Documentation of the implementation of an algorithm to calculate thermodynamic equilibria in multi-component systems for flexible sets of conditions (and some related topics).

Bo Sundman

Gif sur Yvette, France

Version 7.0 August 17, 2022

Most of the text is the same as in version 3, it is mainly the section on the software documentation that is updated and some references.

Abstract

Thermodynamics is a central part in materials science. Thermodynamic models provide a unique way to combine experimental data and results from first-principles calculations in databases. These databases are essential to provide values of many different thermodynamic properties for simulating materials processes and for predicting their final properties.

Here, a well established algorithm to calculate thermodynamic equilibria for multi-component systems is explained in detail. It can handle equilibria using different kinds of conditions and with many non-ideal solution phases modeled in different ways. In particular, we describe how to handle phases with variable amount of atoms and how to handle different types of conditions and changes of the set of stable phases during iterations.

The algorithm can also be used to calculate properties for a single phase outside the equilibrium state as required for the simulation of phase transitions.

This is part of the documentation of the OpenCalphad (OC) initiative. There is also documentation available for the model package [GTP] and there is on-going work to develop and document the step/map/plot procedures. There is also development of an OC Application Software Interface called OCASI. Finally there is a manual to the command oriented user interface to OC [ochelp] and descriptions of the macro examples and the beginnings of an assessment documentation.

The current release of the OC software and its documentation is available at [opencalphad] and the latest updates (under development) at [github].

Contents

1 Introduction

The algorithm presented here was derived and explained in slightly different forms by Gunnar Eriksson [73Eri], Mats Hillert [81Hil], Bo Jansson's thesis [84Jan] and Leo Lukas [82Luk]. These papers give the theoretical background of the algorithm but they are very dense and difficult to understand. In a recent paper the algorithm and its implementation is explained in more detail by Bo Sundman et al. [15Sun2]. Some corrections and extensions compared to the published version are included in this documentation. In Fig. 1 from a recent paper describing also the calculation of diagrams, the basic parts of the minimization algorithm is shown.

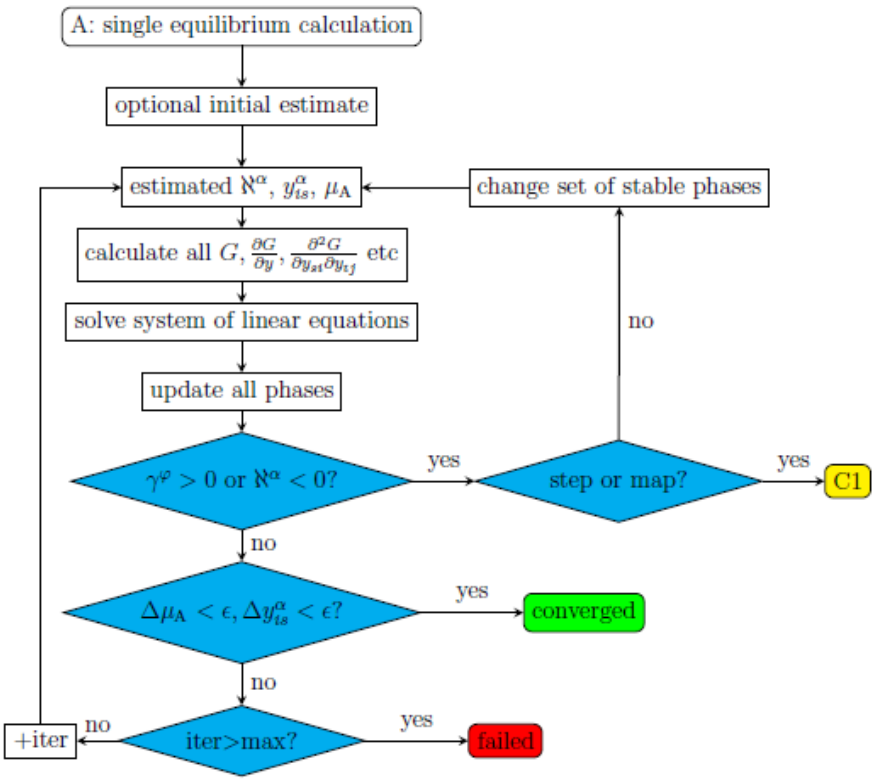


Figure 1: The basic parts of the minimization algorithm. The exit to algorithm C1 for step/map calculations is described in [21Sun]

The implementation is done as part of the OpenCalphad initiative [15Sun1] to provide a free software for thermodynamic calculations. This software will provide a useful link between experimental work, first-principle calculations and applications such as simulations of phase transformations and microstructure evolutions using phase field methods.

2 The thermodynamic model

Phases in a thermodynamic system can be very different, examples being gas, liquid, amorphous and many different crystalline forms. The Gibbs energy, G, for any system is given at equilibrium by

$$G = \sum_{\alpha} \aleph^{\alpha} G_M^{\alpha}(T, P, Y) \tag{1}$$

where $\aleph^{\alpha}$ is the number of moles of formula units of the phase α and G_M^{α} is the Gibbs energy per mole formula unit of α. Each phase can be modeled differently but its Gibbs energy depend on the temperature, pressure and its constitution, here denoted by T, P and Y. A summary of currently available models is given in the book by Lukas et al[07Luk] but all of them are not (yet) implemented in OC.

Note that the definition of the molar Gibbs energy used in eq. 1 is different from the normal

$$G_m = \frac{G}{N} \tag{2}$$

where N is the total number of moles of components. But in this paper G_M, with a capital M as subscript will always be used according to the definition in eq. 1, using the formula unit explained in the next section.

The equilibrium at constant T, P and overall composition will occur at a minimum in the Gibbs energy. For other conditions such as volume and chemical potential, we can add Lagrangian constraints to obtain the appropriate function to minimize. Internal constraints, such as the sum of constituent fractions on each sublattice, can also be handled by Lagrangian multipliers.

Each phase can be described with a different model but the explanations in this paper will mainly concern phases modeled with the compound energy formalism (CEF) [81Sun, 01Hil]. CEF includes models for ideal gases, substitutional regular solutions, interstitial solutions, sublattice models, charged constituents etc. For the ionic liquid model [85Hil] some modifications of the general algorithm is needed as explained in section 5.13.

2.1 The formula unit of a phase

For a phase modeled with CEF, we can have several sublattices. The multiplicity of each sublattice is denoted as a_s, where s is the sublattice. For example the σ phase has 5 sublattices with a_s equal to 2, 4, 8, 8 and 8 respectively. The ratio among the multiplicity of different sublattices can be denoted by integer numbers or fractions but the recommendation is to use the smallest integer values. Thus a Laves phase should be modeled $(A,B)_2(A,B)_1$ rather than $(A,B)_{0.666667}(A,B)_{0.333333}$ as the number of decimal digits may have numerical effects on the mass balance as well as the Gibbs energy.

The constituent fraction for a phase α is denoted y_i^{α}. For a phase with sublattices there will also be a second index, s, y_{is}^{α} indicating the sublattice, as the same species i may be a constituent of several sublattices. When there is no possible confusion the phase and sublattice superscripts are often omitted and a summation over i will mean for all constituents in all sublattices.

For ease of understanding we will from now on use indexes A, B etc. to denote components or elements and indexes i, j etc. to denote constituents of a phase. In many cases a constituent of a phase is an element but it can also be a molecule or an ion. The meaning of the term component is according the the Gibbs phase rule but a component will often be an element.

The number of moles of a component A per mole formula unit of the phase, $M_{\rm A}^{\alpha}$ is calculated as

$$M_{\rm A}^{\alpha} = \sum_s a_s^{\alpha} \sum_i b_{{\rm A}i} y_{is}^{\alpha} \tag{3}$$

where $b_{{\rm A}i}$ is the stoichiometric factor of component A in constituent i and y_{is}^{α} is the fraction of constituent i in sublattice s of phase α, whereas a_s has already been introduced. The sum of constituent fractions on each sublattice is unity.

The total number of moles of components in a formula unit of the phase is thus:

$$M^{\alpha} = \sum_{\rm A} M_{\rm A}^{\alpha} \tag{4}$$

and the mole fraction of component A in phase α is

$$x_{\rm A}^{\alpha} = \frac{M_{\rm A}^{\alpha}}{M^{\alpha}} \tag{5}$$

The total number of moles of component A, $N_{\rm A}$, in a system with several phases is

$$N_{\rm A} = \sum_{\alpha} \aleph^{\alpha} M_{\rm A}^{\alpha} \tag{6}$$

where $\aleph^{\alpha}$ is the number of moles formula unit of the phase α. The total number of moles of components, N, in the system is

$$N = \sum_{\rm A} N_{\rm A} = \sum_{\rm A} \sum_{\alpha} \aleph^{\alpha} M_{\rm A}^{\alpha} \tag{7}$$

and the mole fraction of A in the whole system is

$$x_{\rm A} = \frac{N_{\rm A}}{N} \tag{8}$$

We have to be careful to distinguish between $N, \aleph, M, x, y$ and other composition related variables.

2.2 Differentials

The differential of $M_{\rm A}^{\alpha}$ is

$$dM_{\rm A}^{\alpha} = \sum_s \sum_i \left(\frac{\partial M_{\rm A}^{\alpha}}{\partial y_{is}^{\alpha}} \right)_{y_{j \neq i}} dy_{is}^{\alpha} \tag{9}$$

where each partial derivative of $M_{\rm A}^{\alpha}$ is

$$\left(\frac{\partial M_{\rm A}^{\alpha}}{\partial y_{is}} \right)_{y_{j \neq i}} = a_s^{\alpha} b_{{\rm A}i} \tag{10}$$

Most of these derivatives will be zero as M_A often depends on a single or just a few y_{is}. But we can have molecules with several atoms as constituents and also vacancies in a sublattice and the number of moles of components per formula unit of the phase is often not fixed.

For the ionic liquid model the situation is different as the number of sites for cations and anions depends on the average charge on the other sublattice, as will be discussed in the following. It does not however affect the general procedure for the minimization of the Gibbs energy described here.

2.3 Examples

Understanding the meaning of formula unit is essential to the algorithm so a few examples will be given.

2.3.1 A gas phase with H and O

In a gas with the molecules (H_2, O_2, H_2O) the moles formula unit of the components H and O and the total number of moles in a formula unit, M, are:

$$\begin{aligned} M_H &= 2y_{H_2} + 2y_{H_2O} \\ M_O &= 2y_{O_2} + y_{H_2O} \\ M &= 2y_{H_2} + 2y_{O_2} + 3y_{H_2O} \end{aligned}$$

The number of moles of atoms per formula unit of this gas can thus vary between 2 and 3. The mole fractions are:

$$\begin{aligned} x_H = \frac{M_H}{M} &= \frac{2y_{H_2} + 2y_{H_2O}}{2y_{H_2} + 2y_{O_2} + 3y_{H_2O}} \\ x_O = \frac{M_O}{M} &= \frac{2y_{O_2} + y_{H_2O}}{2y_{H_2} + 2y_{O_2} + 3y_{H_2O}} \end{aligned}$$

2.3.2 A crystalline phase with substitutional and interstitial constituents

An interstitial solution of C and N in the bcc phase in a steel with Cr is modeled using the CEF as (Cr, Fe)$_1$(C, N, Va)$_3$, where Va denotes a vacancy or vacant site. The mole formula units of the elements are:

$$\begin{aligned} M_C &= 3y_{C,2} \\ M_{Cr} &= y_{Cr,1} \\ M_{Fe} &= y_{Fe,1} \\ M_N &= 3y_{N,2} \\ M &= 1 + 3y_{C,2} + 3y_{N,2} \end{aligned}$$

We use a comma between the constituent and the sublattice when they are explicit. If there are only two sublattices one frequently use one or two primes, y'_{Cr} and y''_{C} to indicate the sublattice but with more than two sublattices that become cumbersome. The number of moles of atoms per formula unit of the bcc phase can thus vary between 1 and 4. The mole fractions are:

$$\begin{aligned} x_{\mathrm{C}} &= \frac{3y_{\mathrm{C},2}}{1+3y_{\mathrm{C},2}+3y_{\mathrm{N},2}} \\ x_{\mathrm{Cr}} &= \frac{y_{\mathrm{Cr},1}}{1+3y_{\mathrm{C},2}+3y_{\mathrm{N},2}} \\ x_{\mathrm{Fe}} &= \frac{y_{\mathrm{Fe},1}}{1+3y_{\mathrm{C},2}+3y_{\mathrm{N},2}} \\ x_{\mathrm{N}} &= \frac{3y_{\mathrm{N},2}}{1+3y_{\mathrm{C},2}+3y_{\mathrm{N},2}} \end{aligned}$$

2.3.3 A crystalline phase with long range ordering

Finally for a σ phase in the Cr, Fe, Mo and Ni system modelled with only 3 sublattices and with restricted solubilities, (Cr, Fe, Ni)$_{10}$(Cr, Mo)$_4$(Cr, Fe, Mo, Ni)$_{16}$, the moles formula units are

$$\begin{aligned} M_{\mathrm{Cr}} &= 10y_{\mathrm{Cr},1}+4y_{\mathrm{Cr},2}+16y_{\mathrm{Cr},3} \\ M_{\mathrm{Fe}} &= 10y_{\mathrm{Fe},1}+16y_{\mathrm{Fe},3} \\ M_{\mathrm{Mo}} &= 4y_{\mathrm{Mo},2}+16y_{\mathrm{Mo},3} \\ M_{\mathrm{Ni}} &= 10y_{\mathrm{Ni},1}+16y_{\mathrm{Ni},3} \\ M &= 30 \end{aligned}$$

The number of moles per formula unit is constant and equal to 30. The mole fractions are

$$\begin{aligned} x_{\mathrm{Cr}} &= \frac{10y_{\mathrm{Cr},1}+4y_{\mathrm{Cr},2}+16y_{\mathrm{Cr},3}}{30} \\ x_{\mathrm{Fe}} &= \frac{10y_{\mathrm{Fe},1}+16y_{\mathrm{Fe},3}}{30} \\ x_{\mathrm{Mo}} &= \frac{4y_{\mathrm{Mo},2}+16y_{\mathrm{Mo},3}}{30} \\ x_{\mathrm{Ni}} &= \frac{10y_{\mathrm{Ni},1}+16y_{\mathrm{Ni},3}}{30} \end{aligned}$$

The reason for this somewhat lengthy explanation is that it is very common to make mistakes, or be uncertain using different kinds of variables for the amount of material in a system or phase.

2.4 End-members in the Compound Energy Formalism

An important concept when modeling with the CEF is the *end-member*, i.e. for a crystalline phase is a hypothetical compound in which each sublattice is occupied by one and only one constituent. A phase can consist of a single end-member. For the gas phase each molecule is an end-member. In the bcc phase example above we have 6 end-members which can be denoted: (Cr:C), (Cr:N), (Cr:Va), (Fe:C), (Fe:N) and (Fe:Va). In the σ phase example there are 16 end-members, for example (Fe:Cr:Cr), (Fe:Cr:Fe), (Fe:Cr:Mo), (Fe:Cr:Ni) etc. Note that in most cases end-members represent compounds with several elements.

3 The Gibbs energy

The Gibbs energy, G is an extensive property and can be expressed in many different ways. One well known formula relates the Gibbs energy to the chemical potentials, $\mu_{\rm A}$, and the number of moles, $N_{\rm A}$ of the components:

$$G = \sum_{\rm A} N_{\rm A}\mu_{\rm A} \tag{11}$$

The definition of the chemical potentials is

$$\mu_{\rm A} = \left(\frac{\partial G}{\partial N_{\rm A}}\right)_{T,P,N_{\rm B\neq A}} \tag{12}$$

If a phase has a fixed composition it has only a Gibbs energy and we cannot calculate the individual chemical potentials of the components for this phase alone. But if we can vary the fraction of constituents in one or more sublattices it is possible to calculate some relations between chemical potentials as it will be discussed below for eq. 21.

We have already introduced a different definition of the molar Gibbs energy for each phase which is related to the formula unit of the phase as defined by the structure of the phase. Using eq. 3, the molar Gibbs energy for a formula unit, G^{α}_{M}, and for one mole of components, G^{α}_{m}, for a phase α are equal to:

$$G^{\alpha}_{M} = \sum_{\rm A} M^{\alpha}_{\rm A}\mu_{\rm A} \tag{13}$$

$$G^{\alpha}_{m} = \sum_{\rm A} x^{\alpha}_{\rm A}\mu_{\rm A} \tag{14}$$

It is thus important to know what kind of "molar" quantity we are using.

3.1 The differential of the Gibbs energy

The differential of the Gibbs energy is

$$dG = -SdT + VdP + \sum_{\rm A} \mu_{\rm A} dN_{\rm A} \tag{15}$$

and at equilibrium in a closed system $dG = 0$. If we have several stable phases

$$dG = \sum_{\alpha}(\aleph^{\alpha} dG_M^{\alpha} + G_M^{\alpha} d\aleph^{\alpha}) \quad (16)$$

where dG_M^{α} for each phase can, using the molar Gibbs energy per formula unit, be expressed as differences of the independent variables T, P and $M_{\rm A}$ or the dependent y_{is}^{α}:

$$\begin{aligned} dG_M^{\alpha} &= -S_M^{\alpha} dT + V_M^{\alpha} dP + \sum_{\rm A} \mu_{\rm A} dM_{\rm A}^{\alpha} \\ &= -S_M^{\alpha} dT + V_M^{\alpha} dP + \sum_{\rm A} \mu_{\rm A} \sum_s \sum_i \left(\frac{\partial M_{\rm A}^{\alpha}}{\partial y_{is}^{\alpha}} \right)_{T,P,y_{j\neq i}} dy_{is}^{\alpha} \end{aligned} \quad (17)$$

3.2 The partial Gibbs energy for an end-member

For a phase with sublattices it is not always possible to calculate directly the chemical potential for each component, but we can always calculate the partial Gibbs energies of the end-members, I, of the model. An end-member specifies one constituent in each sublattice:

$$G_I = G_M + \sum_s \left(\frac{\partial G_M}{\partial y_{is\in I}} \right)_{T,P,y_{j\neq i}} - \sum_s \sum_j y_{js} \left(\frac{\partial G_M}{\partial y_{js}} \right)_{T,P,y_{k\neq j}} \quad (18)$$

where the first summation runs over all sublattices s and i is the constituent in the sublattice s for the end-member I. The second double summation is for all constituents j. This equation is derived in [81Sun]. For a phase with fixed composition $G_I = G_M$.

In the example for the bcc interstitial solution (Fe, Cr)$_1$(C, N, Va)$_3$ above it is not possible to express directly the partial Gibbs energy for C in the model. However, we have the end-members (Fe:Va) and (Fe:C) and we can calculate these two partials from the model:

$$G_{\rm Fe:Va} = G_M + \left(\frac{\partial G_M}{\partial y_{\rm Fe,1}} \right)_{T,P,y_{j\neq \rm Fe}} + \left(\frac{\partial G_M}{\partial y_{\rm Va,2}} \right)_{T,P,y_{j\neq \rm Va}} - \sum_s \sum_j y_{js} \left(\frac{\partial G_M}{\partial y_{js}} \right)_{T,P,y_{k\neq j}} \quad (19)$$

$$G_{\rm Fe:C} = G_M + \left(\frac{\partial G_M}{\partial y_{\rm Fe,1}} \right)_{T,P,y_{j\neq \rm Fe}} + \left(\frac{\partial G_M}{\partial y_{\rm C,2}} \right)_{T,P,y_{j\neq \rm C}} - \sum_s \sum_j y_{js} \left(\frac{\partial G_M}{\partial y_{js}} \right)_{T,P,y_{k\neq j}} \quad (20)$$

At equilibrium the partial Gibbs energies of the end-members are related to the chemical potentials of the components as

$$\begin{aligned} G_{\rm FeVa_3} = G_{\rm Fe:Va} = G_{\rm Fe} &= \mu_{\rm Fe} \\ G_{\rm FeC_3} = G_{\rm Fe:C} = G_{\rm Fe} + 3G_{\rm C} &= \mu_{\rm Fe} + 3\mu_{\rm C} \end{aligned}$$

as the chemical potential of Va, $\mu_{\rm Va} = 0$ at equilibrium. We can rearrange to obtain the chemical potential of C:

$$\mu_{\rm C} = G_{\rm C} = \frac{1}{3}(G_{\rm Fe:C} - G_{\rm Fe:Va}) \quad = \quad \frac{1}{3}\left[\left(\frac{\partial G_M}{\partial y_{\rm C,2}}\right)_{T,P,y_{\rm Va}} - \left(\frac{\partial G_M}{\partial y_{\rm Va,2}}\right)_{T,P,y_{\rm C}}\right] \tag{21}$$

As can be seen by the last part of eq. 21 it is independent of the constituent in the first sublattice so it does not matter if we had chosen the end-members (Cr:C) and (Cr:Va). At equilibrium the difference will be the same.

Even if there is no Cr and Fe is the only constituent in its sublattice we can obtain the chemical potentials of both Fe and C because the fraction of C can vary.

But for some CEF models it is not possible to combine the end-members in such a way that one can extract the chemical potentials of the elements. For example in the model for the σ phase above we cannot obtain the individual chemical potentials of the elements from the model. But the method described below to calculate the equilibrium can also be applied to such phases.

3.3 The Gibbs-Duhem relation

In eq. 15 there is no differential of the chemical potentials because the Gibbs-Duhem relation for each phase is:

$$\sum_{\rm A} d\mu_{\rm A} M^\alpha_{\rm A} + S^\alpha_M dT + V^\alpha_M dP = 0 \tag{22}$$

which must be valid at equilibrium.

4 Minimization with constraints

To minimize a function with constraints we apply a Lagrangian equation where each equality constraint has a multiplier. When the constraint is obeyed, the minimum of the Lagrangian is the same as the original function. The multipliers can be used to find the proper way to vary the variables in order to fulfill the constraints.

4.1 The constraints

The variables in the Gibbs energy expression have several constraints. The first is that the sum of the site fractions on each sublattice is unity:

$$g^\alpha_s = 1 - \sum_i y^\alpha_{is} = 0 \tag{23}$$

For phases with ions the net charge must be zero and an external charge balance constraint must be added:

$$Q^\alpha = \sum_s a_s \sum_i \nu_i y^\alpha_{is} = 0 \tag{24}$$

where ν_i is the charge of constituent i.

The total Gibbs energy for a system, G, is given by eq. 1. The number of formula units of a phase α, $\aleph^\alpha$, must be equal to or larger than zero for a stable phase. If it becomes negative during iterations it will be removed from the stable set of phases.

For a closed system we have the constraint on the amount of components

$$f_{\rm A} = \tilde{N}_{\rm A} - N_{\rm A} = \tilde{N}_{\rm A} - \sum_\alpha \aleph^\alpha M^\alpha_{\rm A} = 0 \tag{25}$$

where $\tilde{N}_{\rm A}$ is the prescribed amount of component A. For a phase modeled $({\rm A,B})_{0.75}({\rm A,B})_{0.25}$ the value of $\aleph^\alpha$ is 4 times larger compared to the case that the phase had been modeled $({\rm A,B})_3({\rm A,B})_1$. Both models are allowed and the thermodynamic parameters in the second case must be 4 times larger than those in the first.

4.2 The Lagrangian equation

To minimize the Gibbs energy of a system with constraints we can use a Lagrangian equation as:

$$L = G + \sum_{\rm A} f_{\rm A}\mu_{\rm A} + \sum_\alpha \eta^\alpha_s g^\alpha_s + \sum_\alpha \lambda^\alpha Q^\alpha + \sum_\psi \gamma^\psi \aleph^\psi \tag{26}$$

where $\mu_{\rm A}, \eta^\alpha_s, \lambda^\alpha$ are multipliers for all phases and γ^ψ are multipliers for all phases ψ that are unstable with $\aleph^\psi = 0$. The important property of this Lagrangian is that it has the same extremum points as the Gibbs energy G when the constraints are fulfilled. From now on it will rarely be indicated which variables are kept constant at the partial derivatives, the reader is expected to understand this from the context.

4.3 The derivative of the Lagrangian with respect to phase amounts

For the partial derivative of L with respect to the amount of a stable phase α we get:

$$\frac{\partial L}{\partial \aleph^\alpha} = G^\alpha_M - \sum_{\rm A} f_{\rm A} M^\alpha_{\rm A} = 0 \tag{27}$$

and from this equation we can understand that the Lagrangian multiplier $f_{\rm A} = \mu_{\rm A}$, i.e. the chemical potential of component A. For a rigorous proof see [81Hil].

For an unstable phase ψ which is not included in the stable-phase set we can calculate the derivative:

$$\frac{\partial L}{\partial \aleph^\psi} = G^\psi_M - \sum_i \mu_i M^{\psi_i} + \gamma^\psi = 0 \tag{28}$$

This means that the driving force, γ^ψ, for an unstable phase will be calculated as part of the minimization. An unstable phase may become stable during the iterations for the equilibrium and that is indicated by γ^ψ becoming positive.

On the other hand, if $\aleph^\alpha$ for a stable phase α becomes negative it means this phase has become unstable and should be removed from the stable set. In both cases we must change the set of stable phases as will be discussed in section 5.6.

4.4 The derivative of the Lagrangian with respect to constituent fractions

The partial derivative of L with respect to a constituent fraction y_{is}^{α}, keeping all other variables constant, we get:

$$\frac{\partial L}{\partial y_{is}^{\alpha}} = \aleph^{\alpha}\frac{\partial G_M^{\alpha}}{\partial y_{is}^{\alpha}} - \aleph^{\alpha}\sum_{\mathrm{A}}\mu_{\mathrm{A}}\frac{\partial M_{\mathrm{A}}^{\alpha}}{\partial y_{is}^{\alpha}} - \eta_s^{\alpha} + \lambda^{\alpha}\frac{\partial Q^{\alpha}}{\partial y_{is}^{\alpha}} = 0 \tag{29}$$

We would like to use this equation in an iterative procedure to find the equilibrium and to obtain a linear correction of the difference between the current value of the constituent fractions and those of the equilibrium. To this aim, we expand the partial derivative of $\frac{\partial G_M^{\alpha}}{\partial y_{is}^{\alpha}}$ in a Taylor series of its independent variables dT, dP and dy_{jt}^{α}:

$$\frac{\partial G_M^{\alpha}}{\partial y_{is}^{\alpha}} = \left(\frac{\partial G_M^{\alpha}}{\partial y_{is}^{\alpha}}\right) + \left(\frac{\partial^2 G_M^{\alpha}}{\partial y_{is}^{\alpha}\partial T}\right)dT + \left(\frac{\partial^2 G_M^{\alpha}}{\partial y_{is}^{\alpha}\partial P}\right)dP + \sum_t\sum_j\left(\frac{\partial^2 G_M^{\alpha}}{\partial y_{is}^{\alpha}y_{jt}^{\alpha}}\right)dy_{jt}^{\alpha} \tag{30}$$

where the terms on the right hand side are calculated for the current T, P and Y_i and the term on the left hand side is the linearly "extrapolated" value. The last term on the right hand side is a summation over all constituents j on all sublattices t. In the rest of this paper it will often be written as just a summation over j.

Inserting this in eq. 29 and changing to finite differences we get a system of linear equations depending on the corrections in $\Delta T, \Delta P$ and Δy_{is}^{α}:

$$\begin{aligned}\sum_t\sum_j\left(\frac{\partial^2 G_M^{\alpha}}{\partial y_{is}^{\alpha}y_{jt}^{\alpha}}\right)\Delta y_{jt}^{\alpha} + \frac{\eta_s^{\alpha}}{\aleph^{\alpha}} &= \\ \sum_{\mathrm{A}}\mu_{\mathrm{A}}\left(\frac{\partial M_{\mathrm{A}}^{\alpha}}{\partial y_{is}^{\alpha}}\right) - \left(\frac{\partial G_M^{\alpha}}{\partial y_{is}^{\alpha}}\right) &- \left(\frac{\partial^2 G_M^{\alpha}}{\partial y_{is}^{\alpha}\partial T}\right)\Delta T - \left(\frac{\partial^2 G_M^{\alpha}}{\partial y_{is}^{\alpha}\partial P}\right)\Delta P\end{aligned} \tag{31}$$

In the following we will normally make the simplification that P is constant, i.e. $\Delta P = 0$, as P and T enter the equations in the same way. We are interested to solve this for the fraction corrections Δy_i and can rearrange this in matrix notation, omitting the superscripts:

$$\begin{pmatrix} \frac{\partial^2 G_M}{\partial y_1^2} & \frac{\partial^2 G_M}{\partial y_1\partial y_2} & \cdots & 1 & \cdots \\ \frac{\partial^2 G_M}{\partial y_1\partial y_2} & \frac{\partial^2 G_M}{\partial y_2^2} & \cdots & 1 & \cdots \\ \vdots & & & & \\ 1 & 1 & \cdots & 0 & \cdots \\ \vdots & & & & \end{pmatrix}\begin{pmatrix} \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \frac{\eta_1}{\aleph} \\ \vdots \end{pmatrix} = \begin{pmatrix} \sum_{\mathrm{A}}\mu_{\mathrm{A}}\frac{\partial M_{\mathrm{A}}}{\partial y_1} - \frac{\partial G_M}{\partial y_1} - \frac{\partial^2 G_M}{\partial y_1\partial T}\Delta T \\ \sum_{\mathrm{A}}\mu_{\mathrm{A}}\frac{\partial M_{\mathrm{A}}}{\partial y_2} - \frac{\partial G_M}{\partial y_2} + \frac{\partial S_M}{\partial y_2}\Delta T \\ \vdots \\ 0 \\ \vdots \end{pmatrix} \tag{32}$$

The matrix on the left hand side is called the *phase matrix*, is symmetric and its columns and rows with "1" represent the constraint that sum of constituent fractions on each sublattice is unity. Inverting the *phase matrix* we obtain the corrections of the constituent fractions of each phase in the global potentials $\mu_A, \Delta T$ and ΔP. The use of the inverted phase matrix will be explained in more detail for several specific cases below.

5 Calculating the equilibrium

The solution must be calculated by an iterative process and each iteration is divided into two steps. To simplify the following explanation a substitutional binary system (A,B) is used as an example in many cases. First, an initial step 0 is carried out to find a good start set of stable phases and their constitutions.

5.1 Step 0, Obtaining start values by grid minimizer

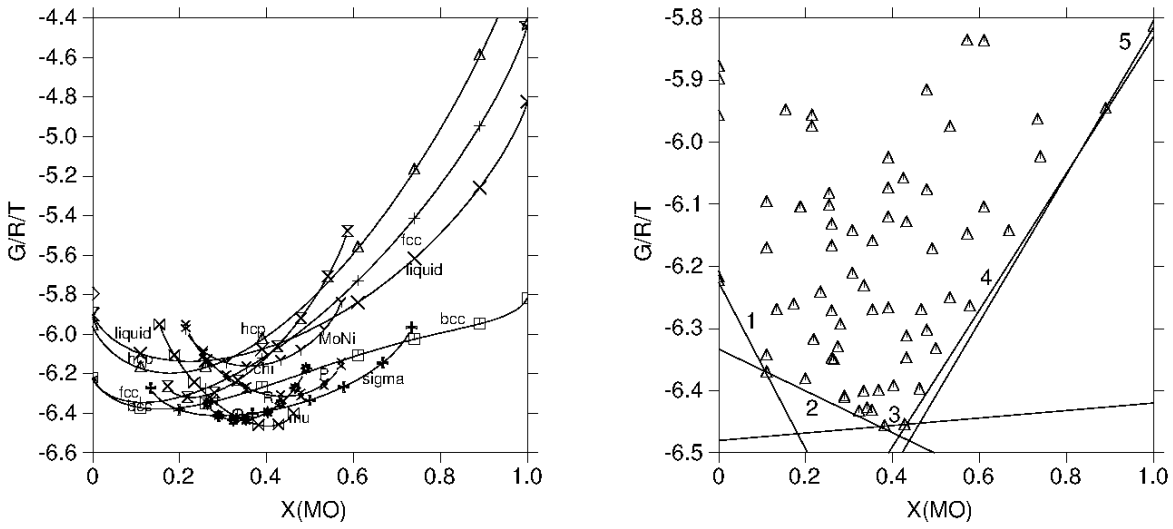
An iterative procedure for calculating the equilibria can only find a local equilibrium, which depends on the initial constitution of the phases. This equilibrium may be significantly different from the global equilibrium. To find the latter, it is necessary to provide a reasonable initial estimate of the stable phases and constitution of all phases for the minimization procedure.

There are many techniques to find a global minimum [09Flo]. A specific problem with thermodynamic equilibrium calculations is that one does not know beforehand how many composition sets of a phase are needed. If a phase has a miscibility gap it may be necessary to create additional composition sets for several phases with different compositions. The solution to this problem, as implemented in OC, is based on the method to find the equilibrium for the case when all phases have a fixed composition. Such an equilibrium is easily found by a combination of simplex and steepest decent methods. In such a calculation we will always have as many stable phases at equilibrium as we have components, and that is the maximum allowed number.

To show how this technique can be adapted to a case when we have solution phases we can use Fig. 2 where the Gibbs energy curves are calculated for the Fe-Mo system at 1400 K. We calculate a number of grid points along the Gibbs energy curves for each phase and then treat each one as a separate phase. The number of grid points does not have to be very large, even in multicomponent systems with more than 10 components a few 1000 points in each phase are sufficient if the grid points are selected with some care.

After finding the grid points representing the equilibrium for the given composition, some of them may be part of the same solution phase and if so we must check if they can be merged into a single point or if they must be treated as separate phases across a miscibility gap. Note that the stable equilibrium for the Fe-Mo system at 1400 K has no miscibility gap but the curvature of one of the phases, bcc, indicate it has a metastable miscibility gap. Such miscibility gaps can be important when using Gibbs energy models to simulate phase transformations.

In Fig. 3 we have a case with two phases where the Gibbs energy curves shows large miscibility gaps. If the iterative algorithm would start with initial compositions as indicated in Fig. 3(b) it will end with the common tangent in Fig. 3(c). This is clearly not the global



(a) (b)

Figure 2: The Gibbs energy curves if the phases in the Fe-Mo system calculated at 1400 K in (a) together with the selected gridpoints. In (b) the gridpoints are treated as individual phases and the "convex hull" is drawn between the gridpoint pairs representing the lowest Gibbs energy at various compositions. Depending on the overall compositions one will end up with a pair of gridpoints indicated by the numbers 1 to 5. Note that the pair 3 has two gridpoints in the same phase (mu) but they do not represent a miscibility gap.

minimum which is shown in Fig. 3(d) but using a grid minimizer as shown in Figs. 3(e) to 3(g), we will obtain start points so the iterative algorithm can find the global equilibrium.

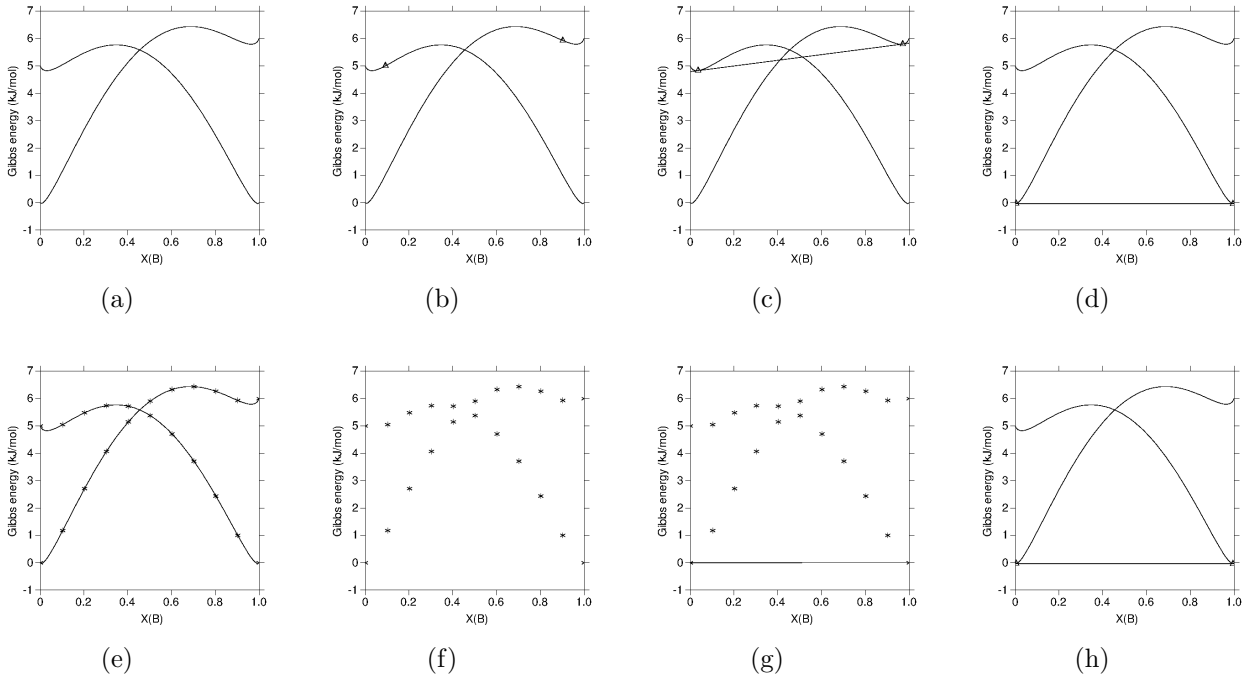


Figure 3: A case where start values matter. In (a) the Gibbs energy curves for two phases with miscibility gaps in a binary system are shown. If we have the initial constitutions marked in (b) the present algorithm will find the local equilibrium shown in (c) which is not the global minimum, which is shown in (d). In order to find good starting values for the two phases we can replace the Gibbs energy curves with calculated gridpoints as shown in (e) and then minimize the Gibbs energy at these gridpoints, treated as a separate phase with fixed composition as shown in (f). The grid minimizer will find the two gridpoints joined by a line in (g) as a minimum and using these compositions as startpoint for the calculation will give the correct global minimum using the present algorithm as in (h).

If an equilibrium has already been calculated with almost the same conditions, as while performing a STEP calculation for a property diagram or a MAP calculation for a phase diagram, it is not necessary to perform a grid minimization each time but we can use the already calculated constitutions of the phases as starting values. But it is important to check now and again if the current equilibrium is still the global one as we may step into a miscibility gap which was not stable initially and which is not detected by the iterative method.

It is also possible that the set of conditions does not allow a global grid minimization, for example if T is not known. In such cases we can start from a default initial constitution of the phases and after the equilibrium has been calculated and the value of T obtained, we can do a grid minimization to find if the calculated equilibrium is the global one. If not, the set of phases found by the grid minimizer can be used to calculate the global equilibrium.

As already mentioned the selection of grid points can be very important for multicomponent systems and as the grid minimizer is part of the model package it is discussed in its documentation [GTP].

5.2 Step 1, the phase matrix

The following derivation will first be given for a binary system, then it will be generalized.

5.2.1 The phase matrix for a binary system

For a substitutional binary phase (A,B) we can derive from eq. 31 the system of equations (P is kept constant)

$$\begin{pmatrix} \frac{\partial^2 G_M^\alpha}{\partial y_1^2} & \frac{\partial^2 G_M^\alpha}{\partial y_1 \partial y_2} & 1 \\ \frac{\partial^2 G_M^\alpha}{\partial y_1 \partial y_2} & \frac{\partial^2 G_M^\alpha}{\partial y_2^2} & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \Delta y_1 \\ \Delta y_2 \\ \eta \end{pmatrix} = \begin{pmatrix} -\frac{\partial G_M}{\partial y_1} - \frac{\partial^2 G_M}{\partial y_1 \partial T}\Delta T + \mu_{\rm A}\frac{\partial M_{\rm A}}{\partial y_1} + \mu_{\rm B}\frac{\partial M_{\rm B}}{\partial y_1} \\ -\frac{\partial G_M}{\partial y_2} - \frac{\partial^2 G_M}{\partial y_2 \partial T}\Delta T + \mu_{\rm A}\frac{\partial M_{\rm A}}{\partial y_1} + \mu_{\rm B}\frac{\partial M_{\rm B}}{\partial y_2} \\ 0 \end{pmatrix} \quad (33)$$

denoting the constituents with index 1 and 2 to emphasize that components and constituents are not the same. On the left hand side we have the phase matrix for the binary (A,B) system, including the constraint that the sum of constituent fractions is unity:

$$\begin{pmatrix} \frac{\partial^2 G_M^\alpha}{\partial y_1^2} & \frac{\partial^2 G_M^\alpha}{\partial y_1 \partial y_2} & 1 \\ \frac{\partial^2 G_M^\alpha}{\partial y_1 \partial y_2} & \frac{\partial^2 G_M^\alpha}{\partial y_2^2} & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Before calculating the second derivatives in this matrix, the constituent fractions should be checked to be sure they are larger than a minimal (positive) value and their sum normalized to unity. We cannot solve eq. 33 now as $\Delta T, \Delta P$ and $\mu_{\rm A}$ are not known but we can invert the phase matrix and as we will use that several times below we write the inverted matrix as:

$$\begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \approx \left(\frac{\partial^2 G_M}{\partial y_i \partial y_j} \right)^{-1} \quad (34)$$

Only a part of this matrix is important because we are not interested in the η multiplier. We can write the solution for Δy_1 and Δy_2 as:

$$\begin{pmatrix} \Delta y_1 \\ \Delta y_2 \end{pmatrix} = \begin{pmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{pmatrix} \begin{pmatrix} -\frac{\partial G_M}{\partial y_{\rm A}} - \frac{\partial^2 G_M}{\partial y_{\rm A} \partial T}\Delta T + \mu_{\rm A}\frac{\partial M_{\rm A}}{\partial y_{\rm A}} + \mu_{\rm B}\frac{\partial M_{\rm B}}{\partial y_{\rm A}} \\ -\frac{\partial G_M}{\partial y_{\rm B}} - \frac{\partial^2 G_M}{\partial y_{\rm B} \partial T}\Delta T + \mu_{\rm A}\frac{\partial M_{\rm A}}{\partial y_{\rm B}} + \mu_{\rm B}\frac{\partial M_{\rm B}}{\partial y_{\rm B}} \end{pmatrix} \quad (35)$$

This gives an important relation between the finite difference of a site fraction expressed as a function of several derivatives of the Gibbs energy and the "multipliers" $\mu_{\rm A}$, which are

identical to the chemical potentials. Writing the equation explicitly for constituent 1, using e_{ij} for the inverted phase matrix, gives:

$$\begin{aligned}\Delta y_1 &= e_{11}\left(-\frac{\partial G_M}{\partial y_1}-\frac{\partial^2 G_M}{\partial y_1 \partial T}\Delta T+\mu_{\rm A}\frac{\partial M_{\rm A}}{\partial y_1}+\mu_{\rm B}\frac{\partial M_{\rm B}}{\partial y_1}\right)+\\ &\quad e_{12}\left(-\frac{\partial G_M}{\partial y_2}-\frac{\partial^2 G_M}{\partial y_2 \partial T}\Delta T+\mu_{\rm A}\frac{\partial M_{\rm A}}{\partial y_2}+\mu_{\rm B}\frac{\partial M_{\rm B}}{\partial y_2}\right)\end{aligned} \tag{36}$$

5.2.2 The general equation for the correction of constituent fractions

Generalizing eq. 36 to any number of constituents, and also including variable P, gives for each constituent i on sublattice s:

$$\Delta y_{is} = c_{iG}+c_{iT}\Delta T+c_{iP}\Delta P+\sum_{\rm A} c_{i{\rm A}}\ \mu_{\rm A} \tag{37}$$

where the coefficients in this equation can be calculated as

$$\begin{aligned} c_{iG} &= -\sum_j e_{ij}\frac{\partial G_M}{\partial y_j}\\ c_{iT} &= -\sum_j e_{ij}\frac{\partial^2 G_M}{\partial T \partial y_j}\\ c_{iP} &= -\sum_j e_{ij}\frac{\partial^2 G_M}{\partial P \partial y_j}\\ c_{i{\rm A}} &= \sum_j e_{ij}\frac{\partial M_{\rm A}}{\partial y_j}\end{aligned} \tag{38}$$

where i is the constituent in sublattice s and the summation over j is for all constituents and all sublattices. A is a component. These coefficients will be used when formulating the equations in section 5.4.1 and also later to calculate partial derivatives of state variables in section 6.1.

As already mentioned we cannot calculate Δy_{is} at present because the values of $\Delta T, \Delta P$ and $\mu_{\rm A}$ are not known.

5.3 Charge balance

If some constituents have a net charge we must add the differential of eq. 24 to ensure that the phase is electrically neutral.

$$Q^\alpha = \sum_s a_s \sum_i \nu_i y_{is}^\alpha = 0 \tag{39}$$

$$\Delta Q^\alpha = \sum_s a_s \sum_i \frac{\partial Q}{\partial y_{is}^\alpha}\Delta y_{is}^\alpha \tag{40}$$

where

$$\frac{\partial Q^{\alpha}}{\partial y_{is}^{\alpha}} = a_s \nu_i \tag{41}$$

This equation is part of the phase matrix and it should be the last row and column (called q). After inverting the phase matrix, the correction of the constituent fractions (eq. 37) will have an additional term

$$\Delta y_{is}^{\alpha} = c_{iG} + c_{iT}\Delta T + c_{iP}\Delta P + \sum_{\mathrm{A}} c_{i\mathrm{A}}\ \mu_{\mathrm{A}} - e_{iQ}Q^{\alpha} \tag{42}$$

where e_{iQ} is the last column in the inverted phase matrix and Q^{α} is the current charge of the phase.

As already mentioned, a more complicated case is the ionic liquid model where the cation/anion site ratios depend on the charge of the opposite site. In such a case there is no extra equation for the charge balance and we cannot use the fact that the derivatives $\frac{\partial M_{\mathrm{A}}}{\partial y_{is}}$ are constants.

5.4 Step 2, the external conditions

After calculating the inverted phase matrices for all phases and saving them we must calculate new values of the intensive variables ($\Delta T, \Delta P$ and μ_i) by making use of the external conditions. The most common set of external conditions is fixed values for T, P and amount of the components, i.e. mass balance conditions. Here we now describe how to formulate the equations to determine these quantities.

For each stable phase α we have

$$G_M^{\alpha} = \sum_{\mathrm{A}} M_{\mathrm{A}}^{\alpha}\mu_{\mathrm{A}} \tag{43}$$

This ensures that all stable phases are on the same hyperplane of chemical potentials. For equilibrium calculations with changing T and P we must take into account any variation in these quantities:

$$G_M^{\alpha} = \sum_{\mathrm{A}} M_{\mathrm{A}}^{\alpha}\mu_{\mathrm{A}} - \frac{\partial G_M}{\partial T}\Delta T - \frac{\partial G_M}{\partial P}\Delta P \tag{44}$$

The other equations depend on the conditions and the number of stable phases.

5.4.1 Condition on the amount of the components

The amount of each element summed over all phases is:

$$N_{\mathrm{A}} - \tilde{N}_{\mathrm{A}} = \sum_{\alpha} \aleph^{\alpha} M_{\mathrm{A}}^{\alpha} - \tilde{N}_{\mathrm{A}} = 0 \tag{45}$$

where $\tilde{N}_{\mathrm{A}}$ is the prescribed amount of moles of component A. The differential of N is

$$\Delta N_{\rm A} \;=\; \sum_{\alpha} \aleph^{\alpha} \Delta M_{\rm A}^{\alpha} + \sum_{\alpha} \Delta\aleph^{\alpha} M_{\rm A}^{\alpha} = 0 \qquad (46)$$

From eq. 9 we have:

$$\Delta M_{\rm A}^{\alpha} \;=\; \sum_{i} \frac{\partial M_{\rm A}^{\alpha}}{\partial y_i^{\alpha}} \Delta y_i^{\alpha} \qquad (47)$$

where the summation over i is for all constituents so we have omitted the sublattice index. We can approximate the differentials with finite differences and for Δy_i^{α} we now use eq. 37 and can write, omitting the phase superscripts and using the coefficients c_{iZ} defined in eq. 38:

$$\Delta M_{\rm A} = \sum_{i} \frac{\partial M_{\rm A}^{\alpha}}{\partial y_i} (c_{iB} \sum_{\rm B} \mu_{\rm B} + c_{iT}\Delta T + c_{iP}\Delta P + c_{iG}) \qquad (48)$$

where the sum over B is for all components. For fixed T and P this becomes:

$$\Delta M_{\rm A} = \sum_{\rm B} \mu_{\rm B} \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} c_{i\rm B} - \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} c_{iG} \qquad (49)$$

Inserting the expression for $c_{i\rm X}$ gives

$$\Delta M_{\rm A} = \sum_{\rm B} \mu_{\rm B} \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} \sum_{j} \frac{\partial M_{\rm B}}{\partial y_j} e_{ij} - \sum_{\rm B} \mu_{\rm B} \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} \sum_{j} \frac{\partial M_{\rm B}}{\partial y_j} e_{ij} - \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} \sum_{j} \frac{\partial G_M}{\partial y_j} e_{ij}$$

$$\Delta M_{\rm A} = \sum_{\rm B} \left(\sum_{i} \sum_{j} e_{ij} \frac{\partial M_{\rm A}}{\partial y_i} \frac{\partial M_{\rm B}}{\partial y_j} \right) \mu_{\rm B} - \sum_{i} \sum_{j} e_{ij} \frac{\partial M_{\rm A}}{\partial y_i} \frac{\partial G_M}{\partial y_j} \qquad (50)$$

5.4.2 Example: a binary system with a single stable phase

If we apply the minimization procedure outline in the previous sections to a binary A-B system with only one stable phase the sum of the fractions of A and B must fulfill the mass balance for each component. We can insert this condition in eq. 46:

$$\Delta N_{\rm A} = \aleph \left(\sum_{\rm B} \mu_{\rm B} \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} c_{i\rm B} - \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} c_{iG} \right) + \Delta\aleph M_{\rm A} \;=\; N_{\rm A} - \tilde{N}_{\rm A} = 0 \qquad (51)$$

In the published paper, [15Sun2], the difference $N_{\rm A} - \tilde{N}_{\rm A}$ was forgotten. After rearranging we obtain for each element:

$$\aleph \sum_{\rm B} \mu_{\rm B} \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} c_{i\rm B} + \Delta\aleph M_{\rm A} \;=\; \aleph \sum_{i} \frac{\partial M_{\rm A}}{\partial y_i} c_{iG} + N_{\rm A} - \tilde{N}_{\rm A} \qquad (52)$$

Again, the sum over i is for all constituents in all sublattices. We can now combine this equation with eq. 43 into a system of linear equations:

$$\begin{pmatrix} M_{\mathrm{A}} & M_{\mathrm{B}} & 0 \\ \aleph \sum_i \frac{\partial M_{\mathrm{A}}}{\partial y_i} c_{i\mathrm{A}} & \aleph \sum_i \frac{\partial M_{\mathrm{A}}}{\partial y_i} c_{i\mathrm{B}} & M_{\mathrm{A}} \\ \aleph \sum_i \frac{\partial M_{\mathrm{B}}}{\partial y_i} c_{i\mathrm{A}} & \aleph \sum_i \frac{\partial M_{\mathrm{B}}}{\partial y_i} c_{i\mathrm{B}} & M_{\mathrm{B}} \end{pmatrix} \begin{pmatrix} \mu_{\mathrm{A}} \\ \mu_{\mathrm{B}} \\ \Delta\aleph \end{pmatrix} = \begin{pmatrix} G_M \\ \aleph \sum_i \frac{\partial M_{\mathrm{A}}}{\partial y_i} c_{iG} + N_{\mathrm{A}} - \tilde{N}_{\mathrm{A}} \\ \aleph \sum_i \frac{\partial M_{\mathrm{B}}}{\partial y_i} c_{iG} + N_{\mathrm{B}} - \tilde{N}_{\mathrm{B}} \end{pmatrix} \quad (53)$$

All terms in eq. 53 (except $\mu_{\mathrm{A}},\ \mu_{\mathrm{B}}$ and $\Delta\aleph$) are known. Solving the above system, we can thus calculate the unknown variables $\mu_{\mathrm{A}},\ \mu_{\mathrm{B}}$ and $\Delta\aleph$ which can then be inserted in eq. 37 to get new constitutions of the α phase. With these new constitutions we calculate again the terms in eq. 53 and continue to iterate until the changes in the quantities are sufficiently small. The matrix on the left hand side of eq. 53 is called the *equilibrium matrix*.

5.4.3 Example: a binary system with two stable phases

When two phases are taking part in the equilibrium calculation, we still use eq. 37 to calculate the corrections to the phase constitutions, but the equilibrium matrix will contain additional terms.

For a binary system with fixed T and P and two stable phases α and β the system of equations to solve is

$$\begin{pmatrix} M_{\mathrm{A}}^{\alpha} & M_{\mathrm{B}}^{\alpha} & 0 & 0 \\ M_{\mathrm{A}}^{\beta} & M_{\mathrm{B}}^{\beta} & 0 & 0 \\ \sum_\varphi \aleph^\varphi \sum_i \frac{\partial M_{\mathrm{A}}^\varphi}{\partial y_i^\varphi} c_{i\mathrm{A}}^\varphi & \sum_\varphi \aleph^\varphi \sum_i \frac{\partial M_{\mathrm{A}}^\varphi}{\partial y_i^\varphi} c_{i\mathrm{B}}^\varphi & M_{\mathrm{A}}^{\alpha} & M_{\mathrm{A}}^{\beta} \\ \sum_\varphi \aleph^\varphi \sum_i \frac{\partial M_{\mathrm{B}}^\varphi}{\partial y_i^\varphi} c_{i\mathrm{A}}^\varphi & \sum_\varphi \aleph^\varphi \sum_i \frac{\partial M_{\mathrm{B}}^\varphi}{\partial y_i^\varphi} c_{i\mathrm{B}}^\varphi & M_{\mathrm{B}}^{\alpha} & M_{\mathrm{B}}^{\beta} \end{pmatrix} \begin{pmatrix} \mu_A \\ \mu_B \\ \Delta\aleph^\alpha \\ \Delta\aleph^\beta \end{pmatrix} =$$

$$\begin{pmatrix} G_M^{\alpha} \\ G_M^{\beta} \\ \sum_\varphi \aleph^\varphi \sum_i \frac{\partial M_{\mathrm{A}}^\varphi}{\partial y_i^\varphi} c_{iG}^\varphi + N_{\mathrm{A}} - \tilde{N}_{\mathrm{A}} \\ \sum_\varphi \aleph^\varphi \sum_i \frac{\partial M_{\mathrm{B}}^\varphi}{\partial y_i^\varphi} c_{iG}^\varphi + N_{\mathrm{B}} - \tilde{N}_{\mathrm{B}} \end{pmatrix} \quad (54)$$

Note that φ is used as phase summation index. Iteratively solving this system for $\mu_{\mathrm{A}}, \mu_{\mathrm{B}}, \Delta\aleph^\alpha$ and $\Delta\aleph^\beta$ together with eq. 37 will eventually lead to the equilibrium.

Note that unstable phases will also have their constitution updated for each iteration using eq. 37 which is valid for all phases. The driving force, γ^ψ for an unstable phase ψ phase is calculated using eq. 28.

If the driving force for a phase that is initially unstable becomes positive during the iteration process, the set of stable phases should be changed. On the contrary, if during the iteration process $\aleph^\alpha$ becomes negative for a phase α, the phase should be set as unstable in the next iteration. Some care must be taken when changing the set of stable phases as discussed in section 5.6.

With fixed T and P and mass balance conditions, there must always be a stable equilibrium even if we may have to change the set of stable phases to find it. But it is possible to prescribe conditions that have no solution as in the next example.

5.4.4 Example: a binary system with unknown T and one stable phase prescribed.

In an equilibrium calculation, it is possible to prescribe that a phase should be stable and this is treated as an "external condition". Such a condition can either be set explicitly by the user or set automatically in a MAP calculation when following a line in a phase diagram. All lines in a phase diagram represent values of the axis variables where the amount of a phase zero. So when mapping a phase diagram one of the axis variables is calculated by the condition that a phase should be stable with zero amount.

We assume in this example that the remaining conditions are fixed amounts of the components A and B and fixed P. Since we must have one phase stable with variable amount, we have in total two stable phases, one with zero amount. Note that with these conditions it is possible that no equilibrium exists.

For a binary system with unknown T and two stable phases, one of which, β, is set with amount zero, we will have an equilibrium matrix as

$$\begin{pmatrix} M_{\rm A}^{\alpha} & M_{\rm B}^{\alpha} & 0 & -\frac{\partial G_M^{\alpha}}{\partial T} \\ M_{\rm A}^{\beta} & M_{\rm B}^{\beta} & 0 & -\frac{\partial G_M^{\beta}}{\partial T} \\ \aleph^{\alpha}\sum_i \frac{\partial M_{\rm A}^{\alpha}}{\partial y_i^{\alpha}} c_{i{\rm A}}^{\alpha} & \aleph^{\alpha}\sum_i \frac{\partial M_{\rm A}^{\alpha}}{\partial y_i^{\alpha}} c_{i{\rm B}}^{\alpha} & M_{\rm A}^{\alpha} & -\sum_i \frac{\partial M_{\rm A}^{\alpha}}{\partial y_i^{\alpha}} c_{iT}^{\alpha} \\ \aleph^{\alpha}\sum_i \frac{\partial M_{\rm B}^{\alpha}}{\partial y_i^{\alpha}} c_{i{\rm A}}^{\alpha} & \aleph^{\alpha}\sum_i \frac{\partial M_{\rm B}^{\alpha}}{\partial y_i^{\alpha}} c_{i{\rm B}}^{\alpha} & M_{\rm B}^{\alpha} & -\sum_i \frac{\partial M_{\rm B}^{\alpha}}{\partial y_i^{\alpha}} c_{iT}^{\alpha} \end{pmatrix} \begin{pmatrix} \mu_A \\ \mu_B \\ \Delta\aleph^{\alpha} \\ \Delta T \end{pmatrix} =$$

$$\begin{pmatrix} G_M^{\alpha} \\ G_M^{\beta} \\ \aleph^{\alpha}\sum_i \frac{\partial M_{\rm A}^{\alpha}}{\partial y_i^{\alpha}} c_{iG}^{\alpha} + N_{\rm A} - \tilde{N}_{\rm A} \\ \aleph^{\alpha}\sum_i \frac{\partial M_{\rm B}^{\alpha}}{\partial y_i^{\alpha}} c_{iG}^{\alpha} + N_{\rm B} - \tilde{N}_{\rm B} \end{pmatrix} \qquad (55)$$

In this system of equations there is only one phase with variable amount, $\Delta\aleph^{\alpha}$, as $\aleph^{\beta} = 0$. Note that we took into account that the Gibbs energy of the phases varies with ΔT according to eq. 44. We must also take into account that the equation for Δy_i depend on ΔT according to eq. 37.

5.4.5 Example: a binary system with two stable phases and one condition on the chemical potential

The final example is for a binary system with fixed T, P, the total amount of one components, $N_{\rm A}$, and the chemical potential, $\mu_{\rm B}$ of the other. Two phases are stable. The system of equations to solve is

$$\begin{pmatrix} M_{\rm A}^{\alpha} & 0 & 0 \\ M_{\rm A}^{\beta} & 0 & 0 \\ \sum_{\varphi}\aleph^{\varphi}\sum_i \frac{\partial M_{\rm A}^{\varphi}}{\partial y_i^{\varphi}} c_{i{\rm A}}^{\varphi} & M_{\rm A}^{\alpha} & M_{\rm A}^{\beta} \end{pmatrix} \begin{pmatrix} \mu_{\rm A} \\ \Delta\aleph^{\alpha} \\ \Delta\aleph^{\beta} \end{pmatrix} = \begin{pmatrix} G_M^{\alpha} - M_{\rm B}^{\alpha}\mu_{\rm B} \\ G_M^{\beta} - M_{\rm B}^{\beta}\mu_{\rm B} \\ \sum_{\varphi}\aleph^{\varphi}\sum_i \frac{\partial M_{\rm A}^{\varphi}}{\partial y_i^{\varphi}}(c_{iG}^{\varphi} - c_{iB}^{\varphi}\mu_{\rm B}) + N_{\rm A} - \tilde{N}_{\rm A} \end{pmatrix} \qquad (56)$$

The summation over φ is over all stable phases.

5.5 Generalizing the equilibrium matrix

All examples shown above have been for binary cases. The phase matrix and the way to calculate corrections to the constituent fractions, eq. 37 are completely general and can be used for every phase in a multicomponent system.

As shown in the examples, the system of equations to solve must be adapted to the conditions set by the user and it will also change if the number of stable phases changes during the iterations. The coefficients calculated from the inverted phase matrix, eq. 38, are important in constructing the equilibrium matrix.

5.6 Changing the set of stable phases

When the amount of a stable phase becomes negative at an iteration it means this phase should be removed from the set of stable phases. On the other hand if the driving force for an unstable phase according to eq. 28 becomes positive that phase should be added. From eq. 28

$$\gamma^{\psi} = \sum_{\mathrm{A}} \mu_{\mathrm{A}} M_{\mathrm{A}}^{\psi_i} - G_M^{\psi} \tag{57}$$

where the sum over A is for all components. If T and P are variable the terms with ΔT and ΔP are also included in this equation as in eq. 44.

This is basically a trivial operation but we must take care that the same phase is not removed and added at every second iteration. Normally we should allow a few iterations after a change in the set of stable phases before another change is allowed.

It can also happen that the amount of the single stable phase becomes negative and it is clearly impossible to have a system without a single stable phase. This may indicate that the set of external conditions is unreasonable or that the model parameters are wrong.

A third case that may cause problem is when more phases become stable than allowed by the Gibbs phase rule:

$$f = n - p + 2 \tag{58}$$

where f is the degrees of freedom, n number of components, p number of stable phases and 2 represent variable T and P. In order to calculate an equilibrium we must set enough conditions so that f is zero. For a binary system this means 4 conditions. If one condition is constant P, we can at most have 3 phases stable.

For a calculation in a binary system with fixed T and P cannot have more than 2 stable phases. If a third phase wants to become stable this one must replace one of the already stable phases or we must allow T to vary in order to determine the invariant T, for example during a phase diagram calculation.

5.7 Condition on volume

If the volume is prescribed as constant, $\tilde{V}$, we have an equation:

$$dV \;\; = \;\; V - \tilde{V} = 0 \tag{59}$$

where

$$V = \sum_\alpha \aleph^\alpha V_M^\alpha \tag{60}$$

$$V_M^\alpha = \left(\frac{\partial G_M^\alpha}{\partial P}\right)_{T,Y} \tag{61}$$

It is not necessary to have variable P, we may be able to change the volume even at constant P, for example by varying the amount of phases with different molar volumes or having a condition on a chemical potential which can change the amount of material in the system. We expand the differential of $dV = \Delta V$ as:

$$\begin{aligned}
\Delta V &= \sum_\alpha \aleph^\alpha \left(\frac{\partial^2 G_M^\alpha}{\partial P \partial T}\Delta T + \frac{\partial^2 G_M^\alpha}{\partial P^2}\Delta P + \sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha}\Delta y_i^\alpha\right) + \sum_\alpha \frac{\partial G_M^\alpha}{\partial P}\Delta\aleph^\alpha \\
&= \sum_\alpha \aleph^\alpha \left(\sum_{\mathrm{A}}\sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha} c_{iA}\mu_{\mathrm{A}} + \left[\frac{\partial^2 G_M^\alpha}{\partial P \partial T} + \sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha} c_{iT}\right]\Delta T + \right. \\
&\quad \left. \left[\frac{\partial^2 G_M^\alpha}{\partial P^2}\Delta P + \sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha} c_{iP}\right]\Delta P + \sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha} c_{iG}\right) + \sum_\alpha \frac{\partial G_M^\alpha}{\partial P}\Delta\aleph^\alpha \\
&= V - \tilde{V} = 0
\end{aligned} \tag{62}$$

where Δy_i can be expressed as a function of $\Delta T, \Delta P$ and μ_{A} using eq. 37. Rearranging the equation for the equilibrium matrix for the unknown $\Delta T, \Delta P$ and μ_{A} gives:

$$\begin{aligned}
\sum_\alpha \aleph^\alpha \sum_{\mathrm{A}}\sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha} c_{iA}\mu_{\mathrm{A}} + \sum_\alpha \aleph^\alpha \left(\frac{\partial^2 G_M^\alpha}{\partial P \partial T} + \sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha} c_{iT}\right)\Delta T \quad + \\
\sum_\alpha \aleph^\alpha \left(\frac{\partial^2 G_M^\alpha}{\partial P^2} + \sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha} c_{iP}\right)\Delta P + \sum_\alpha \frac{\partial G_M^\alpha}{\partial P}\Delta\aleph^\alpha \quad = \\
-\sum_\alpha \aleph^\alpha \sum_i \frac{\partial^2 G_M^\alpha}{\partial P \partial y_i^\alpha} c_{iG} + V - \tilde{V}
\end{aligned} \tag{63}$$

In the next sections we will not specify the phase when obvious from the context.

5.8 Constant Gibbs energy or entropy

In most common calculations, it is rare to have a condition set on the Gibbs energy or entropy of the system. This is however useful for heat balance equations and we thus consider this case too. Similarly to what done for the volume, we obtain

$$dG = G - \tilde{G} = 0 \tag{64}$$

where

$$G = \sum_\alpha \aleph^\alpha G_M^\alpha \tag{65}$$

$$G_M^\alpha = \sum_{\mathrm{A}} M_{\mathrm{A}}^\alpha \mu_{\mathrm{A}} \tag{66}$$

$$dG_M^\alpha = \sum_{\mathrm{A}} dM_{\mathrm{A}}^\alpha \mu_{\mathrm{A}} \tag{67}$$

and we can approximate $dM_{\mathrm{A}}^\alpha = \Delta M_{\mathrm{A}}^\alpha$ as in eq. 48. As we are only dealing with linear changes all terms multiplied with two potentials or potential differences are ignored and in the above equation we keep only:

$$dG = \sum_\alpha \aleph^\alpha \sum_{\mathrm{A}} \sum_i \frac{\partial M_{\mathrm{A}}^\alpha}{\partial y_i} c_{iG} \mu_{\mathrm{A}} = G - \tilde{G} = 0 \tag{68}$$

This looks nice and simple but maybe not so useful. For a condition on the entropy we have

$$dS = S - \tilde{S} = 0 \tag{69}$$

where

$$S = \sum_\alpha \aleph^\alpha S_M^\alpha \tag{70}$$

$$S_M^\alpha = -\left(\frac{\partial G_M^\alpha}{\partial T}\right)_{P,Y} = -\frac{\partial}{\partial T}\left(\sum_{\mathrm{A}} M_{\mathrm{A}}^\alpha \mu_{\mathrm{A}}\right) = -\sum_{\mathrm{A}} M_{\mathrm{A}}^\alpha \frac{\partial \mu_{\mathrm{A}}}{\partial T} \tag{71}$$

$$dS_M^\alpha = -\sum_{\mathrm{A}} dM_{\mathrm{A}}^\alpha \frac{\partial \mu_{\mathrm{A}}}{\partial T} \tag{72}$$

but I have no idea how to calculate $\frac{\partial \mu_{\mathrm{A}}}{\partial T}$. This must be the wrong track. If we do not introduce the chemical potentials we can write

$$dS_M^\alpha = -\frac{\partial^2 G_M^\alpha}{\partial T^2}\Delta T - \frac{\partial^2 G_M^\alpha}{\partial T \partial P}\Delta P - \sum_i \frac{\partial^2 G_M^\alpha}{\partial T \partial y_i}\Delta y_i \tag{73}$$

and we can formulate an equation as we did for the volume in eq. 62:

$$\begin{aligned} dS &= -\sum_\alpha \aleph^\alpha \left(\frac{\partial^2 G_M^\alpha}{\partial T^2}\Delta T + \frac{\partial^2 G_M^\alpha}{\partial T \partial P}\Delta P + \sum_i \frac{\partial^2 G_M^\alpha}{\partial T \partial y_i}\Delta y_i\right) - \sum_\alpha \frac{\partial G_M^\alpha}{\partial T}\Delta \aleph^\alpha \\ &= S - \tilde{S} = 0 \end{aligned} \tag{74}$$

where Δy_i can be expressed as a function of $\Delta T, \Delta P$ and μ_{A} using equation 37.

$$\begin{aligned} -\sum_\alpha \aleph^\alpha \frac{\partial^2 G_M^\alpha}{\partial T^2}\Delta T - \sum_\alpha \aleph^\alpha \frac{\partial^2 G_M^\alpha}{\partial T \partial P}\Delta P - \sum_\alpha \aleph^\alpha \sum_i \frac{\partial^2 G_M^\alpha}{\partial T \partial y_i}\Delta y_i - \sum_\alpha \frac{\partial G_M^\alpha}{\partial T}\Delta \aleph^\alpha &= S - \tilde{S} \\ -\sum_\alpha \aleph^\alpha \frac{\partial^2 G_M^\alpha}{\partial T^2}\Delta T - \sum_\alpha \aleph^\alpha \frac{\partial^2 G_M^\alpha}{\partial T \partial P}\Delta P - \sum_\alpha \frac{\partial G_M^\alpha}{\partial T}\Delta \aleph^\alpha & \\ -\sum_\alpha \aleph^\alpha \sum_i \frac{\partial^2 G_M^\alpha}{\partial T \partial y_i}\left(\sum_{\mathrm{A}} c_{i\mathrm{A}}\mu_{\mathrm{A}} + c_{iT}\Delta T + c_{iP}\Delta P + c_{iG}\right) &= S - \tilde{S} \end{aligned}$$

$$-\sum_\alpha \aleph^\alpha (\frac{\partial^2 G_M^\alpha}{\partial T^2} + \sum_i \frac{\partial^2 G_M^\alpha}{\partial T \partial y_i} c_{iT}) \Delta T - \sum_\alpha \aleph^\alpha (\frac{\partial^2 G_M^\alpha}{\partial T \partial P} + \sum_i \frac{\partial^2 G_M^\alpha}{\partial T \partial y_i} c_{iP}) \Delta P$$
$$- \sum_\alpha \aleph^\alpha \sum_{\mathrm{A}} \sum_i \frac{\partial^2 G_M^\alpha}{\partial T \partial y_i} c_{i\mathrm{A}} \mu_{\mathrm{A}} - \sum_\alpha \frac{\partial G_M^\alpha}{\partial T} \Delta \aleph^\alpha = \sum_\alpha \aleph^\alpha \sum_i \frac{\partial^2 G_M^\alpha}{\partial T \partial y_i} c_{iG} \quad + \quad S - \tilde{S} \tag{75}$$

This means we should not use eq. 68 for a condition on G, but write the differential of G_M^α as:

$$dG_M^\alpha \quad = \quad \frac{\partial G_M^\alpha}{\partial T} \Delta T + \frac{\partial G_M^\alpha}{\partial P} \Delta P + \sum_i \frac{\partial G_M^\alpha}{\partial y_i} \Delta y_i \tag{76}$$

and the equation for a constant Gibbs energy condition would be written

$$\begin{aligned} dG &= \sum_\alpha \aleph^\alpha (\frac{\partial G_M^\alpha}{\partial T} \Delta T + \frac{\partial G_M^\alpha}{\partial P} \Delta P + \sum_i \frac{\partial G_M^\alpha}{\partial y_i} \Delta y_i) + \sum_\alpha G_M^\alpha \Delta \aleph^\alpha \\ &= G - \tilde{G} = 0 \end{aligned} \tag{77}$$

or written as an equation of the variables $\Delta\aleph, \Delta T, \Delta P$ and μ:

$$\sum_\alpha \aleph^\alpha (\frac{\partial G_M^\alpha}{\partial T} + \sum_i \frac{\partial G_M^\alpha}{\partial y_i} c_{iT}) \Delta T + \sum_\alpha \aleph^\alpha (\frac{\partial G_M^\alpha}{\partial P} + \sum_i \frac{\partial G_M^\alpha}{\partial y_i} c_{iP}) \Delta P$$
$$+ \sum_\alpha \aleph^\alpha \sum_{\mathrm{A}} \sum_i \frac{\partial G_M^\alpha}{\partial y_i} c_{i\mathrm{A}} \mu_{\mathrm{A}} + \sum_\alpha G_M^\alpha \Delta \aleph^\alpha = -\sum_\alpha \aleph^\alpha \sum_i \frac{\partial G_M^\alpha}{\partial y_i} c_{iG} \quad + \quad G - \tilde{G} \tag{78}$$

Just to remind you

$$c_{i\mathrm{A}} \quad = \quad \sum_j e_{ij} \frac{\partial M_{\mathrm{A}}}{\partial y_j} \tag{79}$$

where e_{ij} is the inverted phase matrix. The subroutine **calc_dgdytermsh** calculates

$$mamu(\mathrm{A}) \quad = \quad \sum_i F_i c_{i\mathrm{A}} = \sum_i \sum_j F_i e_{ij} \frac{\partial M_{\mathrm{A}}}{\partial y_j} \tag{80}$$

$$maT \quad = \quad \sum_i F_i c_{iT} = \sum_i \sum_j F_i e_{ij} \frac{\partial^2 G_M}{\partial T \partial y_j} \tag{81}$$

$$maP \quad = \quad \sum_i F_i c_{iT} = \sum_i \sum_j F_i e_{ij} \frac{\partial^2 G_M}{\partial P \partial y_j} \tag{82}$$

$$maG \quad = \quad \sum_i F_i c_{iG} = \sum_i \sum_j F_i e_{ij} \frac{\partial G_M}{\partial y_j} \tag{83}$$

where F_i is an array passed to this subroutine and $mamu(A)$ is an array with one value for each component. For a condition on G we have $F_i = \frac{\partial G_M}{\partial y_i}$.

For the heat balance explained below it is $F_i = \frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i}$ and for a mass balance equation on the amount of component B, N_{B}, it is $F_i = \frac{\partial M_{\mathrm{B}}}{\partial y_i}$.

5.9 Heat balance calculation

The enthalpy is $H = G + TS$ and we are frequently interested to calculate the equilibrium at constant enthalpy. We have as before the equation:

$$dH = H - \tilde{H} = 0 \tag{84}$$

where $\tilde{H}$ is the prescribed value of the enthalpy and

$$H = \sum_\alpha \aleph^\alpha H_M^\alpha \tag{85}$$

$$dH = \sum_\alpha \aleph^\alpha dH_M^\alpha + \sum_\alpha H_M^\alpha d\aleph^\alpha \tag{86}$$

$$H_M^\alpha = G_M^\alpha + TS_M^\alpha \tag{87}$$

$$dH_M^\alpha = dG_M^\alpha + TdS_M^\alpha + S_M^\alpha dT \tag{88}$$

Thermodynamics is confusing, maybe we should use:

$$dH_M^\alpha = dG_M^\alpha + TdS_M^\alpha \tag{89}$$

as we use such an equation together with $\Delta G = 0$ when we have a phase transformation to obtain $\Delta H = T\Delta S$. But this relation is only valid at fixed T, i.e. $dT = 0$. In the general case we must also include dT.

In eq. 88 the differentials for G_M and S_M are given by eqs 73 and 76 respectively for a single phase. Combining these equations we have:

$$\begin{aligned} dH_M^\alpha = & \left(\frac{\partial G_M}{\partial T} - T\frac{\partial^2 G_M}{\partial T^2}\right)\Delta T + \left(\frac{\partial G_M}{\partial P} - T\frac{\partial^2 G_M}{\partial T \partial P}\right)\Delta P + \\ & \sum_i \left(\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i}\right)\Delta y_i - \frac{\partial G_M}{\partial T}\Delta T \end{aligned} \tag{90}$$

Summing over all phases and including the change in the phase amount, $\left(G_M - T\frac{\partial G_M}{\partial T}\right)\Delta\aleph^\alpha$, this gives

$$\begin{aligned} \Delta H = & \sum_\alpha \aleph^\alpha \left[\left(\frac{\partial G_M}{\partial T} - T\frac{\partial^2 G_M}{\partial T^2}\right)\Delta T + \left(\frac{\partial G_M}{\partial P} - T\frac{\partial^2 G_M}{\partial T \partial P}\right)\Delta P + \right. \\ & \left. \sum_i \left(\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i}\right)\Delta y_i - \frac{\partial G_M}{\partial T}\Delta T\right] - + \\ & \sum_\alpha \left(G_M - T\frac{\partial G_M}{\partial T}\right)\Delta\aleph^\alpha = H - \tilde{H} = 0 \end{aligned} \tag{91}$$

where we calculate the partial derivatives with the appropriate variables T, P and y_i kept fixed. We can do this for a single phase as we have already separated out the change in the amount of the phases, $\Delta\aleph^\alpha$.

Observing that the two terms $\frac{\partial G}{\partial T}$ cancel and inserting the expression for Δy_i from eq. 37 we get:

$$\begin{aligned}\sum_\alpha \aleph^\alpha \left[\left(-T\frac{\partial^2 G_M}{\partial T^2} \right) \Delta T + \left(\frac{\partial G_M}{\partial P} - T\frac{\partial^2 G_M}{\partial T \partial P} \right) \Delta P + \right. \\ \left. \sum_i \left(\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i} \right) (c_{iG} + c_{iT}\Delta T + c_{iP}\Delta P + \sum_{\mathrm{A}} c_{iA}\mu_{\mathrm{A}}) \right] + \\ \sum_\alpha \left(G_M - T\frac{\partial G_M}{\partial T} \right) \Delta\aleph^\alpha \;=\; H - \tilde{H}\end{aligned} \tag{92}$$

When further rearranging to have the coefficients in the equilibrium matrix for the independent variables $\Delta T, \Delta P, \mu_{\mathrm{A}}$ and $\Delta\aleph^\alpha$ in the equation for fixed H:

$$\begin{aligned}\sum_\alpha \aleph^\alpha \left(-T\frac{\partial^2 G_M}{\partial T^2} + \sum_i (\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i}) c_{iT} \right) \Delta T \;+ \\ \sum_\alpha \aleph^\alpha \left(\frac{\partial G_M}{\partial P} - T\frac{\partial^2 G_M}{\partial T \partial P} + \sum_i (\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i}) c_{iP} \right) \Delta P \;+ \\ \sum_\alpha \aleph^\alpha \sum_{\mathrm{A}} \sum_i \left(\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i} \right) c_{iA}\mu_{\mathrm{A}} + \sum_\alpha \left(G_M - T\frac{\partial G_M}{\partial T} \right) \Delta\aleph^\alpha \;= \\ -\sum_\alpha \aleph^\alpha \sum_i \left(\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i} \right) c_{iG} + H - \tilde{H}\end{aligned} \tag{93}$$

and if we prescribed the enthalpy of a specific phase α, $\tilde{H}^\alpha$:

$$\begin{aligned}\left(-T\frac{\partial^2 G_M}{\partial T^2} + \sum_i (\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i}) c_{iT} \right) \Delta T \;+ \\ \left(\frac{\partial G_M}{\partial P} - T\frac{\partial^2 G_M}{\partial T \partial P} + \sum_i (\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i}) c_{iP} \right) \Delta P \;+ \\ \sum_{\mathrm{A}} \sum_i \left(\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i} \right) c_{iA}\mu_{\mathrm{A}} + \left(G_M - T\frac{\partial G_M}{\partial T} \right) \Delta\aleph^\alpha \;= \\ -\sum_i \left(\frac{\partial G_M}{\partial y_i} - T\frac{\partial^2 G_M}{\partial T \partial y_i} \right) c_{iG} + H^\alpha - \tilde{H}^\alpha\end{aligned} \tag{94}$$

Note that the enthalpy of the α phase in eq. 94 depends on the amount of the phase. If it is not stable, H^α will be zero.

If we prescribe the volume of a system, $\tilde{V}$, we use

$$V \;=\; \left(\frac{\partial G_M}{\partial P} \right)_{T, y_i} \tag{95}$$

$$\Delta V \;=\; \sum_i \frac{\partial^2 G_M}{\partial P \partial y_i} \Delta y_i + \frac{\partial^2 G_M}{\partial P \partial T} \Delta T + \frac{\partial^2 G_M}{\partial P^2} \Delta P \tag{96}$$

and eq. 93 can be transformed into:

$$\sum_\alpha \aleph^\alpha \sum_{\mathrm{A}} \sum_i \frac{\partial^2 G_M}{\partial P \partial y_i} c_{iA}\mu_{\mathrm{A}} + \sum_\alpha \aleph^\alpha (\frac{\partial^2 G_M}{\partial T \partial P} + \sum_i \frac{\partial^2 G_M}{\partial P \partial y_i} c_{iT})\Delta T + \sum_\alpha \aleph^\alpha (\frac{\partial^2 G_M}{\partial P^2} + \sum_i \frac{\partial^2 G_M}{\partial P \partial y_i} c_{iP})\Delta P + \sum_\alpha \frac{\partial G_M}{\partial P}\Delta \aleph^\alpha = -\sum_\alpha \aleph^\alpha \sum_i \frac{\partial^2 G_M}{\partial P \partial y_i} c_{iG} + V - \tilde{V} \quad (97)$$

We see again how useful it is to have expressed Δy_i as a function of the potentials and that all second derivatives are calculated analytically in the model package. It makes it easy to set up equations for each equation independent of all other conditions. In section 5.10 we show how to handle normalized state variables as conditions.

5.10 Normalized state variables as conditions

In most of the examples above the total amounts of the components has been used as condition. This simplifies the equations because if we use normalized state variables like mole fractions rather than total amounts we can take into account the derivatives of the normalizing property when deriving the equations for the system and that these may change during the iterations. In order to handle this, the following equations must be used as derived by [84Jan].

Conditions on extensive properties can in general be set on a global variable (summed over all stable phases) or for a specific phase and it can be a total value or a normalized one (normalized per moles, mass or volume). For a global total property we can formulate the condition as for N_{A} in eq. 45.

For a normalized phase specific variable we have, using Z for the variable and K for the normalizing quantity:

$$z^\alpha = \frac{Z_M^\alpha}{K_M^\alpha} \quad (98)$$

If the prescribed value is $\tilde{z}^\alpha$ we have

$$\Delta z^\alpha = \frac{\Delta Z_M^\alpha - z^\alpha \Delta K_M^\alpha}{K_M^\alpha} = z^\alpha - \tilde{z}^\alpha = 0 \quad (99)$$

where the subscript M means per mole formula unit of the phase. The finite difference ΔZ^α can be expressed in the model variables of the phase, $T,\ P$ and y_{is}^α, as:

$$\Delta Z_M^\alpha = \frac{\partial Z_M^\alpha}{\partial T}\Delta T + \frac{\partial Z_M^\alpha}{\partial P}\Delta P + \sum_t \sum_j \frac{\partial Z_M^\alpha}{\partial y_{js}^\alpha}\Delta y_{js}^\alpha \quad (100)$$

and similarly for ΔK_M^α. Here we can again use eq. 37 to express Δy_{js}^α as a function of $\Delta T,\ \Delta P$ and the chemical potentials μ_{A}.

To prescribe a global value for the whole system we can simply sum over all stable phases:

$$Z = \sum_\alpha \aleph^\alpha Z_M^\alpha \quad (101)$$

Using a normalizing factor K for the whole system gives z:

$$z = \frac{Z}{K} \tag{102}$$

Expressing the deviation from the prescribed value $\tilde{z}$ using finite differences is:

$$\Delta z = \tilde{z} - z = \frac{1}{K}(\sum_\alpha \aleph^\alpha \Delta Z_M^\alpha + Z_M^\alpha \Delta \aleph^\alpha) - \frac{Z}{K^2}(\sum_\alpha \aleph^\alpha \Delta K_M^\alpha + K_M^\alpha \Delta \aleph^\alpha) \tag{103}$$

As before we can replace ΔZ_M^α and ΔK_M^α to express this as a function of ΔT, ΔP and the chemical potentials μ_A using eq. 100 and eq. 37. A phase with prescribed fixed amount will have $\Delta \aleph = 0$.

The use of eq. 103 is when there is a condition on the mole fractions of a component together with a chemical potential as shown in section 5.12.1.

5.10.1 Condition on the molar enthalpy

In this case we have to sum the enthalpy for the stable phases and divide with the total almount. This section were rewritten 2019 as I wanted to have condition on molar entropy and I found that normallizzed enthalpy did not work. Several changes has been made both in this documentation and in the code. Hopefully it will make it easier to generallize and handle future additions.

$$\begin{aligned}
Z = H &= \sum_\alpha \aleph^\alpha H_M^\alpha && (104)\\
K = N &= \sum_\alpha \aleph^\alpha M^\alpha && (105)\\
z = \frac{Z}{K} = H_m = \frac{H}{N} &= \frac{\sum_\alpha \aleph^\alpha H^\alpha}{\sum_\alpha \aleph^\alpha M^\alpha} && (106)\\
dz &= \frac{dH}{N} - \frac{H}{N^2} dN && (107)\\
H_M^\alpha &= G_M^\alpha - T\left(\frac{\partial G_M^\alpha}{\partial T}\right)_{P,y_i} && (108)\\
M^\alpha = N^\alpha &= \sum_\mathrm{A} \sum_s a_s^\alpha \sum_i b_{\mathrm{A}i} y_{is}^\alpha && (109)
\end{aligned}$$

where G_M^α is the molar Gibbs energy and M^α (according to eq. 4) is the amount of moles of components, in both cases per mole formula unit of phase α.

When used as condition we need the differential, eq. 107, and for dH we use:

$$\begin{aligned}
dH = \Delta H &= \sum_\alpha \aleph^\alpha \left(\sum_i \frac{\partial H_M^\alpha}{\partial y_i}\Delta y_i + \frac{\partial H_M^\alpha}{\partial T}\Delta T + \frac{\partial H_M^\alpha}{\partial P}\Delta P\right) + \sum_\alpha H_M^\alpha \Delta \aleph^\alpha \\
&= \sum_\alpha \aleph^\alpha \sum_i \frac{\partial H_M^\alpha}{\partial y_i} \sum_\mathrm{A} c_{i\mathrm{A}} \mu_\mathrm{A} + \sum_\alpha \aleph^\alpha \left(\frac{\partial H_M^\alpha}{\partial T} + \sum_i \frac{\partial H_M^\alpha}{\partial y_i} c_{iT}\right)\Delta T + \\
&\quad \sum_\alpha \aleph^\alpha \left(\frac{\partial H_M^\alpha}{\partial P} + \sum_i \frac{\partial H_M^\alpha}{\partial y_i} c_{iP}\right)\Delta P + \sum_\alpha \aleph^\alpha \sum_i \frac{\partial H_M^\alpha}{\partial y_i} c_{iG} + \sum_\alpha H_M^\alpha \Delta \aleph^\alpha
\end{aligned} \tag{110}$$

and for dN we use:

$$\begin{aligned} dN = \Delta N &= \sum_\alpha \sum_{\mathrm{A}} M^\alpha_{\mathrm{A}} \Delta \aleph^\alpha + \sum_\alpha \aleph^\alpha \sum_i \sum_{\mathrm{A}} \frac{\partial M^\alpha_{\mathrm{A}}}{\partial y_i} \Delta y_i \\ &= \sum_\alpha \sum_{\mathrm{A}} M^\alpha_{\mathrm{A}} \Delta \aleph^\alpha + \sum_\alpha \aleph^\alpha \sum_i \sum_{\mathrm{A}} \frac{\partial M^\alpha_{\mathrm{A}}}{\partial y_i} (\sum_{\mathrm{B}} c_{i\mathrm{B}} \mu_{\mathrm{B}} + c_{iT} \Delta T + c_{iP} \Delta P + c_{iG}) \end{aligned} \tag{111}$$

Note that $c_{i\mathrm{A}}, c_{iT}$ etc. are of course different for each phase even if this is not indicated. As there is are special subroutines for calculating the terms in the Δy_i expression there is no need to rearrange the terms in eqs. 110 and 111. When the coefficient in front of Δy_i is an array like $\frac{\partial H^\alpha_M}{\partial y_i}$ the subroutine "calc_dgdytermshm" is used, when it is a constant like $H_m = H/N$ one can use "calc_dgdytems1".

Without expanding Δy_i we can insert eqs. 110 and 111 in eq. 107 together with $z = \frac{H}{N}$:

$$\begin{aligned} \frac{1}{N} \Bigg[\sum_\alpha \aleph^\alpha \left(\sum_i \frac{\partial H^\alpha_M}{\partial y_i} \Delta y_i + \frac{\partial H^\alpha_M}{\partial T} \Delta T + \frac{\partial H^\alpha_M}{\partial P} \Delta P \right) + \sum_\alpha H^\alpha_M \Delta \aleph^\alpha - \\ H_m \sum_\alpha \sum_{\mathrm{A}} M^\alpha_{\mathrm{A}} \Delta \aleph^\alpha - H_m \sum_\alpha \aleph^\alpha \sum_i \sum_{\mathrm{A}} \frac{\partial M^\alpha_{\mathrm{A}}}{\partial y_i} \Delta y_i \Bigg] &= H_m - \tilde{H}_m \end{aligned} \tag{112}$$

If we expand the Δy_i from eq. 37 and rearrange a little we have:

$$\begin{aligned} \frac{1}{N} \sum_\alpha \aleph^\alpha \Bigg(\sum_i \frac{\partial H^\alpha_M}{\partial y_i} \sum_{\mathrm{B}} c_{i\mathrm{B}} \mu_{\mathrm{B}} + (\frac{\partial H^\alpha_M}{\partial T} + \sum_i \frac{\partial H^\alpha_M}{\partial y_i} c_{iT}) \Delta T + \\ (\frac{\partial H^\alpha_M}{\partial P} + \sum_i \frac{\partial H^\alpha_M}{\partial y_i} c_{iP}) \Delta P + \sum_i \frac{\partial H^\alpha_M}{\partial y_i} c_{iG} \Bigg) + \\ \frac{1}{N} \left(\sum_\alpha H^\alpha_M - H_m \sum_\alpha \sum_{\mathrm{A}} M^\alpha_{\mathrm{A}} \right) \Delta \aleph^\alpha - \\ \frac{H_m}{N} \sum_\alpha \aleph^\alpha \sum_i \sum_{\mathrm{A}} \frac{\partial M^\alpha_{\mathrm{A}}}{\partial y_i} \left(\sum_{\mathrm{B}} c_{i\mathrm{B}} \mu_{\mathrm{B}} + c_{iT} \Delta T + c_{iP} \Delta P + c_{iG} \right) &= H_m - \tilde{H}_m \end{aligned} \tag{113}$$

but it is neither convenient nor necessary because the different subroutines calc_dgdytermsX provide the different coefficients for $\mu_{\mathrm{A}}, \Delta T, \Delta P$ and the constant term multiplied with c_{iG}. We just replace $H_M = G_M - T\frac{\partial G_M}{\partial T}$ and $\frac{H}{N} = H_m$, noting that $\frac{\partial H_M}{\partial T} = -T\frac{\partial^2 G_M}{\partial T^2}$:

$$\begin{aligned} \frac{1}{N} [\sum_\alpha \aleph^\alpha (\sum_i (\frac{\partial G^\alpha_M}{\partial y_i} - T\frac{\partial^2 G^\alpha_M}{\partial T \partial y_i}) \Delta y_i - T\frac{\partial^2 G^\alpha_M}{\partial T^2} \Delta T + \\ (\frac{\partial G_M}{\partial P} - T\frac{\partial^2 G^\alpha_M}{\partial T \partial P})) \Delta P + \sum_\alpha (G_M - T\frac{\partial G^\alpha_M}{\partial T}) \Delta \aleph^\alpha - \\ H_m \sum_\alpha \sum_{\mathrm{A}} M^\alpha_{\mathrm{A}} \Delta \aleph^\alpha + H_m \sum_\alpha \aleph^\alpha \sum_i \sum_{\mathrm{A}} \frac{\partial M^\alpha_{\mathrm{A}}}{\partial y_i} \Delta y_i] &= H_m - \tilde{H}_m \end{aligned} \tag{114}$$

Using eq. 114 we have to call the subroutine calc_dgdytermshm once with

$$f^\alpha_i = \frac{\partial G^\alpha_M}{\partial y_i} - T\frac{\partial G^\alpha_M}{\partial y_i \partial T} \tag{115}$$

and calc_dgdyterms1 with

$$f_i^\alpha = \sum_{\mathrm{A}} \frac{\partial M_{\mathrm{A}}^\alpha}{\partial y_i} \tag{116}$$

to obtain the relevant coefficients for $\mu_{\mathrm{B}},\ \Delta T,\ \Delta P$ and the constant term in eq. 114.

For a condition on the enthalpy of a specific phase, H_H^α, see section 5.10.2.

5.10.2 Equi-entropy condition on $S_m^\alpha - S_m^{\mathrm{liq}} = 0$

This section was added in 2019 because I had not implemented entropy conditions and I had forgotten completely how to handle conditions. I needed to add the equi-entropy condition for solid extrapolations. At the same time I discovered some errors in the equations in section 5.10.1 and reallized that some other conditions were not implemented, or did not work, probably because of the errors. But fixing those will be another day.

The equientropy condition means that two phases have the same entropy, $S_m^{\mathrm{bcc}} - S_m^{\mathrm{liquid}} = 0$ but the entropy condition can be implemented having several terms also.

We have for a condition $S_m^\alpha = \tilde{S}_m^\alpha$:

$$S_m^\alpha = \frac{S^\alpha}{N^\alpha} = -\frac{1}{N^\alpha}\frac{\partial G_M^\alpha}{\partial T} \tag{117}$$

$$\Delta S_m^\alpha = -\frac{1}{N^\alpha}\frac{\partial^2 G_M}{\partial T^2} + \frac{1}{(N^\alpha)^2}\frac{\partial G_M^\alpha}{\partial T}\Delta N^\alpha \tag{118}$$

$$N^\alpha = M^\alpha = \sum_{\mathrm{A}} M_{\mathrm{A}}^\alpha = \sum_s a_s \sum_i b_{i,\mathrm{A}} y_{si} \tag{119}$$

$$\Delta N^\alpha = \sum_i \sum_{\mathrm{A}} \frac{\partial M_{\mathrm{A}}^\alpha}{\partial y_i}\Delta y_i^\alpha \tag{120}$$

$$\frac{\partial M_{\mathrm{A}}^\alpha}{\partial y_i} = a_s \sum_j b_{j,\mathrm{A}}\delta_{ij} \tag{121}$$

$$\Delta y_i^\alpha = \sum_{\mathrm{B}} c_{i\mathrm{B}}\mu_{\mathrm{B}} + c_{iT}\Delta T + c_{iP}\Delta P + c_{iG} \tag{122}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

The simplest is to start from eq. 112, ignore the summation over α, replace H_M^α by S_M^α and H_M^α/N^α by S_m^α. The condition do not depend on $\Delta\aleph^\alpha$ either.

$$\frac{1}{N^\alpha}\left(\sum_i \frac{\partial S_M^\alpha}{\partial y_i}\Delta y_i + \frac{\partial S_M^\alpha}{\partial T}\Delta T + \frac{\partial S_M^\alpha}{\partial P}\Delta P - S_m^\alpha \sum_i \sum_{\mathrm{A}} \frac{\partial M_{\mathrm{A}}^\alpha}{\partial y_i}\Delta y_i\right) = S_m^\alpha - \tilde{S}_m^\alpha \tag{123}$$

and then $S_M^\alpha = -\frac{\partial G_M^\alpha}{\partial T}$:

$$\frac{1}{N^\alpha}\left(\sum_i -\frac{\partial^2 G_M^\alpha}{\partial y_i \partial T}\Delta y_i - \frac{\partial^2 G_M^\alpha}{\partial T^2}\Delta T - \frac{\partial^2 G_M^\alpha}{\partial P \partial T}\Delta P - S_m^\alpha \sum_i \sum_{\mathrm{A}} \frac{\partial M_{\mathrm{A}}^\alpha}{\partial y_i}\Delta y_i\right) = S_m^\alpha - \tilde{S}_m^\alpha \tag{124}$$

To handle an expression like $S_m^\alpha - S_M^{\mathrm{liquid}} = 0$ means to calculate eq. 124 for the two phases and take the difference.

In the subroutine calc_dgdyterms I can provide an array f_i which should be multiplied with the appropriate c_{iZ}. For nomallized extensive variables I will have to call this twice, once with

$$f_i = -\frac{\partial^2 G_M^\alpha}{\partial T \partial y_i} \tag{125}$$

and the second time with

$$f_i = \sum_{\mathrm{A}} \frac{\partial M_{\mathrm{A}}^\alpha}{\partial y_i} \tag{126}$$

to obtain the relevant coefficients for eq. 124.

5.11 Condition on constituent fractions

In rare cases one may be interested to set a condition on a constituent fraction of a phase. One particular case is to calculate a 2nd order transition when the fraction of the two constituent fractions of the same component on two different sublattices differ by a small amount:

$$y_{1,\mathrm{A}}^{\mathrm{B2}} - y_{2,\mathrm{A}}^{\mathrm{B2}} = 0.001 \tag{127}$$

This may seem to be a trivial condition but it has to be transformed to fit into the equilibrium matrix where we use expressions which depend on the potentials and the amount of the phases. A condition such as eq. 127 means that $\Delta y_{si}^{\mathrm{B2}} = \Delta y_{tj}^{\mathrm{B2}}$ because the difference between the two constituents is constant. We already have an expresson to express Δy_i, for a phase, eq. 37:

$$\Delta y_i = \sum_{\mathrm{A}} c_{i\mathrm{A}} \mu_{\mathrm{A}} + c_{iG} + c_{iT} \Delta T + c_{iP} \Delta P \tag{128}$$

where c_{iZ} include the inverted phase matrix of α according to eq. 38. For a condition on a constituent constituent fraction of a phase, $y_{si} = \tilde{y}$, we can (at constant P) formulate a row in the equilibrium matrix (note sign of prescribed value):

$$\Delta y_i = \sum_{\mathrm{A}} c_{i\mathrm{A}} \mu_{\mathrm{A}} + c_{iG} + c_{iT} \Delta T = 0 = \tilde{y} - y_i \tag{129}$$

recalling eq. 38 (note the sum over j is over all constituents):

$$\begin{aligned} c_{iG} &= -\sum_j e_{ij} \frac{\partial G_M}{\partial y_j} \\ c_{iT} &= -\sum_j e_{ij} \frac{\partial^2 G_M}{\partial T \partial y_j} \\ c_{i\mathrm{A}} &= \sum_j e_{ij} \frac{\partial M_{\mathrm{A}}}{\partial y_j} \end{aligned} \tag{130}$$

we get:

$$\sum_{\mathrm{A}} \sum_j e_{ij} \frac{\partial M_{\mathrm{A}}}{\partial y_j} \mu_{\mathrm{A}} - \sum_j e_{ij} \frac{\partial^2 G_M}{\partial T \partial y_j} \Delta T = \sum_j e_{ij} \frac{\partial G_M}{\partial y_j} + \tilde{y} - y_i \tag{131}$$

provided (?the value of $y_i = \tilde{y}$ has been set before the calculation of e_{ij}?) and P is constant (compare with eq. 52 for a condition on the amount of a component).

With a condition as:

$$\sum_i f_i y_i = d \tag{132}$$

the equation in the equilibrium matrix will be:

$$\sum_i f_i \Delta y_i = d - \sum_i f_i y_i \tag{133}$$

where each Δy_i is calculated according to eq. 128.

(134)

5.12 Chemical potentials as conditions

In most of the above cases T and P have been fixed and in one example, section 5.4.5, the chemical potential of a component was fixed. We cannot have all conditions as potentials because then we are trying to minimize the Gibbs-Duhem relation which is always zero. At least one condition must be an extensive property.

A potential set to a fixed value means we have one variable less in the equilibrium matrix and the coefficient of the chemical potential multiplied with the fixed value is moved to the right hand side. If there is a relation between two potentials that must be added as an equation (not yet implemented).

5.12.1 Conditions on a mole fractions and a chemical potential

In section 5.4.5 the fixed chemical potential was combined with the total amount of one component in the system. If we instead have a condition on a normallized state variable, like a mole fractions, this creates a slightly more rather complex equation for the mole fraction as the coefficients for the fixed chemical potential must be moved to the right hand side. The mole fraction, $x_\mathrm{A} = \frac{M_\mathrm{A}}{N}$ and we evaluate the terms needed for eq. 103 starting from eq. 50 for each phase α at fixed T and P:

$$\begin{aligned}
\Delta M^\alpha_\mathrm{A} &= \sum_\mathrm{B} \left(\sum_i \sum_j e_{ij} \frac{\partial M^\alpha_\mathrm{A}}{\partial y_i} \frac{\partial M^\alpha_\mathrm{B}}{\partial y_j} \right) \mu_\mathrm{B} - \sum_i \sum_j e_{ij} \frac{\partial M^\alpha_\mathrm{A}}{\partial y_i} \frac{\partial G^\alpha_M}{\partial y_j} \\
&= \sum_\mathrm{B} c_\mathrm{AB} \mu_\mathrm{B} - d_\mathrm{A}
\end{aligned} \tag{135}$$

where the sum over B is for all components. For clearity some indices and superscripts have been suppressed, for example that e_{ij}, c_AB as well as y_i etc. of course depend on the phase. For simplicity the second row of eq. 135 will be used below.

For the normallizing property, the total number of moles according to eq. 111, at constant T and P, is:

$$\begin{aligned}
\Delta N &= \sum_\alpha \sum_\mathrm{A} M^\alpha_\mathrm{A} \Delta \aleph^\alpha + \sum_\alpha \aleph^\alpha \sum_\mathrm{A} \Delta M^\alpha_\mathrm{A} \\
&= \sum_\alpha \sum_\mathrm{A} M^\alpha_\mathrm{A} \Delta \aleph^\alpha + \sum_\alpha \aleph^\alpha \sum_\mathrm{A} (\sum_\mathrm{B} c_\mathrm{AB} \mu_\mathrm{B} - d_\mathrm{A})
\end{aligned} \tag{136}$$

These finite differences can be inserted in eq. 103:

$$\begin{aligned}
\Delta x_{\mathrm{A}} &= \tilde{x}_{\mathrm{A}} - x_{\mathrm{A}} = 0 = \frac{1}{N}(\sum_{\alpha} \aleph^{\alpha} \Delta M_{\mathrm{A}}^{\alpha} + \sum_{\alpha} M_{\mathrm{A}}^{\alpha} \Delta \aleph^{\alpha}) - \frac{x_{\mathrm{A}}}{N} \Delta N \\
&= \frac{1}{N}(\sum_{\alpha} \aleph^{\alpha} \Delta M_{\mathrm{A}}^{\alpha} + \sum_{\alpha} M_{\mathrm{A}}^{\alpha} \Delta \aleph^{\alpha}) - \frac{x_{\mathrm{A}}}{N}(\sum_{\alpha} \sum_{\mathrm{B}} M_{\mathrm{B}}^{\alpha} \Delta \aleph^{\alpha} + \sum_{\alpha} \aleph^{\alpha} \sum_{\mathrm{B}} \Delta M_{\mathrm{B}}^{\alpha}) \\
&= \frac{1}{N} \sum_{\alpha} (M_{\mathrm{A}}^{\alpha} - x_{\mathrm{A}} \sum_{\mathrm{B}} M_{\mathrm{B}}^{\alpha}) \Delta \aleph^{\alpha} + \frac{1}{N} \sum_{\alpha} (\Delta M_{\mathrm{A}}^{\alpha} - x_{\mathrm{A}} \sum_{\mathrm{B}} \Delta M_{\mathrm{B}}^{\alpha}) \aleph^{\alpha} \\
&= \frac{1}{N} \sum_{\alpha} (M_{\mathrm{A}}^{\alpha} - x_{\mathrm{A}} \sum_{\mathrm{B}} M_{\mathrm{B}}^{\alpha}) \Delta \aleph^{\alpha} + \frac{1}{N} \sum_{\alpha} \aleph^{\alpha} \sum_{\mathrm{C}} (c_{\mathrm{AC}} - x_{\mathrm{A}} \sum_{\mathrm{B}} c_{\mathrm{BC}}) \mu_{\mathrm{C}} - \\
&\quad \frac{1}{N} \sum_{\alpha} \aleph^{\alpha} (d_{\mathrm{A}} - x_{\mathrm{A}} \sum_{\mathrm{B}} d_{\mathrm{B}})
\end{aligned} \tag{137}$$

In the last line we have rearranged the terms so we can see the coefficients for the independent variables $\Delta\aleph^{\alpha}$ and μ_{C}. The constant terms will be on the right hand side of the equation and if any chemical potential is fixed that will also be moved the the right hand side (both with changed sign). If T or P are variable there will be coefficients for them also but they are trivial. To make the equation a little easier to understand we introduced in eq. 135:

$$c_{\mathrm{AB}} = \sum_i \sum_j e_{ij} \frac{\partial M_{\mathrm{A}}^{\alpha}}{\partial y_i} \frac{\partial M_{\mathrm{B}}^{\alpha}}{\partial y_j} \tag{138}$$

$$d_{\mathrm{A}} = \sum_i \sum_j e_{ij} \frac{\partial M_{\mathrm{A}}^{\alpha}}{\partial y_i} \frac{\partial G_M^{\alpha}}{\partial y_j} \tag{139}$$

where e_{ij} are terms from the inverted phase matrix. We have not specified the phase superscripts whereever they are obvious.

As I have problem calculating with conditions of mass fractions together with a fixed chemical potential I try to derive the equation for fixed mass fraction also. Using f_{A} for the mass of one mole of component A and B_{A} is the mass of A in the phase and w_{A} is the mass fraction:

$$B_{\mathrm{A}} = M_{\mathrm{A}} f_{\mathrm{A}} \tag{140}$$

$$w_{\mathrm{A}} = \frac{B_{\mathrm{A}}}{\sum_{\mathrm{C}} B_{\mathrm{C}}} = \frac{B_{\mathrm{A}}}{B} \tag{141}$$

we get

$$\Delta B = \sum_{\alpha} \sum_{\mathrm{A}} M_{\mathrm{A}}^{\alpha} f_{\mathrm{A}} \Delta \aleph^{\alpha} + \sum_{\alpha} \aleph^{\alpha} \sum_{\mathrm{A}} (\sum_{\mathrm{C}} c_{\mathrm{AC}} + \cdots \tag{142}$$

UNFINISHED

5.13 Special treatment of the ionic liquid model

The two-sublattice partially ionic liquid model is a unified model for liquids with or without tendency for ionization. It is derived and explained in detail in [85Hil]. In this model the

short range ordering is in fact modeled as long range ordering using two different sites for cations and anions. However, the sites do not represent fixed positions in space but just a way to calculate the configurational entropy with separate mixing of anions and cations.

The notation is

$$(C^{+\nu_C})_P(D^{-\nu_D}, \mathrm{Va}, N)_Q \tag{143}$$

where $C^{+\nu_C}$ represent cations with charge $+\nu_C$, $D^{-\nu_D}$ represent anions with charge $-\nu_D$, Va represent hypothetical vacancies with an induced charge $-Q$ and N represent neutral constituents. P is used with a new meaning here as P and Q are site ratios which are equal to the average charge on the opposite sublattice:

$$P = \sum_A y_A \nu_A + y_{\mathrm{Va}} Q \tag{144}$$

$$Q = \sum_C y_C \nu_C \tag{145}$$

An example of this model is the liquid Cu-Fe-S modeled as

$$(Fe^{+2}, Cu^{+1})_P(S^{-2}, \mathrm{Va}, S)_Q \tag{146}$$

where the binary metallic liquid Cu-Fe is described with just Va on the second sublattice to compensate for the charge. The pure sulfur liquid has just neutral S. Note that in that case $P = 0$. There is short range ordering in the liquid at the compositions FeS and Cu_2S when the second sublattice has mainly S^2 and the first a single metallic ion.

The mass balance equation is the same as for the CEF model:

$$M_{\mathrm{A}} = P \sum_i b_{i\mathrm{A}} y_i + Q(\sum_j b_{j\mathrm{A}} y_j + \sum_k b_{k\mathrm{A}} y_k) \tag{147}$$

where the $b_{i\mathrm{A}}, b_{j\mathrm{A}}$ and $b_{k\mathrm{A}}$ represent the stoichiometric factor of element A in cations, anions and neutrals respectively and all y_i the fractions of the constituent i. The derivative of this is slightly more complicated than for the CEF model as P and Q are not constant:

$$dM_{\mathrm{A}} = dP \sum_i b_{Ai} y_i + P \sum_i b_{Ai} dy_i + dQ(\sum_j b_{Aj} y_j + \sum_k b_{Ak} y_k) + Q(\sum_k b_{Ak} dy_k + \sum_j b_{Aj} dy_j) \tag{148}$$

where

$$dP = \sum_A dy_A \nu_A + dy_{\mathrm{Va}} Q + y_{\mathrm{Va}} dQ \tag{149}$$

$$dQ = \sum_C dy_C \nu_C \tag{150}$$

The partial derivatives $\frac{\partial M_{\mathrm{A}}}{\partial y_i}$ are no longer constants which makes the equations for the equilibrium calculation slightly more complicated to implement.

6 Some useful derivatives

Many partial derivatives of the Gibbs energy are useful in particular cases. The first derivatives are discussed in section 3.1. Several second derivatives are also important like the heat capacity, thermal expansion and the compressibility. The latter three are not available directly after a calculation but can be obtained using the technique shown in the next section. The second derivatives with respect to composition variables is also discussed in section 6.2

6.1 Calculating derivatives using the result of an equilibrium calculation, the dot derivative

After an equilibrium calculation it is possible to obtain additional properties using the calculated values of G and its first and second derivatives. This type of calculations was first implemented in Thermo-Calc by Bo Jansson but never documented.

6.1.1 Calculating heat capacity

The heat capacity is not directly available after an equilibrium calculation but by combining the values of appropriate coefficients in eq. 38 we can calculate this without making any new equilibrium calculation with a small step in T. The heat capacity at constant P is defined as

$$C_P = \left(\frac{\partial H}{\partial T}\right)_{P,N_i} \tag{151}$$

We must derive a way to obtain this from the Gibbs energy, G, for a system at equilibrium distributed over a set of stable phases α, i.e.

$$\begin{aligned} G(T,P,N_i) &= \sum_\alpha \aleph^\alpha G_M^\alpha(T,P,y_{js}) && (152)\\ N_i &= \sum_\alpha \aleph^\alpha N_i^\alpha && (153)\\ N_i^\alpha &= \sum_s a_s^\alpha \sum_j b_{ij} y_{js}^\alpha && (154) \end{aligned}$$

where N_i is the amount in moles of component i, $\aleph^\alpha$ is the amount of formula units of phase α and G_M^α is the Gibbs energy of α for one formula unit.

The constitution of the α phase is modeled using constituent fractions y_{js} for fraction of constituent j on sublattice s, a_s is the number of sites on sublattice s and b_{ij} is the stoichiometric factor of component i in constituent j.

The enthalpy, H, can be calculated from the Gibbs energy as:

$$\begin{aligned} G &= H - TS = H + T\left(\frac{\partial G}{\partial T}\right)_{P,N_i} && (155)\\ H &= G - T\left(\frac{\partial G}{\partial T}\right)_{P,N_i} && (156) \end{aligned}$$

From the model we cannot calculate a derivative with respect to constant N_i so the derivative of G must be expanded as:

$$\begin{aligned}\left(\frac{\partial G}{\partial T}\right)_{P,N_i} &= \sum_\alpha \frac{\partial \aleph^\alpha}{\partial T} G_M^\alpha + \sum_\alpha \aleph^\alpha \left(\frac{\partial G_M^\alpha}{\partial T}\right)_{P,N_i} \\ &= \sum_\alpha \frac{\partial \aleph^\alpha}{\partial T} G_M^\alpha + \sum_\alpha \aleph^\alpha \left[\left(\frac{\partial G_M^\alpha}{\partial T}\right)_{P,y_{js}} + \sum_{js} \left(\frac{\partial G_M^\alpha}{\partial y_{js}}\right)_{P,y_{kt\neq js}} \frac{\partial y_{js}}{\partial T}\right]\end{aligned} \tag{157}$$

The last equation comes from the fact that the equilibrium is calculated for fixed amounts of the components but the Gibbs energy of the phase depends on the constituent fractions, the term $\left(\frac{\partial G_M^\alpha}{\partial T}\right)_{P,y_{js}}$ is calculated for fixed constituents fractions and the sum over the derivatives with respect to y_{js} takes care of the contribution from the variation of the constitution with the temperature.

The heat capacity finally is defined and calculated as

$$C_P = \left(\frac{\partial H}{\partial T}\right)_{P,N_i} \tag{158}$$

$$\left(\frac{\partial H}{\partial T}\right)_{P,N_i} = \sum_\alpha \frac{\partial \aleph^\alpha}{\partial T} H_M^\alpha + \sum_\alpha \aleph^\alpha \left(\frac{\partial H_M^\alpha}{\partial T}\right)_{P,N_i} \tag{159}$$

Again changing the derivative from constant N_i to constant y_{is} and replacing H by G gives:

$$\begin{aligned}\left(\frac{\partial H}{\partial T}\right)_{P,N_i} &= \sum_\alpha \frac{\partial \aleph^\alpha}{\partial T} H_M^\alpha + \sum_\alpha \aleph^\alpha \frac{\partial}{\partial T}\left[G_M^\alpha - T\left(\frac{\partial G_M^\alpha}{\partial T}\right)_{P,N_i}\right] \\ &= \sum_\alpha \frac{\partial \aleph^\alpha}{\partial T} H_M^\alpha - T\sum_\alpha \aleph^\alpha \frac{\partial}{\partial T}\left[\left(\frac{\partial G_M^\alpha}{\partial T}\right)_{P,N_i}\right] \\ &= \sum_\alpha \frac{\partial \aleph^\alpha}{\partial T} H_M^\alpha - T\sum_\alpha \aleph^\alpha \frac{\partial}{\partial T}\left[\left(\frac{\partial G_M^\alpha}{\partial T}\right)_{P,y_{js}} + \sum_{js}\left(\frac{\partial G_M^\alpha}{\partial y_{js}}\right)_{P,y_{k\neq j}} \frac{\partial y_{js}}{\partial T}\right] \\ &= \sum_\alpha \frac{\partial \aleph^\alpha}{\partial T} H_M^\alpha - T\sum_\alpha \aleph^\alpha \left[\left(\frac{\partial^2 G_M^\alpha}{\partial T^2}\right)_{P,y_{js}} + \sum_{js}\left(\frac{\partial^2 G_M^\alpha}{\partial T \partial y_{js}}\right)_{P,y_{k\neq j}} \frac{\partial y_{js}}{\partial T}\right]\end{aligned} \tag{160}$$

For a small change in T, we replace the derivatives of $\aleph$ and y_{js} by finite differences. We also drop the subscript s as the summation over j is for all constituents.

$$\left(\frac{\partial H}{\partial T}\right)_{P,N_i} = \sum_\alpha \Delta\aleph^\alpha H_M^\alpha - T\sum_\alpha \aleph^\alpha \left[\left(\frac{\partial^2 G_M^\alpha}{\partial T^2}\right)_{P,y_j} + \sum_j \left(\frac{\partial^2 G_M^\alpha}{\partial T \partial y_j}\right)_{P,y_{k\neq j}} \Delta y_j\right] \tag{161}$$

In order to find the appropriate values for $\Delta\aleph$ and Δy_j we have to calculate one iteration of the equilibrium matrix with an extra equation for ΔT as an additional variable. In order

to be able to add such an equation the temperature must be a condition in the original equilibrium calculation.

As an example we take the equilibrium matrix for a binary system with conditions on T, P and the amount of the components and with a single stable phase in eq. 53.

In this matrix we can add a line representing the variable ΔT and for all previous lines in the matrix we add a column representing the derivate with respect to T. The system of equations for the same case would be:

$$\begin{pmatrix} M_{\rm A} & M_{\rm B} & 0 & s_{14} \\ \aleph \sum_i \frac{\partial M_{\rm A}}{\partial y_i} c_{i{\rm A}} & \aleph \sum_i \frac{\partial M_{\rm A}}{\partial y_i} c_{i{\rm B}} & M_{\rm A} & s_{24} \\ \aleph \sum_i \frac{\partial M_{\rm B}}{\partial y_i} c_{i{\rm A}} & \aleph \sum_i \frac{\partial M_{\rm B}}{\partial y_i} c_{i{\rm B}} & M_{\rm B} & s_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta\mu_{\rm A} \\ \Delta\mu_{\rm B} \\ \Delta\aleph \\ \Delta T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad (162)$$

with the terms in the 4th column calculated as:

$$s_{14} = \frac{\partial G_M}{\partial T} \quad (163)$$

$$s_{24} = \aleph \sum_i \frac{\partial M_{\rm A}}{\partial y_i} c_{iT} \quad (164)$$

$$s_{34} = \aleph \sum_i \frac{\partial M_{\rm B}}{\partial y_i} c_{iT} \quad (165)$$

The right hand side is all zeros, except for ΔT, as we are calculating a difference.

We do not iterate this system of equations, just solve it once to obtain the values of $\Delta\mu_{\rm A}, \Delta\mu_{\rm A}$ and $\Delta\aleph$, the value of ΔT is unity of course. The values of $\Delta\mu_{\rm A}$ and $\Delta\mu_{\rm B}$ are related to this change and the value of Δy_i can be calculated from eq. 37 using these $\Delta\mu_{\rm A}$ and $\Delta\mu_{\rm B}$:

$$\Delta y_i = c_{iT}\Delta T + \sum_{\rm A} c_{i{\rm A}}\ \Delta\mu_{\rm A} \quad (166)$$

In this way we obtain a heat capacity that corresponds to the enthalpy change with an increase in temperature of one degree. This also includes the latent heat if there are several stable phases and contributions from internal degrees of ordering.

6.1.2 The liquidus slope

Another property that is useful for simulating phase transformations is the slope of the solubility lines. For example in order to calculate the slope of the liquidus versus T in the direction of a component A we can set the appropriate conditions together with specifying a solid phase as fixed, and no condition on the temperature, and calculate the equilibrium as outlined in the previous sections.

6.2 The Darken stability matrix and related functions

The Darken stability function for a phase α contains the derivatives of the partial Gibbs energies of a phase with respect to all constituents. For a substititional model this gives:

$$M = \begin{pmatrix} \frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm A}} & \frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm B}} & \dots \\ \frac{\partial G^\alpha_{\rm B}}{\partial x_{\rm A}} & \frac{\partial G^\alpha_{\rm B}}{\partial x_{\rm B}} & \dots \\ \dots & & \end{pmatrix} \tag{167}$$

For a binary system the determinant of the stability matrix is simply

$$\det(M) = \frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm A}}\frac{\partial G^\alpha_{\rm B}}{\partial x_{\rm B}} - \frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm B}}\frac{\partial G^\alpha_{\rm B}}{\partial x_{\rm A}} \tag{168}$$

and if this is negative the phase composition is inside a spinodal and the phase is unstable. For a multicomponent system we can calculate the eigenvalues of the Darken stability matrix and if there is at least one negative eigenvalue the phase is unstable.

For kinetic simulations these derivatives are also needed to convert from mobilities to diffusion coefficients. As a phase undergoing a transformation may be far from its equilibrium state it must be calculated using the local values of T, P and compositions for each phase.

We know that the partial Gibbs energy for a substitutional phase α can be calculated as given in eq. 18:

$$G^\alpha_{\rm A} = G^\alpha_M + \left(\frac{\partial G^\alpha_M}{\partial x_{\rm A}}\right)_{T,P,x_{\rm C\neq A}} - \sum_{\rm B} x_{\rm B}\left(\frac{\partial G^\alpha_M}{\partial x_{\rm B}}\right)_{T,P,x_{\rm C\neq B}} \tag{169}$$

where the summation over B is for all constituents.

For a phase α with a substitutional model the partial derivative of $G^\alpha_{\rm A}$ can be calculated from the current composition of α as:

$$\frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm B}} = \frac{1}{N}\left[\left(\frac{\partial^2 G^\alpha_M}{\partial x_{\rm A}\partial x_{\rm B}}\right)_{T,P,x_{\rm C\neq A,B}} - \sum_{\rm C} x_{\rm C}\left(\left(\frac{\partial^2 G^\alpha_M}{\partial x_{\rm A}\partial x_{\rm C}}\right)_{T,P,x_{\rm D\neq A,C}} + \left(\frac{\partial^2 G^\alpha_M}{\partial x_{\rm B}\partial x_{\rm C}}\right)_{T,P,x_{\rm D\neq B,C}}\right) + \sum_{\rm C}\sum_{\rm D} x_{\rm C}x_{\rm D}\left(\frac{\partial^2 G^\alpha_M}{\partial x_{\rm C}\partial x_{\rm D}}\right)_{T,P,x_{\rm E\neq C,D}}\right] \tag{170}$$

where the summation over C and D are for all components. It is quite interesting that this is symmetric:

$$\frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm B}} = \frac{\partial G^\alpha_{\rm B}}{\partial x_{\rm A}} \tag{171}$$

However, eq. 170 is not obtained if we simply calculate the derivative of eq. 169 with respect of $x_{\rm B}$ but we must calculate it by taking the derivative with respect to the amount, $N_{\rm B}$. As it is easy to make errors the derivation is given here (without specifying the variables kept constant):

$$\frac{\partial G^\alpha_{\rm A}}{\partial N_{\rm B}} = \sum_C \frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm C}}\frac{\partial x_{\rm C}}{\partial N_{\rm B}} \tag{172}$$

$$x_{\rm C} = \frac{N_{\rm C}}{\sum_{\rm D} N_{\rm D}} = \frac{N_{\rm C}}{N} \tag{173}$$

$$\frac{\partial x_{\rm C}}{\partial N_{\rm B}} = \frac{\delta_{\rm BC} - x_{\rm C}}{N} \tag{174}$$

where $\delta_{\rm BC}$ is unity if B and C are the same component, otherwise zero. Using these relations we get:

$$\begin{aligned}
\frac{\partial G^\alpha_{\rm A}}{\partial N_{\rm B}} &= \sum_C \frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm C}} \frac{\delta_{\rm BC}-x_{\rm C}}{N} \\
&= \left(\sum_{\rm C} \frac{\partial G^\alpha_M}{\partial x_{\rm C}} + \frac{\partial^2 G^\alpha_M}{\partial x_{\rm A}\partial x_{\rm C}} - \frac{\partial G^\alpha_M}{\partial x_{\rm C}} - \sum_{\rm D} x_{\rm D} \frac{\partial^2 G^\alpha_M}{\partial x_{\rm D}\partial x_{\rm C}}\right) \frac{\delta_{\rm BC}-x_{\rm C}}{N} \\
&= \frac{1}{N}\left(\sum_{\rm C} \frac{\partial^2 G^\alpha_M}{\partial x_{\rm A}\partial x_{\rm C}} - \sum_{\rm D} x_{\rm D} \frac{\partial^2 G^\alpha_M}{\partial x_{\rm D}\partial x_{\rm C}}\right)(\delta_{\rm BC}-x_{\rm C}) \\
&= \frac{1}{N}\left(\frac{\partial^2 G^\alpha_M}{\partial x_{\rm A}\partial x_{\rm B}} - \sum_{\rm D} x_{\rm D} \frac{\partial^2 G^\alpha_M}{\partial x_{\rm D}\partial x_{\rm B}} - \sum_{\rm C} x_{\rm C}\left(\frac{\partial^2 G^\alpha_M}{\partial x_{\rm A}\partial x_{\rm C}} - \sum_{\rm D} x_{\rm D} \frac{\partial^2 G^\alpha_M}{\partial x_{\rm D}\partial x_{\rm B}}\right)\right) \\
&= \frac{1}{N}\left(\frac{\partial^2 G^\alpha_M}{\partial x_{\rm A}\partial x_{\rm B}} - \sum_{\rm C} x_{\rm C}\left(\frac{\partial^2 G^\alpha_M}{\partial x_{\rm C}\partial x_{\rm B}} + \frac{\partial^2 G^\alpha_M}{\partial x_{\rm A}\partial x_{\rm C}}\right) + \sum_{\rm C}\sum_{\rm D} x_{\rm C} x_{\rm D} \frac{\partial^2 G^\alpha_M}{\partial x_{\rm C}\partial x_{\rm D}}\right)
\end{aligned} \tag{175}$$

For a system at constant composition the difference between $\frac{\partial G^\alpha_{\rm A}}{\partial N_{\rm B}}$ or $\frac{\partial G^\alpha_{\rm A}}{\partial x_{\rm B}}$ is probably not very important.

If the phase is modeled with sublattices we can only calculate chemical potentials for the endmembers but it is possible to obtain an equivalent stability matrix.

7 Applications

The HMS package can calculate a single multicomponent equilibrium for many different kinds of conditions and models for the phases. There are three main applications of this described briefly below.

7.1 Calculation of phase diagrams and other diagrams

There is a separate documentation to the package for calculating phase diagrams and property diagrams [SMP]. This application need some additional data structures inside the HMS package as the routines calculating the diagrams need to take generate node points where several lines meet when the set of phases changes. The algorithm desccribed in this document is very suitable for this.

7.2 Assessments of model parameters

The HMS package also contains a subroutine needed for assessing model parameters using experimental and theoretical data, this technique is described in detail in the book by Lukas et al.[07Luk] and in the thesis by Jansson [84Jan]. In an assessment experimental data for various thermodynamic properties such as enthalpies, heat capacities and phase diagram data, are used to determine the parameters in thermodynamic models. The sum of the squares of the differences between experimental data and the corresponding quantity obtained

from an equilibrium calculation using the model parameters is minimized by a least square fitting routine.

$$F(q_j) = \sum_i \left(\frac{x_i^{\text{experimental}} - x_i^{\text{model}}(q_j)}{\rho_i} w_i \right)^2 \tag{176}$$

where the summation is over all experimental values, q_j are the model parameters, $x_i^{\text{experimental}}$ are the ith experimental data for a thermodynamic property, $x_i^{\text{model}}(q_i)$ the same properties obtained from an equilibrium calculation using the model parameters q_i. ρ_i is the experimental uncertainty and w_i is a weight assigned to this information by the assessor.

The least square routine provides the subroutine with a set of variables, q_j and these are transferred by the subroutine to model coefficients in the thermodynamic model package, GTP. The experimental data have been obtained at an equilibrium state and thus the subroutine must make several equilibrium calculations to obtain the same data from the thermodynamic models. The information about the conditions for these equilibria is transferred to the subroutine by a structured datatype defined in GTP. The individually calculated values are returned to the least square routine:

$$F_i(q_j) = (x_i^{\text{experimental}} - x_i^{\text{model}}(q_j)) \frac{w_i}{\rho_i} \tag{177}$$

Inside the least square routine various numerical techniques are used to minimize the sum of squares in eq. 176 by varying the variables q_j.

It is also possible to use inequalities for the experimental data, i.e. setting that a property is larger or smaller than a certain value. Finally one may also prescribe limits of the model parameters.

Eventually there will be a separate documentation of the assessment technique [ASSESS].

7.3 Application Software interface

There is a separate documentation of the use of the HMS minimizer in the OC Application Software Interface [OCASI] and there are two publications [16Sun].

8 Data structures

The verbatim parts of this and the next section are directly extracted from the source code to simplify understanding how to use the subroutines. A very brief explanation of the main parts of the data structures and subroutines in the HMS package is given here.

For each equilibrium calculation a structure called meqrec is created of the type meq_data explained below. In meqrec there is an array with record type meq_phase for each phase to hold intermediate values.

8.1 Data for each phase

For each phase in the system, a structure with these data is created

```
  TYPE meq_phase
! parts of the data in this structure should be in the gtp_equilibrium_data
! it contains phase specific results from various subroutines during
! equilibrium calculation
! iph: phase number
! ics: composition set number
! idim: the dimension of phase matrix,
! ncc: the number of constituents (same as idim??)
! stable: is 1 for a stable phase
! xdone: set to 1 for stoichiometric phases after calculating xmol first time
! dormlink: link to next phase that has temporarily been set dormant
! eec_check for equi-entropy check
! phtupix phase tuple index
     integer iph,ics,idim,stable,ncc,xdone,dormlink,eeccheck,phtupix
! value of phase status (-1,0=ent, 1=stable, 2=fix, -2=dorm, -3=sus, -4 hidden)
     integer phasestatus
! inverted phase matrix
     double precision, dimension(:,:), allocatable :: invmat
! mole fractions of components and their sum
     double precision, dimension(:), allocatable :: xmol
     double precision :: sumxmol,sumwmol
! Derivatives of moles of component wrt all constituent fractions of the phase
     double precision, dimension(:,:), allocatable :: dxmol
! link to phase_varres record
     TYPE(gtp_phase_varres), pointer :: curd
! value of amount and driving force at previous iteration
     double precision prevam, prevdg
! iteration when phase was added/removed
     integer itadd, itrem
! chargebal is 1 if external charge balance needed, ionliq<0 unless
! ionic liquid when it is equal to nkl(1)=number of cations
     integer chargebal,ionliq,i2sly(2)
     double precision iliqcharge,yva
```

```
! end specific ionic liquids
  end TYPE meq_phase
```

8.2 Data for the system

The equilibrium calculation starts by filling this structure with data. Each phase that can be stable has an entry in the meq_phase array.

```
  TYPE meq_setup
! one structure of this type is created when an equilibrium calculation
! is started and it holds all global data needed for handling the
! calculation of an equilibrium.  The phase specific data is in meq_phase
! nv: initial guess of number of stable phases
! nphase: total number of phases and composition sets
! nstph: current number of stable phases
! dormlink: is start of list of phases temporarily set dormant
! noofits current number of iterations
! status for various things
! nrel number of elements (components)
! typesofcond: types of conditions, =1 only massbal, =2 any conditions
! nfixmu number of fixed chemical potentials
! nfixph number of conditions representing fix phases
     integer nv,nphase,nstph,dormlink,noofits,status
     integer nrel,typesofcond,maxsph,nfixmu,nfixph
! component numbers of fixed potentials, reference and value
     integer, dimension(:), allocatable :: mufixel
     integer, dimension(:), allocatable :: mufixref
! in this array the mu value as calculated from SER is stored
     double precision, dimension(:), allocatable :: mufixval
! in this array the mu value for user defined reference state is stored
     double precision, dimension(:), allocatable :: mufixvalref
! fix phases and amounts
     integer, dimension(:,:), allocatable :: fixph
     double precision, dimension(:), allocatable :: fixpham
! indices of axis conditions that has been inactivated
!     integer, dimension(:), allocatable :: inactiveaxis
! iphl, icsl: phase and composition sets of intial guess of stable phases
! aphl: initial guess of amount of each stable phase
     integer iphl(maxel+2),icsl(maxel+2)
     double precision aphl(maxel+2)
! stphl: current list of stable phases, value is index in phr array
     integer, dimension(maxel+2) :: stphl
! current values of chemical potentials stored in gtp_equilibrium_data
! if variable T and P these are TRUE, otherwise FALSE
     logical tpindep(2)
! these are the maximum allowed changes in T and P during iterations
```

```
     double precision tpmaxdelta(2)
! individual phase information
     type(meq_phase), dimension(:), allocatable :: phr
! this is used for EEC, pointer to liquid phr record and highest liquid entropy
     type(meq_phase), pointer :: pmiliq
     double precision seecliq
! information about conditions should be stored here.  Note that conditions
! may change during STEP and MAP
  end TYPE meq_setup
```

8.3 Data needed for applications like STEP and MAP calculations

When calculating phase diagrams or property diagrams one must have control of when a new phase wants to be stable or a stable phase will disappear. This data structure helps to transfer such information and in addition is used to specify the phase kept fixed with zero amount along the lines in the phase diagram.

```
  TYPE map_fixph
! provides information about phase sets for each line during mapping
     integer nfixph,nstabph,status
     type(gtp_phasetuple), dimension(:), allocatable :: fixph
     type(gtp_phasetuple), dimension(:), allocatable :: stableph
! most likely some of these variables are redundant stable_phr added 2020.03.05
     integer, dimension(:), allocatable :: stable_phr
     double precision, dimension(:), allocatable :: stablepham
! new 180814 to have nonzero fix phase amounts  ... not yet used
     double precision, dimension(:), allocatable :: fixphamap
  end TYPE map_fixph
```

8.4 Data needed for the assessment application

These are data needed for calculating the Relative Standarad Deviation (RSD) during assessments and some other things for other purposes.

```
! This is for returning the calculated value of an experimental property
! as we need an array to store the calculated values of the experimental
! properties in order to calculate the Relative Standarad Deviation (RSD)
  double precision, allocatable, dimension(:) :: calcexp
! We cannot have EEC variabler here as it does bot work in parallel
! this is for EEC test
!  type(meq_phase), pointer :: pmiliq
! if several liquids check for largest S
!  type(meq_phase), pointer :: pmiliqsave
!  double precision eecliqentropy
! this is set TRUE when entering meq_onephase and false after one solid checked?
```

```
! it is now check for EEC
!  logical eecextrapol
! This is an (failed) attempt to limit Delta-T when having condition on y
  logical ycondTlimit
  double precision deltatycond
! TZERO and PARAEQUIL calculation need these (CANNOT BE USED IN PARALLEL)
  type(gtp_equilibrium_data), pointer :: tzceq
  type(gtp_condition), pointer :: tzcond
  type(gtp_state_variable), target :: musvr,xsvr
  integer tzph1,tzph2
! To prevent calculating a dot derivative at a given equilibrium
  integer :: special_circumstances=0
```

9 Software documentation

The verbatim parts of this section are directly extracted from the source code.

9.1 Top level calculation routines

There are three routines for equilibrium calculation depending on the context.

9.1.1 The simplest ones, for single equilibrium calculation

This subroutine organizes the calculation of a single equilibrium. The user must have stored all data in the GTP package including all conditions. This is found via the pointer ceq. Each such datastructure is independent and it should be possible to run several of these in parallel.

If mode is nonzero it will call a grid minimizer whenever possible to find a set of stable phases. It does some clean up and copy of results at the end if the calculation is successful.

The difference between calceq2 and 3 is that the latter does not write anything, except possible error messages.

```
  subroutine calceq2(mode,ceq)
! calculates the equilibrium for the given set of conditions
! mode=0 means no global minimization
! ceq is a datastructure with all relevant thermodynamic data
    implicit none
    integer mode
    TYPE(gtp_equilibrium_data), pointer :: ceq
 ======================
  subroutine calceq3(mode,confirm,ceq)
! calculates the equilibrium for the given set of conditions
! mode=0 means no global minimization
! confirm is TRUE if output of CPU time
! ceq is a datastructure with all relevant thermodynamic data
    implicit none
    integer mode
```

```
    logical confirm
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.1.2 Equilibrium calculations during step/map calculations

Calling this routine makes it possible to retrieve more information when the set of stable phases changes etc. It is used in STEP/MAP when creating new lines in a diagram.

```
  subroutine calceq7(mode,meqrec,mapfix,ceq)
! calculates the equilibrium for the given set of conditions
! mode=0 means no global minimization
! mode=-1 means used during step/map, no gridmin and do not deallocate phr
! ceq is a datastructure with all relevant thermodynamic data
! calling this routine instead of calceq2 makes it possible to extract
! additional information about the equilibrium from meqrec.
! Meqrec is also used for calculation of derivatives of state vatiables
    implicit none
    integer mode
    TYPE(meq_setup), pointer :: meqrec
    type(map_fixph), allocatable :: mapfix
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.2 Subroutine to change the set of stable phases

This subroutine will call meq_sameset with a list of stable phases. The meq_sameset subroutine will return to this if there is a phase with negative amount to be removed or a phase with positive driving force to be added.

```
  subroutine meq_phaseset(meqrec,formap,mapfix,ceq)
! this subroutine can change the set of stable phase and their amounts
! and constitutions until equilibrium is found for the current conditions.
    implicit none
    TYPE(meq_setup) :: meqrec
    type(map_fixph), allocatable :: mapfix
    TYPE(gtp_equilibrium_data), pointer :: ceq
    logical formap
```

9.3 Iterations with same set of stable phases

This subroutine will iterate updating the chemical potentials, the phase amounts and constitution of all phases as long as the set of stable phases does not change. Several criteria are used to test if the calculation have converged.

```
  recursive subroutine meq_sameset(irem,iadd,mapx,meqrec,phr,inmap,ceq)
! iterate until phase set change, converged or error (incl too many its)
! iadd = -1 indicates called from calculating a sequence of equilibria
! mapx is used when calling meq_sameset from step/map
```

```
    implicit none
    integer irem,iadd,inmap,mapx
    TYPE(meq_setup) :: meqrec
    TYPE(meq_phase), dimension(*), target :: phr
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.4 Formulating the equilibrium matrix for a single phase at known T, P and overall composition

This routine calculates the internal equilibrium for a phase with internal degrees of freedom when T, P and the composition is known. For example the equilibrium in a gas phase or the distribution of components on different sublattices.

```
  subroutine setup_comp2cons(meqrec,phr,nz1,smat,tval,xknown,converged,ceq)
! calculate internal equilibrium in a phase for given overall composition
! meqrec and phr contains data for phases, nz1 is dimension of equlibrium
! matrix, smat is the equilibrium matrix, tval is fixed T and P
! xknown is the overall composition
    TYPE(meq_setup) :: meqrec
    TYPE(meq_phase), dimension(*), target :: phr
    double precision smat(nz1,*),tval(*),xknown(*)
    integer nz1,converged
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.5 Formulating the equilibrium matrix for a general case

This routine sets up the equilibrium matrix depending on the set of stable phases and the conditions set by the user.

```
  subroutine setup_equilmatrix(meqrec,phr,nz1,smat,tcol,pcol,&
       dncol,converged,ceq)
! handels external conditions on extensive variables in the equil matrix
! meqrec and phr contains data for phases, nz1 is dimension of equlibrium
! matrix, smat is the equilibrium matrix, tcol and pcol are columns for
! variable T or P, dncol is the column with phase amount variables.
! converged is used to indicate calling routine and set if not converged
! external variable.
    TYPE(meq_setup) :: meqrec
    TYPE(meq_phase), dimension(*), target :: phr
    double precision smat(nz1,nz1+1)
    integer nz1,tcol,pcol,converged,dncol
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.6 Routine to calculate the inverse phase matrix

At each iteration the inverted phase matrix is calculated for all phases. Additionally a lot of data needed for the mass balance equations are calculated and stored here.

```
  subroutine meq_onephase(meqrec,pmi,ceq)
! this subroutine calculates new constituent fractions for a phase iph+ics
! with given T, P and chemical potentials for the components
! For ionic liquids the sites on the sublattices varies with composition
! THIS IS A FIRST VERSION WITHOUT ANY TRICKS FOR SPEED
! this will check if EEC set and modify G for solid phases with higher entropy
! pmi is pointer to a record in meq_phase, local to this thread
! than the liquid
    implicit none
    TYPE(meq_phase), pointer :: pmi
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(meq_setup) :: meqrec
```

9.7 Some utility routines

These are called when necessary for each iteration.

9.7.1 Correction of of second derivatives for the ionic liquid model

In the ionic liquid model the size of the formula unit of the phase varies with the composition. This requires some extra care when calculating how the amount of the components varies with the constitution.

```
  subroutine corriliq_d2gdyidyj(nkl,knr,curmu,pmi,ncc,nd1,pmat,ceq)
! correction of d2G/dy1dy2 for ionic liquid because the formula unit is
! not fixed.  This contributes ONLY to the second derivaties of G and
! is not really part of the model itself, only needed when minimizing G
    implicit none
    type(gtp_equilibrium_data), pointer :: ceq
    TYPE(meq_phase), pointer :: pmi
    integer ncc,nd1,nkl(*),knr(*)
    double precision curmu(*),pmat(nd1,*)
```

9.7.2 Test of same composition

When there are two or more composition sets of the same phase needed to handle miscibility gaps, these may sometimes converge to the same composition and try to become stable. This subroutine is needed to prevent having two composition sets with the same composition stable.

```
  logical function same_composition(jj,phr,meqrec,ceq,dgm)
! returns .TRUE. if phase phr(jj) has almost exactly the same composition
! as another composition set of the same phase that is stable
! dgm just for debug output
! ============================================================
! The composition of the phases are compared as ordered phases one can have
! the same constitution but distributed on different sets of sublattices ....
```

```
! ============================================================
    implicit none
    integer jj
    double precision dgm
    TYPE(meq_phase), dimension(*) :: phr
    TYPE(meq_setup) :: meqrec
    TYPE(gtp_equilibrium_data), pointer :: ceq

  recursive subroutine two_stoich_same_comp(irem,iadd,mapx,meqrec,inmap,ceq)
! we have found two  phases stable with same composition
! ONLY USED WHEN MAPPING with tie-lines in plane
! ceq is equilibrium record
    implicit none
    integer irem,iadd,inmap,mapx
    type(meq_setup) :: meqrec
    type(gtp_equilibrium_data), pointer :: ceq
```

9.8 The coefficients in the Δy equation

These subroutines return the terms in the Δy expression, eq. 37. There are several of them as I have not had time to make them uniform. Some are used during the equilibrium calculation, others when calculating "dot derivatives".

When profiling the OC software for a case with 15 elements and 150 solution phases it was found that almost 20% of the CPU time was spent in these routines. They are called to calculate the terms in Δy_{si}^{α} in eq. 37 for each condition, and with 15 elements with 15 conditions on $N_{\rm a}$ (or equivalent) it means they are called at least 225 times for each stable phases at each iteration. In an attempt to reduce the CPU time it was decided to save some intermediate summations which correspond to the terms:

$$c_{iA}^{\alpha} \;=\; \sum_j \frac{\partial M_{\rm A}^{\alpha}}{\partial y_j} e_{ij}^{\alpha} \tag{178}$$

where $c_{i\rm A}$ is a term in eq. 37 and e_{ij} is the inverted phase matrix from eq. 34. $c_{i\rm A}$ are saved in a new array, "invsaved" allocated in the phase_varres record for each phase and composition set (in order to allow parallelization).

At a new iteration the $c_{i\rm A}^{\alpha}$ for all elements A is calculated the first time it is needed and saved. For all remaining cases it is needed during that iteration the saved value is used. This modification reduced the CPU time inside this routine to a very small number.

At present there are several versions of this routine to find the best method and calc_dgdyterms1X is the one normally used. But the calc_dgdytemsh was recently introduced to handle conditions on enthalpy. It is more flexible and will probably replace the others eventually.

```
  subroutine calc_dgdyterms1(nrel,ia,tpindep,mamu,mag,mat,map,pmi,&
       curmux,noofits)
! THIS SUBROUTINE IS NO LONGER USED!!  Cannot be used in parallel
! any change must also be made in subroutine calc_dyterms2 and calc_dgdytermsh
! calculate the terms in the deltay expression for amounts of component ia
```

```
!
! DM_A = \sum_B mu_B*MAMU(B) - MAG - MAT*dt - MAP*dp
!
! where MAMU=\sum_i dM_A/dy_i*\sum_j invmat(i,j)*dM_B/dy_j
!       c_iB=\sum_j invmat(i,j)*dM_B/dy_j etc etc
!
! it may not be very efficient but first get it right ....
! tpindep(1) is TRUE if T variable, tpindep(2) is TRUE if P are variable
!
! >>> ATTENTION, there is a FASTER VERSION calc_dgdyterms1X
! >>> ATTENTION not safe for parallelization ....
!
    implicit none
    integer ia,nrel,noofits
    logical tpindep(2)
    double precision, dimension(*) :: mamu
    double precision mag,mat,map
    double precision curmux(*)
! pmi is the phase data record for this phase
    type(meq_phase), pointer :: pmi
-------------------------------------
  subroutine calc_dgdyterms1X(nrel,ia,tpindep,mamu,mag,mat,map,pmi,noofits)
! THIS SUBROUTINE using allocatable arrays in phase_varres!!
! any change must also be made in subroutine calc_dyterms2 and calc_dgdytermsh
! calculate the terms in the deltay expression for amounts of component ia
!
! DM_A = \sum_B mu_B*MAMU(B) - MAG - MAT*dt - MAP*dp
!
! where MAMU=\sum_i dM_A/dy_i*\sum_j invmat(i,j)*dM_B/dy_j
!       c_iB=\sum_j invmat(i,j)*dM_B/dy_j etc etc
!
! it may not be very efficient but first get it right ....
! tpindep(1) is TRUE if T variable, tpindep(2) is TRUE if P are variable
!
! >>> THIS IS THE PRINCIPAL VERSION of calc_dgdyterms WITH SAVE
!
    implicit none
    integer ia,nrel,noofits
    logical tpindep(2)
    double precision, dimension(*) :: mamu
    double precision mag,mat,map
! no longer used ...
!    type(saveddgdy), pointer :: saved
! pmi is the phase data record for this phase
    type(meq_phase), pointer :: pmi
 ======================
```

```
  subroutine calc_dgdyterms2(iy,nrel,mamu,mag,mat,map,pmi)
! Called only by meq_calc_phase_derivative
! for the contribution to G for a single phase
! it should be similar to calc_dgdyterms1
    implicit none
    integer iy,nrel
    double precision mag,mat,map,mamu(*)
    type(meq_phase), pointer :: pmi
 ======================
  subroutine calc_dgdytermsh(nrel,ia,tpindep,hval,mamu,mag,mat,map,pmi,&
       curmux,noofits)
! This is a variant of dgdyterms1 including a term multiplied with each
! term (hval) in the summation over the comstituents as needed when calculating
! an equation for fix V or H.  If hval(i)=1.0 it should give the same
! results as dgdyterms1
!
! calculate the terms in the deltay expression for amounts of component ia
!
! DM_A = \sum_B mu_B*MAMU(B) - MAG - MAT*dt - MAP*dp
!
! where MAMU=\sum_i dM_A/dy_i*\sum_j invmat(i,j)*dM_B/dy_j
!       c_iB=\sum_j invmat(i,j)*dM_B/dy_j etc etc
!
! it may not be very efficient but first get it right ....
! tpindep(1) is TRUE if T variable, tpindep(2) is TRUE if P are variable
    implicit none
    integer ia,nrel,noofits
    logical tpindep(2)
    double precision, dimension(*) :: hval,mamu
    double precision mag,mat,map
    double precision curmux(*)
! pmi is the phase data record for this phase
    type(meq_phase), pointer :: pmi
 ======================
  subroutine calc_dgdytermshm(nrel,ia,tpindep,hval,mamu,mag,mat,map,&
       mamu1,mag1,mat1,map1,pmi,curmux,noofits)
! This is a variant of dgdyterms1 including a term multiplied with each
! term (hval) in the summation over the comstituents as needed when calculating
! an equation for fix V or H.  If hval(i)=1.0 it should give the same
! results as dgdyterms1
!
! Probably only one of calc_dgdytermshm or calc_dgdytermsh is needed
! calculate the terms in the deltay expression for amounts of component ia
!
! DM_A = \sum_B mu_B*MAMU(B) - MAG - MAT*dt - MAP*dp
!
```

```
! where MAMU=\sum_i dM_A/dy_i*\sum_j invmat(i,j)*dM_B/dy_j
!       c_iB=\sum_j invmat(i,j)*dM_B/dy_j etc etc
!
! it may not be very efficient but first get it right ....
! tpindep(1) is TRUE if T variable, tpindep(2) is TRUE if P are variable
    implicit none
    integer ia,nrel,noofits
    logical tpindep(2)
    double precision, dimension(*) :: hval,mamu,mamu1
    double precision mag,mat,map,mag1,mat1,map1
    double precision curmux(*)
! pmi is the phase data record for this phase
    type(meq_phase), pointer :: pmi
```

9.9 Some subroutines to deal with state variable function and in particular dot derivatives

The dot derivatives require second derivatives of the Gibbs energy from the most recent calculation. These are saved in the ceq record but it also requires the equilibrium matrix to know the current set of conditions. Thus some of the routines originally in the thermodynamic model package had to be moved here.

9.9.1 Evaluate all state variable functions

This is called from the user interface to calculate values of all entered state variable functions. It writes the functions and values on unit kou.

```
 subroutine meq_evaluate_all_svfun(kou,ceq)
! evaluate (and list if kou>0) the values of all state variable functions
   implicit none
   integer kou
   TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.9.2 Get the value of one or more state variable or function

This is called whenever the value of a state variable or a state variable function is needed. In particular the post processor will call this to obtain values to be plotted.

```
 subroutine meq_get_state_varorfun_value(statevar,value,dummy,ceq)
! used in OCPLOT to extact value of state variable of symbol
! NOTE if a specific function is given only this function evaluated
   implicit none
   character statevar*(*),dummy*(*)
   double precision value
   TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.9.3 Evaluate a state variable function

This function returns the value of a state variable function. The argument lrot is the index where the function is stored. The feature of having formal argument to state variable function has not been implemented.

```
 double precision function meq_evaluate_svfun(lrot,actual_arg,mode,ceq)
! evaluates all funtions as they may depend on each other
! actual_arg are names of phases, components or species as @Pi, @Ci and @Si
! needed in some deferred formal parameters  (NOT IMPLEMENTED YET)
! if mode=1 always evaluate, if mode=0 several options
   implicit none
   integer lrot,mode
   character actual_arg(*)*(*)
   TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.9.4 Initiate the equilibrium matrix for a derivative calculation

When calculating a dot derivative this subroutine must be called to create a temporary equilibrium data structure.

```
  subroutine initiate_meqrec(svr,svar,meqrec,ceq)
! this is to setup data for a state var derivative calculation
! taken from the normal initialization of an equilibrium calculation
! it also solves a modified equil matrix once to get delta-amounts and mu
    TYPE(meq_setup), pointer :: meqrec
    TYPE(gtp_state_variable), pointer :: svr
    double precision, allocatable :: svar(:)
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.9.5 Calculate the value of a state variable derivative

This calculates a derivative of state variable svr1 with respect to state variable svr2. At present svr2 must be T or P but the intention is to handle all kinds of derivatives. The values returned depend on the conditions set at the last calculated equilibrium.

```
  subroutine meq_state_var_dot_derivative(svr1,svr2,value,ceq)
! calculates a state variable value, dot derivative, (in some cases)
! svr1 and svr2 identifies the state variables in (dstv1/dstv2)
! check that svr2 2 is a condition
! value is calculated value
! ceq is current equilibrium
! NOTE that when plotting after a STEP/MAP the number of phases in the
! dynamic memory (ceq) may be different that the static memory
! this is indicated by setting mmdotder nonzero
!
    implicit none
```

```
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(gtp_state_variable), pointer :: svr1,svr2
    double precision value
```

9.9.6 Calculate the value of a state variable derivative for a single phase

This is called by meq_state_var_value_derivative for each stable phase in the system.

```
  subroutine meq_calc_phase_derivative(svr1,svr2,meqrec,iph,iel,&
       svar,jj,value,ceq)
! Calculate contribution for one phase, one or all elements
! svr1 and svr2 identifies the state variables in (dstv1/dstv2)
! value is calculated value returned
! iph and iel indicate possible phase or element
! svar is solution to equil matrix, potentials and phase amounts
! jj is an attempt to index phases in svar, starting with 1
! ceq is current equilibrium
!
! THIS IS UNFINISHED can only handle H.T
!
    implicit none
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(gtp_state_variable), target :: svr1,svr2
    TYPE(meq_setup), pointer :: meqrec
    integer iph,iel,jj
    double precision value,svar(*)
```

9.9.7 Calculate the slope of a liquidus surface

This is a tentative unfinished routine to calculate the slope of a liquidus surface in the direction of component X as X(liquid,X).T

```
 subroutine meq_slope(mph,svr,meqrec,value,ceq)
! Test subroutine for x(phase,A).T   UNFINISHED
    TYPE(meq_setup) :: meqrec
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(gtp_state_variable) :: svr
    double precision value
    integer mph
```

9.10 Subroutines used in assessments

These subroutines calculates the difference between the experimental data and the same property calculated from the model.

They make use of the data structure firstash declared in the GTP package and which must have values for all relevant coefficients and data. The values of X must be transferred to the relevant TPfun constants and F must be calculated for all experiments.

```
  subroutine calfun(m,n,x,f,info,niter)
! This is called by the LMDIF1 routines and calls an OC subroutine
! as I had problems using EXTERNAL
! M is number of errors
! N is number of variables
! NOTE order of X and F switched in CALFUN and ASSESSMENT_CALFUN !!!
    integer m,n,info,i,niter
    double precision f(m),x(n)
=========================
  subroutine assessment_calfun(nexp,nvcoeff,errs,xyz)
! nexp is number of experiments, nvcoeff number of coefficients
! errs is the differences between experiments and value calculated by model
! returned by this subroutine
! xyz are the scaled current model parameter values
    implicit none
    integer nexp,nvcoeff
    double precision errs(*),XYZ(*)
!    type(gtp_assessmenthead), pointer :: ash
```

9.11 Subroutines to extract extra data

This routine lists results from an assessment by should maybe be in the GTP package.

```
  subroutine listoptshort(lut,mexp,nvcoeff,errs)
! short listing of optimizing variables and result
    integer lut,mexp,nvcoeff
    double precision, allocatable, dimension(:) :: errs
!    type(gtp_equilibrium_data), pointer :: ceq
```

It is possible to add extra output to calculations in connection with the command "enter many_equil" used in a "calculate all" command. These are calculated by this routine and the values listed on the output unit lut or, if ceq%extra(3) has a 0 as first non-blank character, a GNUPLOT graphics file called oc_many0.plt will be opened and pun set to 30 and the calculated values will be written on this file.

```
 subroutine meq_list_experiments(lut,ceq)
! list all experiments into text, special to handle derivatives ...
   implicit none
   integer lut
   TYPE(gtp_equilibrium_data), pointer :: ceq
---------------
 subroutine meq_get_one_experiment(ip,text,seqz,ceq)
! list the experiment with the index seqz into text
! It lists also experiments that are not active ??
! UNFINISHED current value should be appended
   implicit none
```

```
   integer ip,seqz
   character text*(*)
   TYPE(gtp_equilibrium_data), pointer :: ceq
---------------
 subroutine list_equilibrium_extra(lut,ceq,pun)
! list the extra character variables for calculate symboles and
! list characters (if any),  It is used in pmon and is part of matsmin
! because it calls subroutines which need access to calculated results
! If the first non-blank character of ceq%eqextra(3) is 0 (zero) then pun
! will be used as a file number to generate a plotfile with calculated values
   implicit none
   integer lut,pun
   TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.12 Equilibrium calculations for a single phase

These routines are used to calculate the equilibrium for a single phase with internal degrees of freedoms at known T, P and overall composition.

```
  subroutine equilph1a(phtup,tpval,ceq)
! equilibrates the constituent fractions of a phase using its corrent comp.
! phtup is phase tuple
! tpval is T and P
! ceq is a datastructure with all relevant thermodynamic data
    implicit none
    double precision tpval(*)
    TYPE(gtp_phasetuple), pointer :: phtup
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(meq_setup) :: meqrec
---------------
  subroutine equilph1b(phtup,tpval,xknown,gval,cpot,tyst,ceq)
! equilibrates the constituent fractions of a phase for mole fractions xknown
! phtup is phase tuple
! tpval is T and P
! ceq is a datastructure with all relevant thermodynamic data
! gval is the Gibbs energy calculated as xknown(i)*cpot(i)
! cpot are the (calculated) chemical potentials
! tyst is TRUE means no outut
    implicit none
!    integer mode
    TYPE(meq_setup) :: meqrec
    double precision tpval(*),xknown(*),cpot(*),gval
    TYPE(gtp_equilibrium_data), pointer :: ceq
    logical tyst
---------------
  subroutine equilph1c(meqrec,phr,tpval,xknown,ovar,ceq)
```

```
! iterate constituent fractions of a phase for mole fractions xknown
! tpval is T and P
! xknown are mole fractions
! ceq is a datastructure with all relevant thermodynamic data
! ovar are the chemical potentials
    implicit none
!    integer phase
    double precision tpval(*),xknown(*),ovar(*)
    TYPE(meq_setup) :: meqrec
    TYPE(meq_phase), dimension(*), target :: phr
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.12.1 Calculation of the Darken stability matrix

These routines calculate the terms in the stability matrix, i.e. $\frac{\partial G_A}{\partial N_B}$ and some related properties. If mobility values are provided it also calculates the (unreduced) diffusion coefficients.

```
  subroutine equilph1d(phtup,tpval,xknown,cpot,tyst,nend,mugrad,mobval,ceq)
! equilibrates the constituent fractions of a phase for mole fractions xknown
! and calculates the Darken matrix and unreduced diffusivities
! phtup is phase tuple
! tpval is T and P
! ceq is a datastructure with all relevant thermodynamic data
! cpot are the (calculated) chemical potentials
! tyst is TRUE means no outut
! nend is the number of values returned in mugrad
! mugrad are the derivatives of the chemical potentials wrt mole fractions??
! mobval are the mobilities
    implicit none
    integer nend
    logical tyst
!CCI
    double precision, intent ( inout ) :: mugrad(*),mobval(*)
    double precision tpval(*),xknown(*),cpot(*)
!CCI
    TYPE(gtp_phasetuple), pointer :: phtup
    TYPE(gtp_equilibrium_data), pointer :: ceq
---------------
  subroutine equilph1e(meqrec,phr,tpval,xknown,ovar,tyst,&
       noofend,mugrad,mobval,ceq)
! iterate constituent fractions of a phase for mole fractions xknown
! and calculate derivatives of MU and diffusion coefficients
! tpval is T and P
! xknown are mole fractions
! nrel is the number of components (elements)
! ovar are the chemical potentials
```

```
! tyst is TRUE if no output
! mugrad is the derivatives of the chemical potentials wrt mole fractions??
! mobval are the mobilities
! ceq is a datastructure with all relevant thermodynamic data
    implicit none
    integer noofend
!CCI
    double precision,  intent ( inout ) :: mugrad(*),mobval(*)
    double precision tpval(*),xknown(*),ovar(*)
!CCI
    logical tyst
    TYPE(meq_setup) :: meqrec
    TYPE(meq_phase), dimension(*), target :: phr
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

9.12.2 Utility routine for single phase calculation

This routine sets up the equilibrium matrix for a single phase calculation.

```
  subroutine equilph1_meqrec(phtup,meqrec,tyst,ceq)
!  subroutine equilph1b(phtup,tpval,xknown,cpot,tyst,ceq)
! equilibrates the constituent fractions of a phase for mole fractions xknown
! phtup is phase tuple
! tpval is T and P
! ceq is a datastructure with all relevant thermodynamic data
! cpot are the (calculated) chemical potentials
! tyst is TRUE means keep quiet
    implicit none
!    integer mode
    TYPE(meq_setup) :: meqrec
!    double precision tpval(*),xknown(*),cpot(*)
    TYPE(gtp_equilibrium_data), pointer :: ceq
    logical tyst
```

9.12.3 Utility routines for Equi-entropy tests

The equi-entropy check for solid phases have used these routines. The check is that a solid with an entropy higher than the liquid is not allowed to be stable. It is not used as superseeded by other code.

```
  subroutine check_eec_old(pmisol,pmiliq,meqrec,ceq)
! This checks EEC after calculating all phases if the solid phase has S > S^liq
! it is called if T>globaldata%sysreal(1) (set in user i/f)
! pmisol is pointer to solid data
! pmiliq is pointer to liquid data
! ceq is a datastructure with all relevant thermodynamic data
    implicit none
```

```
    type(meq_phase), pointer :: pmiliq,pmisol
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(meq_setup) :: meqrec
```

9.13 Some special calculation routines

These routines are used to calculate some special, often non-equibrium situations.

9.13.1 Calculation of T-zero

These routines are used to find the T and composition when two phases have the same Gibbs energy. It is the limit of diffixionless transformation.

```
  subroutine tzero(iph1,iph2,icond,value,ceq)
! calculates the value of condition "icond" for two phases to have same G
    implicit none
    integer iph1,iph2,icond
    double precision value
    type(gtp_equilibrium_data), pointer :: ceq
 ======================
  subroutine tzcalc(nv,xv,fvec,iflag)
! calculates the value of a condition for two phases to have same G
    implicit none
    integer nv,iflag
    double precision xv(*),fvec(*)
 ======================
  subroutine tzcalc_stoich(nv,xv,fvec,iflag)
! calculates the value of a condition for two phases to have same G
! called from smp2A during mapping
! Both phases have the same composition (stoichiometric constraits)
    implicit none
    integer nv,iflag
    double precision xv(*),fvec(*)
```

9.13.2 Calculation of paraequilibrium

These routines are used to find the composition of two phases with one element (carbon) has the same chemical potential in both phases but all other element have the same mole fraction. It is common form of transformation in steels.

```
 subroutine calc_paraeq(tupix,icond,xcond,meqrec,meqrec1,ceq)
! calculates a paraequilibrium between two phases tupix(1&2)
! icond is the index of the fast diffusing element
! xcond are the fractions of the element in the two phases at paraequilibrium
   implicit none
   integer tupix(2),icond
   double precision xcond(2)
```

```
   TYPE(meq_setup), pointer :: meqrec
   TYPE(meq_setup), allocatable, target :: meqrec1
   TYPE(gtp_equilibrium_data), pointer :: ceq
 ======================
 subroutine paraeqfun(nv,fracs,fvec,iflag)
! called by hydrid1 to solve a nonlinear system of equations setup
! by calc_paraeq to calculate the difference in chemical potential
! for a two-phase paraequilibrium.  Arguments are:
! nv number of variables, fracs the variable values, fvec the functions
! calculated by this routine
   implicit none
   integer nv,iflag
   double precision fracs(*),fvec(*)
```

9.13.3 Calculate the equilibrium with special care

There is a command to use this subroutine when there are problems to converge.

```
  subroutine calculate_carefully(mode,ceq)
    implicit none
! calculate an equilirium carefully (bosses_method)
! step 1: Calculate with gridminimizer and merge (already done)
!         Alternatively enter with a set of stable phases
!         which has converged at another calculation.
!      2: suspend unstable phases
!      3: calculate with iterative method,
!      4: set all suspended as dormant,
!      5: calculate iterative again to see if any dormant has dgm>0
!      6: if so set it entered and goto 5
!      7: set all phases entered
! mode 0 means all step, nonzero may change some of these steps
! mode 1 means set phases entered one by one, largest driving force first
    integer mode
    type(gtp_equilibrium_data), pointer :: ceq
```

9.13.4 Calculate when a new phase becomes stable

This allowes the user to find the value of a condition to find when a specified phase becomes stable, for example the melting point.

```
  subroutine calctrans(cline,last,ceq)
! calculate a phase transition
    character cline*(*)
    integer last
    type(gtp_equilibrium_data), pointer :: ceq
```

9.13.5 Confidence interval of a calculation

This routine valies the current conditions within limits specified by the user and lists if there are any phase change and how much the amount of phase varies and the chemical potentials of the elements.

```
  recursive subroutine calc_conf_interval(lut,unc,ceq)
! Provide some confidence intervals of the results
! lut is output unit
! unc is condition uncertainty in %
! ceq is equilibrium record
    implicit none
    integer lut
    double precision unc
    type(gtp_equilibrium_data), pointer :: ceq
```

9.14 Miscelaneous

Used for debugging.

```
  subroutine list_stable_phases(text,its,iadd,irem,meqrec,ceq)
! debug listing of stable phases
! meqrec contains all necessary data ...
    character*(*) text
    integer its,iadd,irem
    type(meq_setup) :: meqrec
    type(gtp_equilibrium_data), pointer :: ceq
```

10 Table of all 44 functions and subroutines

Name	File
double precision function meq_evaluate_svfun	matsmin.F90
logical function same_composition	matsmin.F90
subroutine assessment_calfun	matsmin.F90
subroutine calc_conf_interval	matsmin.F90
subroutine calc_dgdyterms1	matsmin.F90
subroutine calc_dgdyterms1X	matsmin.F90
subroutine calc_dgdyterms2	matsmin.F90
subroutine calc_dgdytermsh	matsmin.F90
subroutine calc_dgdytermshm	matsmin.F90
subroutine calc_paraeq	matsmin.F90
subroutine calceq2(mode,ceq)	matsmin.F90
subroutine calceq3	matsmin.F90
subroutine calceq7	matsmin.F90
subroutine calctrans	matsmin.F90
subroutine calculate_carefully	matsmin.F90
subroutine calfun	matsmin.F90
subroutine check_eec	matsmin.F90
subroutine corriliq_d2gdyidyj	matsmin.F90
subroutine equilph1_meqrec	matsmin.F90
subroutine equilph1a	matsmin.F90
subroutine equilph1b	matsmin.F90
subroutine equilph1c	matsmin.F90
subroutine equilph1d	matsmin.F90
subroutine equilph1e	matsmin.F90
subroutine initiate_meqrec	matsmin.F90
subroutine list_equilibrium_extra	matsmin.F90
subroutine list_stable_phases	matsmin.F90
subroutine listoptshort	matsmin.F90
subroutine meq_calc_phase_derivative	matsmin.F90
subroutine meq_evaluate_all_svfun	matsmin.F90
subroutine meq_get_one_experiment	matsmin.F90
subroutine meq_get_state_varorfun_value	matsmin.F90
subroutine meq_list_experiments	matsmin.F90
subroutine meq_onephase	matsmin.F90
subroutine meq_phaseset	matsmin.F90
subroutine meq_sameset	matsmin.F90
subroutine meq_slope	matsmin.F90
subroutine meq_state_var_dot_derivative	matsmin.F90
subroutine paraeqfun	matsmin.F90
subroutine setup_comp2cons	matsmin.F90
subroutine setup_equilmatrix	matsmin.F90
subroutine two_stoich_same_comp	matsmin.F90
subroutine tzcalc	matsmin.F90
subroutine tzcalc_stoich	matsmin.F90
subroutine tzero	matsmin.F90

11 Summary

That's all.

References

[GTP] The General Thermodynamic Package, part of the OC software documentation, available at the opencalphad website.

[ochelp] Manual of the OpenCalphad software, part of the OC software documentation, available at the opencalphad website.

[opencalphad] http://www.opencalphad.org

[github] opencalphad repository at http://www.github.com/sundmanbo/opencalphad

[73Eri] G Eriksson and E Rosén, Chem Scripta **4** (1973) 193

[81Sun] B Sundman and J Ågren, J Chem Phys **42** (1981) 297

[81Hil] M Hillert, Physica B, **103B** (1981) 31

[82Luk] H L Lukas, J Weiss and E-Th Henig, Calphad **6** (1982) 229

[84Jan] B Jansson, Ph D thesis, KTH, Stockholm (1984)

[85Hil] M Hillert, B Jansson, B Sundman and J Ågren Metall Trans A **16A** (1985) 261

[01Hil] M Hillert, J Alloys Compd **320** (2001) 161

[07Luk] H L Lukas, S G Fries and B Sundman, *Computational Thermodynamics, the CALPHAD method*, Cambridge Univ Press (2007)

[09Flo] C A Floudas and C E Gounaris, J of Global Optimization **45** (2009) 3

[15Sun1] B Sundman, U R Kattner, M Palumbo, S G Fries, Integr Mat and Manu Innov., (2015) 4:1

[15Sun2] B Sundman, X-G Lu, H Ohtani, Comp Mat Sci **101**(2015) 127

[21Sun] Bo Sundman, Nathalie Dupin and Bengt Hallstedt, Calphad **75** (2021) 102330

[SMP] The STEP, MAP and PLOT package, part of the OC software documentation, available at the opencalphad website.

[ASSESS] The assessment module of OC, part of the OC software documentation, available at the opencalphad website.

[OCASI] The OCASI documentation, part of the OC software documentation, available at the opencalphad website.

[16Sun] B Sundman, U R Kattner, C Sigli, M Stratmann, R Le Tellier, M Palumbo and S G Fries, Comp Mat Sci, **125** (2016) 188