with any discrete numerical approximation, there may still be a chance of missing a minimum. If after a sufficient number of refinements, critical points are still not found, the algorithm resorts to taking points with the lowest first derivative values as starting points for the later optimization. However, such situations are very rare in practice and are unlikely to cause troubles for most energy functionals due to the adaptive refinement strategy described above. The detection of a miscibility gap is straightforward due to availability of second derivative information.

It has to be noted that, in the 2D case, it is possible to avoid explicit calculations of the derivatives by making use of polynomial (in this case, quadratic) approximation. This approach has the same order of complexity as the method described above, but loses effectiveness when the dimension of the problem is increased. For the sake of generality we will use direct differentiation in all algorithms presented in this paper.

We now are ready to present the algorithm of calculating the stable binary 2-phase equilibria. First let us introduce a couple of auxiliary structures.

The matrix $A(1:ind,1:4)$ is used to record stability regions after the first sweep. Its first and second columns represent the coordinates of the left and right ends of the stability regions respectively, while the indices of the phases having lowest energy at those ends are recorded in the third and fourth columns.

Matrix C contains all the points that are obtained as suitable candidates for starting positions after the second sweep. The coordinate(s) of these points is recorded in the first (or first two in the ternary case) column(s) and the index of the corresponding phase goes into the last column of this matrix. The operation of adding a new row to this matrix is denoted everywhere in the text by the arrow sign "$\leftarrow$". With these notations, the stability region calculation procedure is given as follows.

Function $[A, ind] = StabilityRegions(a,b,N,K)$

Input parameters: *a,b – ends of the interval. N – number of grid points*
K - number of phases present

Output parameters: A – array recording stability regions information
ind – total number of stability regions in array A

1) Subdivide domain $V = [a,b]$ into $N-1$ subdomains $V_j = [x_j, x_{j+1}]$,
$a = x_1 < x_2 < \ldots < x_N = b$

2) Initialize $ind = 1, A(1,1) = a, A(1,2) = a, A(1,3) = 1, A(1,4) = 1;$
For $j = 2,..., N-1$ do
For $i = 1,\ldots K$ do

(a) Calculate $G^{(i)}(x_j)$

(b) Find the phase with lowest energy among calculated energy values at x_j and x_{j+1}:
$\sigma_{j,left} = \{s \mid G^{(s)}(x_j) < G^{(i)}(x_j), \forall i < s\}$; $\sigma_{j,right} = \{s \mid G^{(s)}(x_{j+1}) < G^{(i)}(x_{j+1}), \forall i < s\}$

(c) If $\sigma_{j,left} = \sigma_{j,right}$, $A(ind,2) = x_j$, $A(ind,4) = \sigma_{j,right}$ % extend old stability region
else % start new stability region
$A(ind,2) = x_j$, $A(ind,4) = \sigma_{j,right}$
$ind = ind + 1$; $A(ind,1) = x_j$, $A(ind,3) = \sigma_{j,right}$
end if
end for;
end for
return [*A*, ind]

The algorithm for constructing binary phase diagram with K phases can be summarized as follows:

Algorithm 1. Binary diagram construction

1) Fix N – the number of grid points in major axis subdivision,
ε - tolerance, *Niter* - maximum number of allowed refinements
2) Do $[A, ind] = StabilityRegions(0,1,2N,K)$ % Identify stability regions

3) Calculate starting points for optimization
For $i = 1,...,ind$, do
$phase_1 = A(i,3)$, $phase_2 = A(i,4)$
If $(phase_1 \neq phase_2)$ % add points at the boundaries of stability regions
$C \leftarrow (A(i,1), phase_1)$ and $C \leftarrow (A(i,2), phase_2)$;
else % find minima inside each stability region
$minima = AdaptiveSearch(A(i,1), A(i,2), phase_1, 1)$
$C \leftarrow (minima,\ phase_1)$
end if
end for
4) Perform coplanarity checks to get the convex hull of points in C
5) Carry out optimization for all remaining pairs of points, check result for consistency
6) Construct phase diagram using solution obtained in step 5.

Essentially, the method first detects the stability regions of the diagram, i.e. identifies which phase has the lowest energy in each of the intervals formed by the intersection points (see Figure 3). Then it proceeds to examine each of the intervals separately, identifying extrema and possibly other points (at most two per each region) that would serve as candidate ends of the common tangents between curves. As soon as all such points are found, a coplanarity check removes all points that could possibly appear inside the convex hull. At this stage, an exact (or more precisely, a good numerical) solution is obtained by solving the nonlinear optimization problem described earlier. To ensure the most reliable results, the solution is further checked for consistency with one additional coplanarity check.

Notice that, in general, it is not possible to provide a good initial guess by only considering critical points of the energy curves. An example of such a situation is shown in Figure 4. Although the minima of both parabolas can be easily detected by the adaptive scheme described above, only one of them provides a suitable initial guess for the optimization procedure. In this case it is necessary to pay attention to the appropriate endpoint of the stability region, as explained in the step 3(a) of the algorithm.

3.1.2 Ternary case

In two and higher dimensions, due to changes in topological properties comparing to the 1D case, our algorithm needs to be modified accordingly in order to keep the calculation efficient. The first issue is the difficulty of working with curvilinear boundaries. If we want to identify the stability regions like we did in the binary case, we are up for a complicated task of working with unordered sets of data with various possible intersections. Instead of following this approach, we first identify critical point locations for all phases and then perform the coplanarity checks. This reduces the overall complexity, but leaves the necessity of adding boundary points to the set of candidate starting positions.

Another observation that has to be made is that the derivative calculation can hardly be avoided in dimensions higher than two, hence the scheme relies on the numerical differentiation, which calls for a good meshing approach. Some of the possible sampling strategies will be discussed in

section 3.3. Here we only mention the importance of data ordering for the successful implementation of the general algorithm to be presented.

The recursive procedure of finding the critical points in the ternary case is as follows.

Function $minima = AdaptiveSearch2D(V, phase, iter)$
Global parameters: N - the number of axis subdivisions, ε - tolerance,
$Niter$ - maximum number of allowed refinements
Input parameters: V – given domain, $phase$ – phase index, $iter$ – iteration index
Output parameters: *minima* – positions of the minima for the energy of the *phase*
while *(iter<=Niter)*
(1) Sample N points $x_j = (x_j(1), x_j(2)), j = 1,..., N$ on V.
(2) For $(iter == 1)$ % finding concavity regions
(a) Calculate $G''^{(phase)}(x_j)$ for $j = 1,\ldots, N$
(b) Find regions of positive concavity by identifying the sets V_i such that $G''^{(phase)}(x) > 0$ for any x in V_i. If there is only one such set, put $V = V_1$
(c) Perform recursive search on each of the identified regions V_i:
$minima(i) = AdaptiveSearch2D(V_i, phase, 2)$)
(3) For $(iter > 1)$ % recursive search procedure
(a) Calculate $G'^{(phase)}(x_j)$, $j = 1,\ldots, N$
(b) Find $s = \arg\min_{j=1,\ldots,N} G'^{(phase)}(x_j)$
(c) If ($G'^{(phase)}(x_s) < \varepsilon$) or $(iter == Niter)$ % met stopping criteria
$minima = x_s$, return *minima*
else
For $\delta = diam(V)/(2\sqrt{N})$ and $V_s = [x_s(1) - \delta, x_s(1) + \delta] \times [x_s(2) - \delta, x_s(2) + \delta]$
do $minima = AdaptiveSearch2D(V_s, phase, iter + 1)$ %apply recursive refinement
end if
end while
return *minima*

Below we give the details of the algorithm for computing stable 2-phase equilibia in ternary systems with a total of K phases. ZPF stands for the Zero Phase Fraction method, traditionally used to trace phase boundaries (see for example [10]).

Algorithm 2. Ternary diagram construction

1) Fix original domain as $V = \{(x, y) \mid x + y \le 1\}$, N – the number of grid points in major axis subdivision, ε - tolerance, $Niter$ - maximum number of allowed refinements
2) For $phase = 1,..., K$ do
(a) $minima = AdaptiveSearch2D(V, phase, 1)$
$C \leftarrow$ *(minima, phase)*

(b) Sample N points *bdrypts* on the boundary of domain V,
$C \leftarrow$ (*bdrypts*, *phase*)
end
3) Perform coplanarity checks to get the convex hull of points in C
4) Carry out optimization for all remaining pairs of points, check result for consistency
5) Use ZPF to track the boundaries and complete the phase diagram construction.

Similar scheme can be constructed in higher dimensions. Notice that the algorithm is capable of predicting multiple minima using the second order derivative information, which makes it applicable even in the difficult multiphase miscibility gap situations.

The overall performance of this scheme is mostly influenced by the two major factors: the accuracy in the detection of the critical points and the effectiveness of the chosen sampling scheme. In the later sections, we discuss the theoretical and practical advantages of the new method, in comparison with other existing techniques.

3.2 Computational complexity estimate for the binary case

In this section we will compare complexity of our algorithm to other existing schemes from the point of view of required resources and computational workload. First, let h be the smallest mesh size required in order to identify points with the lowest energy on any one of stability regions with a given accuracy ε. By measuring the total number of grid points needed to reach this mesh size, we claim that, due to the adaptivity, our proposed scheme requires significantly less subdivisions than other comparable methods.

Indeed, suppose the number of levels required for the adaptive scheme to reach this mesh size is denoted as L. Since after a refinement stage, each interval is either subdivided into N subintervals (there are at most two such intervals) or left unchanged, the mesh size at each level is reduced by a factor of $1/N$ and the total number of intervals is increased by at most $2(N-1)$. Hence $h = 1/N^L$ or $L = \log_N 1/h$. It follows also that the total number of intervals needed to reach a mesh size h is $N_T = N + 2(N-1)\cdot L = N + (2(N-1)\ln 1/h)/\ln N$. Note that N is taken to be a constant independent of h (we use a fixed $N = 10$ in the numerical experiments). A comparable full uniform grid scheme (like the one in the Chen et al algorithm) should have approximately $N_T' = 1/h$ grid points to yield the same accuracy as the scheme proposed above. In other words, $N_T = O(\ln N_T')$, which implies a significant reduction in the number of axis subdivisions comparing to the uniform scheme.

Second, let us assess the amount of work required by each of the algorithms for finding a starting point for the two-phase equilibrium calculation prior to the optimization based on the same mesh size h. We again can claim an advantage of our scheme in comparison with the approach of Chen et al due to a significant reduction in the number of required coplanarity checks. Below is the step by step analysis of the computational complexity that verifies to our claim.

From the point of view of complexity, Algorithm 1 as given in section 3.1.1 can be divided into the following major stages:

i) $[A, ind] = StabilityRegions(0,1,2N,K)$ *– stability region calculation*

To compare energy values for all K phases at each grid point, we need $2N \cdot K$ operations, which is the total complexity for this stage. Notice that this estimate has no dependence on h.

ii) Starting points calculation

AdaptiveSearch procedure is performed on each of the *ind* stability regions where the total number of stability regions *ind* is a constant independent of. h. At the first sweep, we detect inflection points by calculating second derivatives at each grid point: a total of $3N$ function evaluations if a 3-point stencil is used in the derivative calculation. At the second sweep, N first derivatives are calculated on at most 2 subintervals, which adds up to a total of $4N$ function evaluations if a 2-point stencil is used in derivative calculation. If the critical point was detected with predefined accuracy ε after $L = (\ln 1/h)/\ln N$ adaptive grid refinements, the first derivative calculation had to be repeated $(\ln 1/h)/\ln N$ times. It follows that the overall complexity of this stage is given by $3N \cdot ind + 4N \cdot ind \cdot (\ln 1/h)/\ln N = O(\ln 1/h)$.

iii) *Coplanarity checks*

At the last stage of the Algorithm 1, two coplanarity checks are performed for all selected pairs. A coplanarity check for p selected points requires a calculation of a total of $0.5p(p-1)(2+12(p-2))$ operations (determinant calculations for each of the $0.5p(p-1)$ pairs). The total complexity of this stage is thus $P = p(p-1)(2+12(p-2))$. Since the total number of selected pairs is specific to the system configuration and does not depend on the refinements, the final stage does not raise the overall algorithm complexity in terms of $1/h$.

It follows from the above estimation that, for fixed parameter N without further refinement, our scheme has a total complexity of $O(\ln 1/h)$.

Likewise we can calculate the number of operations required for the Chen et al algorithm of comparable accuracy ($N_T' = 1/h$ is the total number of subdivisions required). As we have seen in Section 2.3, it can be roughly estimated as

$$K \cdot N_T' + 0.5 \cdot N_T'(N_T' - 1)(2 + 12(N_T' - 2)) = K/h + (1/(2h))(1/h - 1)(2 + 12(1/h - 2)) = O(1/h^3).$$

Here $K \cdot N_T'$ operations are spent on stability checks, while the coplanarity check for each of the pairs takes up the rest of the complexity. It is obvious that for small h our complexity $O(\ln 1/h)$ is significantly lower than the $O(1/h^3)$ complexity of the Chen et al algorithm.

As an illustration, let us fix $N = 10$, $ind = 2K$ (the maximum number of stability regions in 2d) and $p = 2 \cdot ind$. The graph in Figure 5 illustrates the behavior of calculated complexity estimates for the uniform Chen et al type scheme versus the new algorithm for the $K = 2$ (two phase) case. It is clear that for a mesh size smaller than a critical value ($h \approx 0.05$ for our example), the adaptive scheme proposed above outperforms the uniform grid algorithm, and its advantage becomes even more visible as the mesh size required to detect the lowest energy decreases.

3.3 Generalization to higher dimensions and sampling schemes

In light of the results derived in the previous section, the new adaptive technique possesses an advantage over other schemes in that it needs significantly less points and operations to achieve

the required accuracy. Still, such a scheme also becomes computationally demanding when the dimension of the system starts to grow. However, there are some ideas that can be entertained in order to reduce the complexity of the method in higher dimensions.

For instance, the Hammersley or Halton quasi-random sequences can help reduce the complexity while allowing for critical point detection with the same accuracy in dimensions up to $s = 8$ (degradation and correlation can occur in higher dimensions). These sequences are low-discrepancy point sets in a sense that the discrepancy (deviation from the uniform distribution) of an N point sequence in s-dimensional case satisfies $D_N^* = O(N^{-1}(\log N)^{s-1})$ (see [11]). The Halton sequence is superior to that of the Hammersley in that it builds upon the previous sets as the number of points increases. We can hope that the use of these sequences will allow to detect critical points with the accuracy similar to a uniform approach while reducing the overall computational cost.

The difficulty of using quasi-random approach with any adaptive refinement strategy is the need for a careful point ordering. One should also think about the possible loss of accuracy in derivative calculations done on such a mesh. The last obstacle can be avoided by introducing a finer regular grid around each point for finite difference calculations. For sufficiently small grid size h, the error introduced by such types of calculation is at least of the order of $O(h)$ and often $O(h^2)$ for the first and second order derivatives depending on the difference scheme [12]. The overall complexity will not be affected if a fixed number of auxiliary points is used for all grid points. To overcome the ordering difficulty, one can reorder the points independently on each subset after each refinement.

The most attractive property of quasi-random sequences is that they dramatically reduce the error bounds for integration [11]. This gives us reason to believe that the quasi-random construction can potentially work very well in our case, especially if the derivative information is used adaptively in the process of mesh construction.

4. Results for binary and ternary systems

All examples given below rely on the following form of the Gibbs energy functional, where the excess Gibbs energy is expressed in the form of Redlich-Kister polynomial:

$$G_m^\Phi = \sum_i x_i^0 G_i^\Phi + RT \sum_i x_i \ln x_i + {}^{xs}G_m^\Phi$$

$${}^{xs}G_m^\Phi = \sum_{j>i} x_i x_j \sum_{k=0}^{n} L_{i,j}^\Phi (x_i - x_j)^k$$

Performance estimates have been done with the Matlab 6.5 implementation of the algorithm on a Pentium 4 2.4Ghz machine with 512MB RAM Figures 6(a) through 8(a) show phase diagrams computed using this implementation of the new method, while Figures 6(a) through 7(b) are reproduced from [13] and are created using Thermocalc software with the aid of a priori knowledge of the system.

4.1 Binary examples

First, we consider a Ca-Li-Na system. Figure 6(a) represents its binary Li-Na projection for $T = 900K$, where the miscibility gap occurs. As shown in Figure 1, the phase diagram calculation done by the Thermocalc software independently produces unacceptable results when the miscibility gap is not specified manually. The method described above detects the existing liquid and bcc miscibility gaps and correctly predicts the corresponding phase diagram with absolutely no additional input from the user.

In the Matlab 6.5 implementation, the complete Li-Na phase diagram construction with the temperature step size $dt = 1^o K$ took about 232 sec. Another example is the Ca-Na projection calculated at $T = 900K$. Here all three phases (liquid, bcc, fcc) form stable equilibria at different temperatures and a liquid miscibility gap occurs at temperatures higher than $1000^o K$. It took 261 sec to produce the complete diagram which is given in Figure 7(a) below.

4.2 Ternary examples

Figure 8 shows the Gibbs energy of the ternary Ca-Li-Na system at $T = 900K$. The straight line indicates the common tangent found by the new algorithm for the miscibility gap, which remained undetected during unassisted Thermocalc run producing incorrect diagram in Figure 1(b). The outline of the procedure used to compute ternary diagrams is as follows. The preprocessing module was designed that is capable of handling arbitrary ternary systems from given database specifications and can be integrated directly into the Thermocalc. Steps 1 and 2 of the Algorithm 2 discussed in section 3.1.2 are performed in this preprocessing module prior to the optimization. Results of the preprocessing calculation are then automatically recorded in the corresponding macro file that can be further fed into Thermocalc to produce the complete diagram as the one shown in Figure 1(a). The timing overhead of the preprocessing routine did not exceed 5 sec for any of the above fixed temperature calculations.

5. Conclusions

In this paper, we propose a new scheme to optimize the phase diagram construction algorithm adopted in Thermocalc. The new algorithm possesses advantages over existing methods in terms of the convergence speed, the computational complexity and the robustness. It can be used to automate the calculation of phase equilibria in complicated systems. Numerical results for binary and ternary systems show good agreement of automatic calculations with prior results.

As discussed earlier for the higher space dimensions, the new approach carries an increased computational load, so a tradeoff must be made between the accuracy of the solution and the complexity of the scheme. Possible higher dimensional solutions including better sampling techniques discussed above are the main focus of our current research and will be discussed in future publications.

6. Acknowledgements

The authors are very grateful to the referee for offering valuable suggestions that greatly improved the overall presentation of the results in this paper. We also wish to thank Shengjun

Zhang for useful discussions during the work on this paper and acknowledge the support of this work by the National Science Foundation through the grant DMR-0205232.

7. References

[1] J. Nocedal, S. Wright, Numerical Optimization, Springer-Verlag, 1999

[2] Bo Jansson, A General Method for Calculating Phase Equilibria under Different Types of Conditions, TRITA-MAC-0233, 1984.

[3] Mats Hillert, A discussion of methods of calculating phase diagrams, Bulletin of Alloy Phase Diagrams, 2, 265-268.

[4] J. O. Andersson, T. Helander, L. H. Hoglund, P. F. Shi and B. Sundman, THERMO-CALC & DICTRA, computational tools for materials science, CALPHAD, Vol.26, 2002, 273-312.

[5] H. L. Lukas, J. Weiss, E-Th. Henig, Strategies for the calculation of phase diagrams, CALPHAD, 6 (1982), 229-251

[6] J.A.D. Connoly, D. M. Kerrick, An algorithm and computer program for calculating composition diagrams, CALPHAD, 11 (1987) 1-54

[7] Samuel A. Safran, Statistical Thermodynamics of Surfaces, Interfaces and Membranes, Addison-Wesley, 1994

[8] S.-L. Chen, K.-C. Chou and Y.A. Chang, CALPHAD, 17 (1993) 237-250.

[9] S.-L. Chen, K.-C. Chou and Y.A. Chang, CALPHAD, 17 (1993) 287-302.

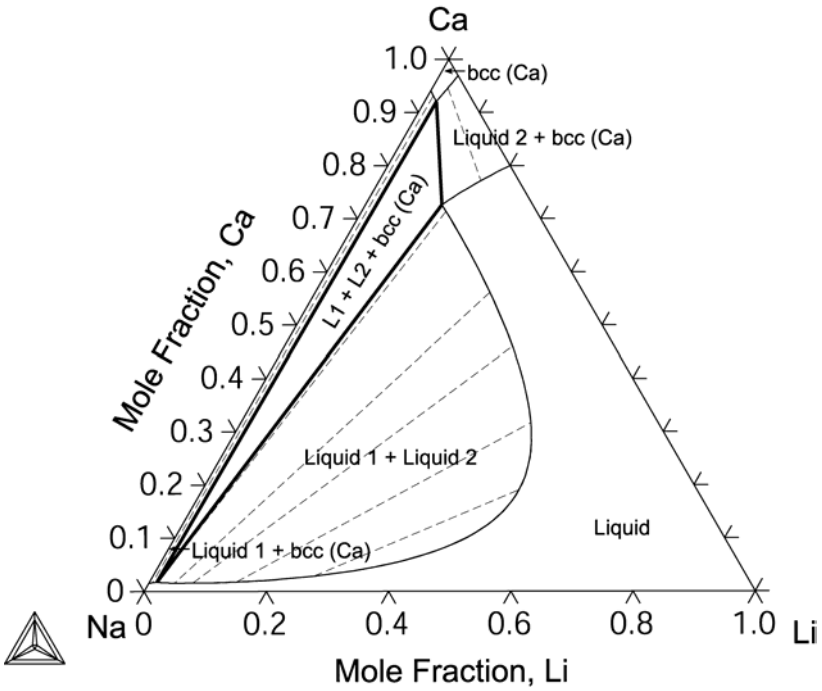
[10] S.-L. Chen,S. Daniel, F. Zhang, Y. A. Chang, X.-Y. Yan, F.-Y. Xie, R. Schmid-Fetzer, W. A. Oates,The Pandat Software Package and its Applications, CALPHAD, 26 (2002) 175-188

[11] Harald Niederreiter, Random Number Generation and Quasi-Monte Carlo Methods, CBMS-NSF regional conference series in applied mathematics, 1992

[12] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained optimization and nonlinear equations, Prentice-Hall, 1983

[13] S. J. Zhang, D. W. Shin and Z. K. Liu, Thermodynamic modeling of the Ca-Li-Na system, CALPHAD, Vol.27, 2003, 235-241.

(a)



(b)

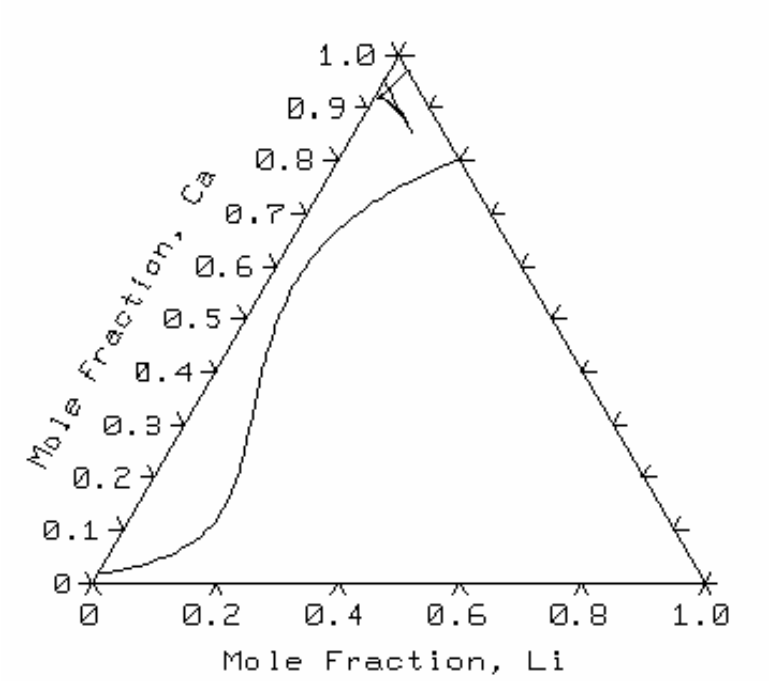


Figure 1: (a) Correct Ca-Li-Na phase diagram at T=900K produced by specifying the `set_miscibility_gap` option; (b) diagram produced with no `set_miscibility_gap` option

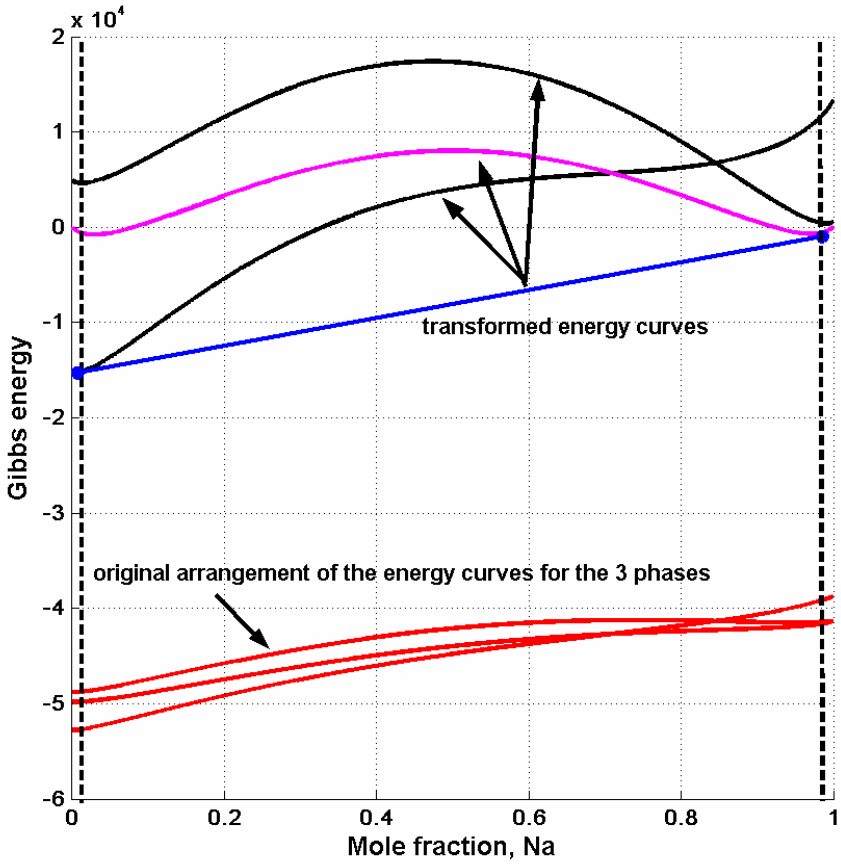


Figure 2: Linear axis transformation performed at first stage of the algorithm.

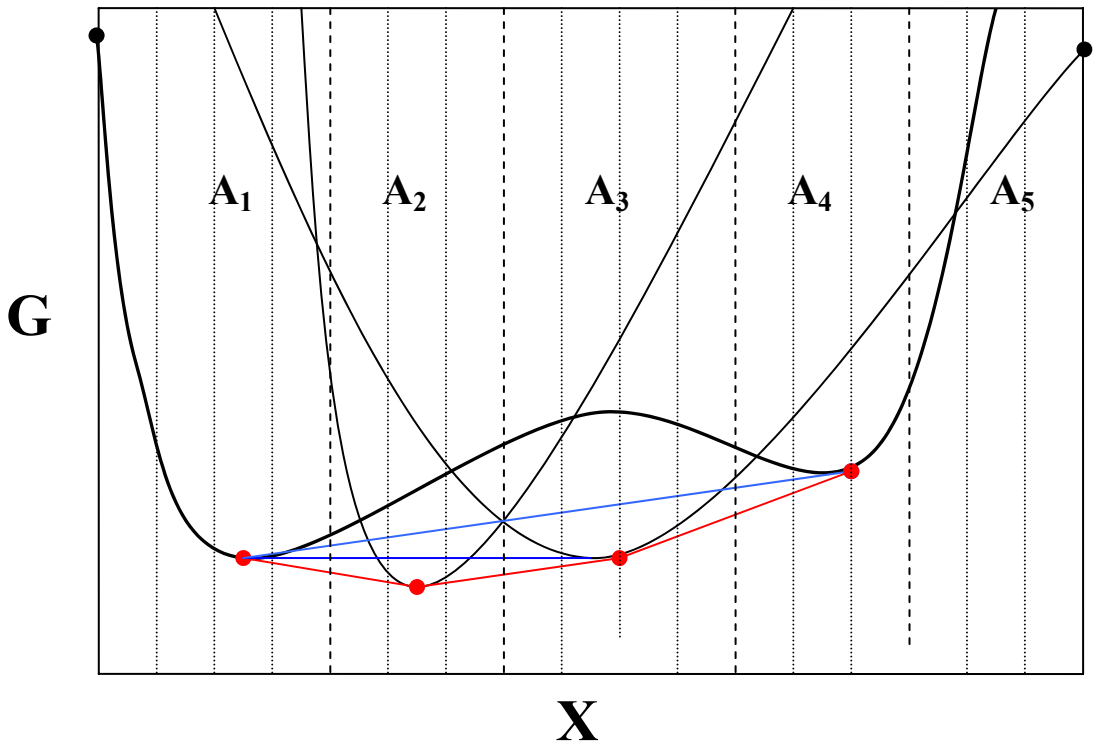


Figure 3: Stability regions for a binary 4-phase system and corresponding starting values for the optimization procedure .

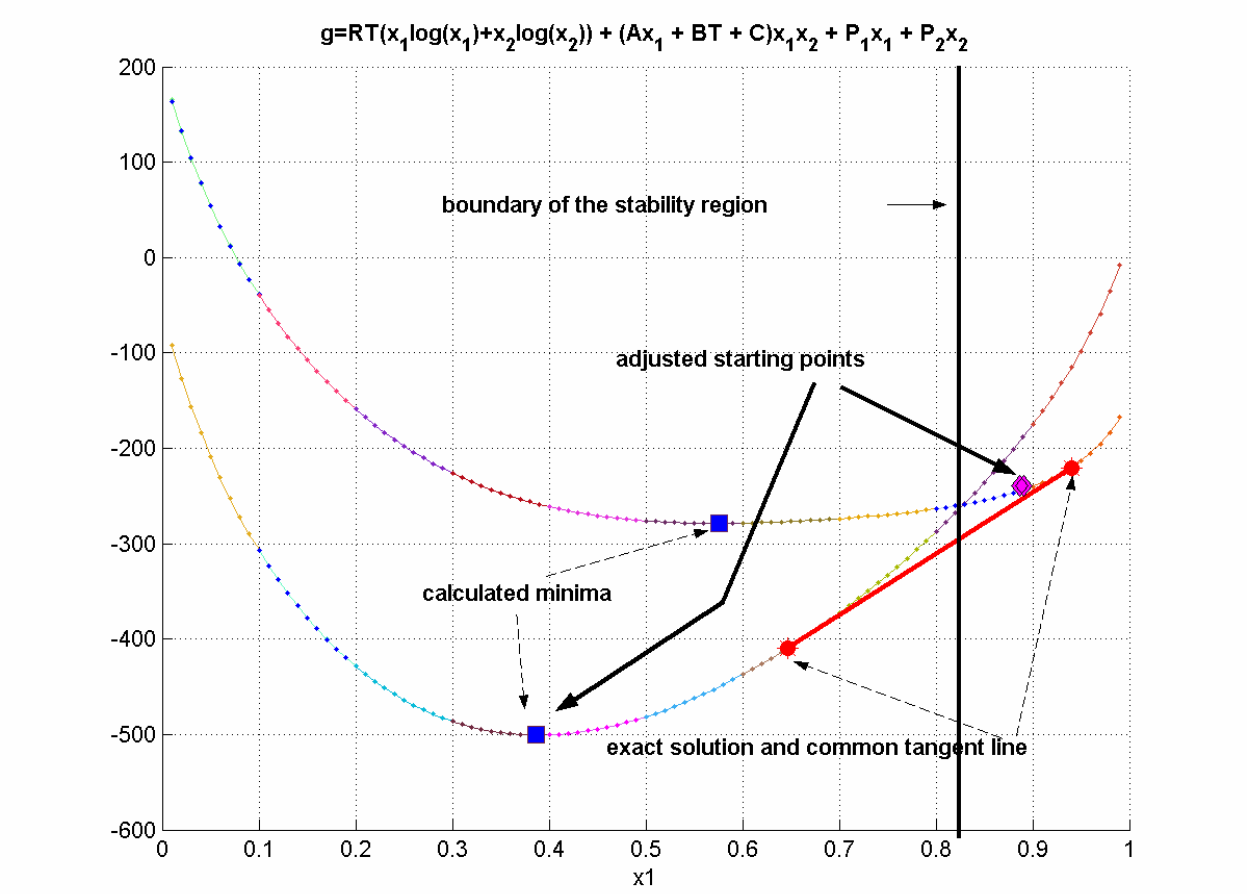


Figure 4. One of possible arrangements of critical points and corresponding starting points

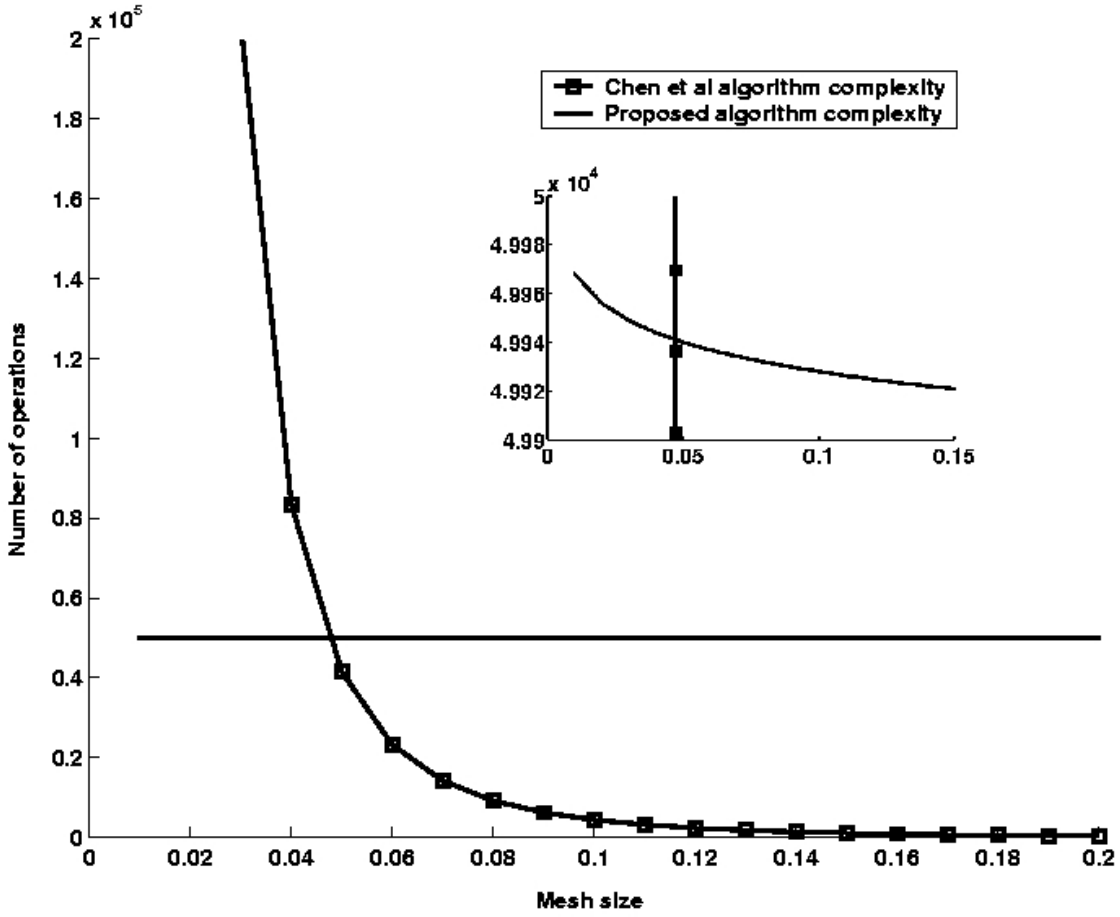
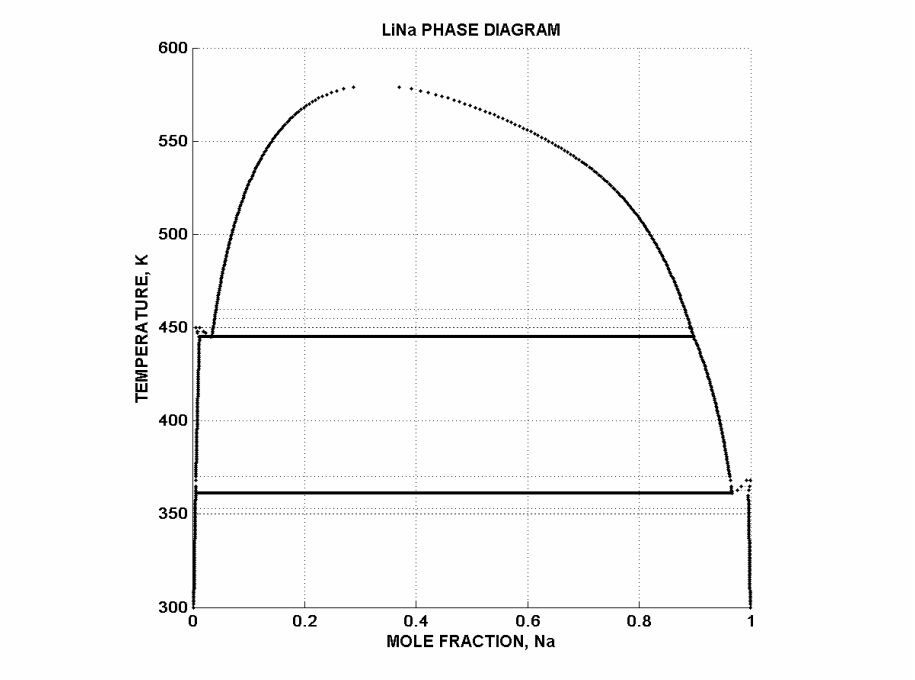


Figure 5. Complexity estimates for the proposed scheme compared to the Chen et al

(a)



(b)

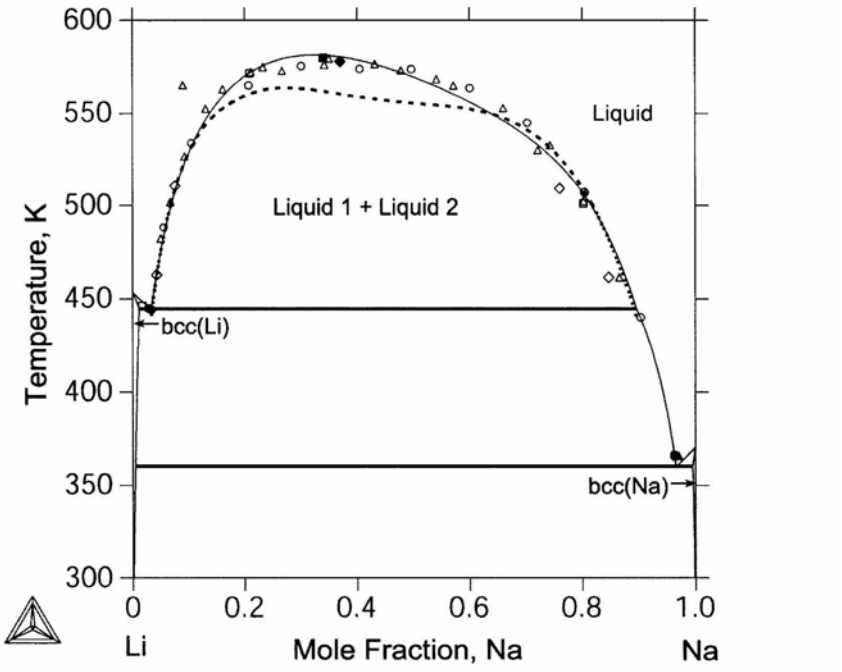
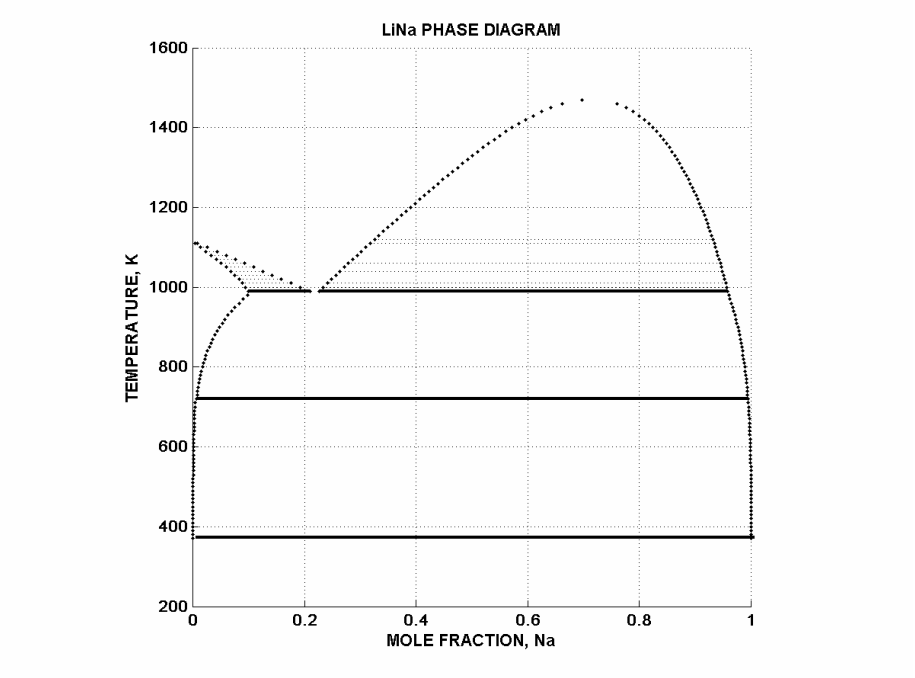


Figure 6: Li-Na phase diagrams calculated by the new (a) and Thermocalc (b) methods.

(a)



(b)

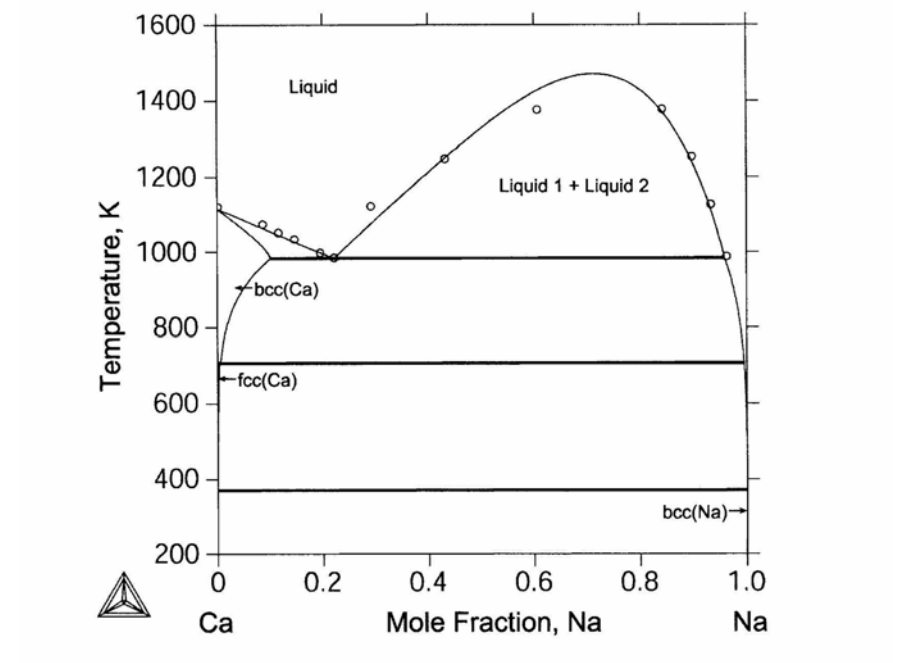


Figure 7. Ca-Na phase diagrams calculated by the new (a) and Thermocalc (b) methods.

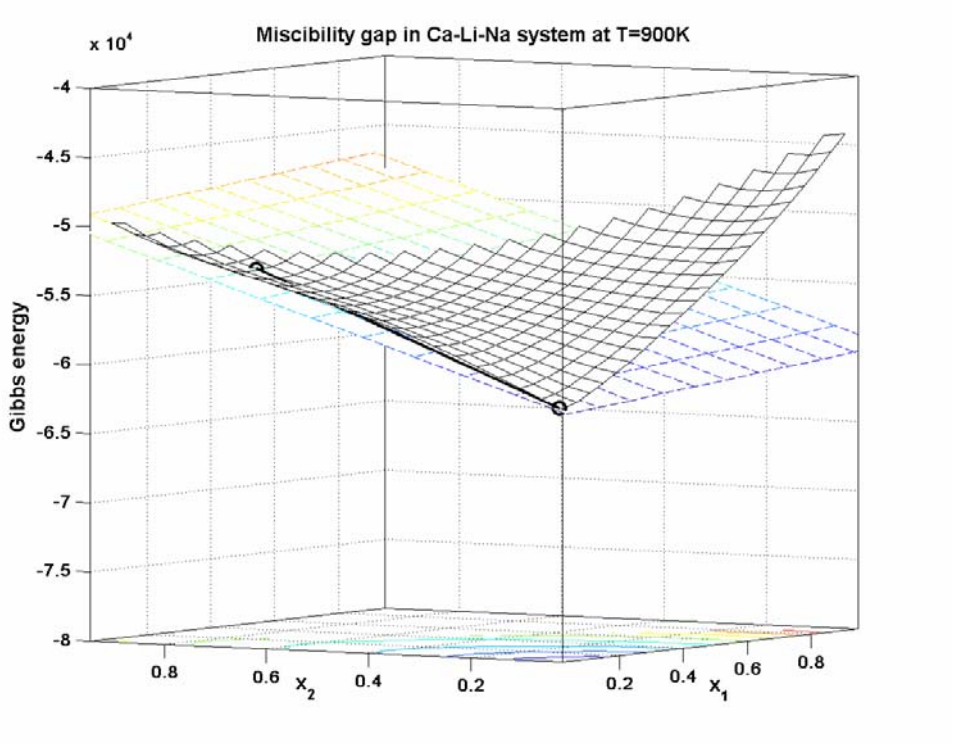


Figure 8. Gibbs energy for the Ca-Li-Na system showing miscibility gap.