Step, Map and Plot (SMP) in OpenCalphad (OC)

Basic description and documentation

Bo Sundman, August 26, 2019

NOTE: This documentation unfinished and very rudimentary.

This is part of the Open Calphad (OC) documentation of the free software. The other parts are the General Thermodynamic package (GTP) the equilibrium calculation module (HMS), the Application Software Interface (OCASI) and the metlib utility package.

The software is available on http://www.opencalphad.org or on the opencalphad repository at http://www.gitbub.com/sundmanbo/opencalphad.

An introduction to the Open Calphad initiative can be found in [1]. To learn using the OC software there are the following information is provided:

- Installation guides for different OS
- Getting starting
- news-OC
- OC-macros
- User Guide (also available as on-line help)

This part of the documentation describes the data structures and software for the calculation of diagrams. For examples using OC and in particular the step, map and plot commands see the user guide and the macro examples.

Contents

1 Introduction **4**

1.1 Equilibrium calculations 5

1.2 Graphics driver 5

1.3 Examples of phase diagrams 5

1.4 Parallel calculations 6

2 Property diagrams **6**

2.1 Lines and nodes 6

2.2 Diagrams for phase fractions and thermodynamic properties 6

3 Phase diagrams **7**

3.1 The Zero Phase Fraction (ZPF) lines 7

3.2 Projecting and sectioning a multidimensional diagram to 2D 7

3.3 Phase diagrams in 3D 8

3.4 Potential and extensive axis variables 8

3.5 Solubility lines 8

3.6 Phase diagrams with tie-lines in the plane 9

3.6.1 An alternative method to calculate diagrams with tie-lines in the plane 9

3.7 Unary phase diagrams 11

3.8 Binary phase diagrams 12

3.9 Isothermal sections of ternary phase diagrams 12

3.10 Multicomponent phase diagrams 12

4 Basic data structures for STEP and MAP **13**

4.1 The line record 13

4.2 The node record 15

4.3 The axis record 16

4.4 The save record 16

4.5 Record to store texts plotted on a diagram 17

4.6 The graphics option record 17

5 Subroutines and functions **18**

5.1 The setup routine for step and mapping 18

5.2 Converts a start equilibrium to a start point along a line 19

5.3 Replace an axis variable by a fix phase 19

5.4 Finding a ZPF line to follow 20

5.5 Storing a calculated point along the line 20

5.6 Handling the last equilibrium along a line 21

5.7 Changing the active axis for next equilbrium calculation 21

5.8 Calculating the next equilibrium along a line 22

5.9 Calculating a node point 23

5.10 Creates a node with several exit lines 23

5.11 Reserves a record to save a calculated equilibrium 24

5.12 Finds a new line to be calculated 24

5.13 Creates records for saving equilibria 25

5.14 Delete results 25

5.15 Checks if the diagram has tie-lines in the plane 25

5.16 Checks if the node point is an invariant equilibrium 26

5.17 Tries to handle problems when mapping 26

5.18 Using smaller steps when there are convergence problems 26

5.19 List or amend stored equilibria 27

5.20 Plotting diagrams 27

5.21 Plotting ternary isothermal sections 29

5.22 Calculate a point in a diagram to list stable phases 31

5.23 Calculates equilibria for one phase at a time 31

5.24 Check an abbreviated phase name 32

5.25 Extract conditions for a diagram 32

6 Summary **32**

1 Introduction

The calculation of phase diagram is an important part of any thermodynamic software. A phase diagram gives a general "map" of a system for varying external conditions. Behind the phase diagram lies many individual equilibrium calculations which determine the solubility of the different elements in the phases. This is the basis of the so called "Calphad" technique. Fairly complete binary phase diagrams can be found in collections but very few ternary ones and it is unlikely one can find a aparticular ternary phase diagram in these collections. Phase diagrams with more than 3 components are quite complicated and not useful unless one can obtain one that is cenered at the composition one is interested. As modern alloys usually contain at least 4-7 components this limits the use of published collections of phase diagrams.

The big advantage with the Calphad method is that if one has an assessed database thermodynamic with the interesting elements it is possible to calculate several different phase diagrams in the composition and temperature ranges of interest starting from binary to diagrams with 8-10 components. This make sit possible to find interesting composition and heat treatment ranges to obtain an alloy with the desired propeties.

For systems with 3 or more components the phase diagrams cannot be plotted as easily as a binary system although the calculations is not much more complicated using modern software. In many cases so called "property diagrams" calculated varying a single variable condition, for example T, are equally useful as a phase diagram in order to understand a multicomponent system.

A multicomponent thermodynamic system consists of several phases which are stable at various temperature and composition ranges. The Gibbs energy of each phase is independently modeled in the thermodynamic software package as a function of T, P and the phase composition. The equilibrium of a system is found by minimizing the Gibbs energy for the given set of conditions using the thermodynamic models for the phases. At equilibrium the system may consist of one or more phases with different amounts and compositions. By varying one or more conditions along an axis it is possible to calculate how the amount and composition of the phase changes and also many other properties like chemical potentials, heat capacities etc.

To calculate a diagram the user must first provide the conditions needed to calculated a single equilibrium, a start point, and then select one, two or more of the conditions in his system as axis variables before trying to generate a diagram. The simplest set of conditions is on T, P and the overall composition. For such conditions there is always a well defined equilibrium. But the user may have conditions on the composition of a phase or the chemical potentials, MU of a component or the enthalpy of the system. In such cases the specified equilibrium may not exist.

1.1 Equilibrium calculations

In order to calculate a property or phase diagram the primary tool is a routine that can calculate the equilibrium for a flexible set of conditions. This is achieved by the HMS package of OC described in the paper by Sundman et al.[2] and in the HMS documentation which is part of the OC software. For a general introduction to equilibrium calculations see the book by Hillert[3] or Lukas et al.[4]. A general presentation of to OpenCalphad project is given in [1] and the minimization algorithm in [2].

A particular feature of the algorithm used for the equilibrium calculation is the possibility to prescribe that one or more phases are stable (with the status fixed). This feature is the key to follow the lines in the phase diagram where the set of stable phases changes.

It is also possible to prescribe that a phase should not be stable and this makes it possible to calculate metastable diagrams.

The HMS package has a well defined data structure where the result of an equilibrium is stored and HMS make use of the GTP package where the models of the Gibbs energy of each phase is described. In particular the equilibrium record, gtp_equilibrium_data is used to retrieve the composition and the amount of the phases. This is described in the GTP documentation, descriptions pf others record referenced here may be found in the HMS documentation.

1.2 Graphics driver

The OC software generates a data file for the GNUPLOT [5] software which if free and can be downloaded and installed for almost any computer. If GNUPLOT is installed correctly it will be started automatically by OC and the graphics displayed on the terminal by the plot command in OC. However, the graphics terminal drivers may differ on different systems and on Windows the "wxt" terminal is recommended for plotting on the screen. For other systems the user may have to select other terminal drivers.

The OC software has now been tested on Windows, serveral Linux dialects and Mac OS and athough it may need some adjustments for each OS it should work. You may have to change the definition of the terminals in the pmon6.F90 to obtain the best results.

1.3 Examples of phase diagrams

For examples of phase diagrams calculated by OC please look at the OC-macros.pdf manual.

1.4 Parallel calculations

Each line in a diagram can be calculated independently of all others (but at node points where several lines meet) and there are provisions, although not yet implemented, to calculate diagrams in parallel.

2 Property diagrams

For any diagram the user must first set conditions for a single equilibrium calculations. That means c+2 conditions where c is the number of components. The most common conditions are for T, P and the amount of the components. When the composition should vary in the diagram it is normally best to calculate for a given size of the system by giving a condition on the size of the system N and then specify the minor components as mole fractions, $x(i)$, or mass percent $w\%(i)$ where i is the component. Many other types of conditions can be used as explained in the documentation of the equilibrium module.

For a property diagram the user must select one of his conditions as axis variable and give a minimum and maximum value for the axis and a maximal increment. The STEP procedure will start at the start point and change the axis variable with the prescribed increment until it has calculated all equilibria between the minimum and maximum axis values. The calculated equilibria will be stored in a separate data array.

2.1 Lines and nodes

Whenever there is a change in the set of stable phases the STEP procedure in the OC software will calculate the exact value of the axis variable for the phase set change. Such a "node" point will be stored in the normal equilibrium list with a name like _STEP_ij where ij is a sequential index.

2.2 Diagrams for phase fractions and thermodynamic properties

There are several kinds of property diagrams for example showing the amount of phases as function of temperature or how the chemical potentials depend on composition in a binary system. In the macro_OC.pdf many examples are shown.

3 Phase diagrams

A phase diagram has at least two axis and the lines (or surfaces in 3D and hypersurfaces in higher dimensions) in a phase diagram separate regions with different sets of stable phase. Phase diagrams in 3D may be useful for teaching ternary phase diagrams but they are of limited interest in practice because most real alloys have more then 3 components and complete phase diagrams for such systems cannot be calculated or drawn.

3.1 The Zero Phase Fraction (ZPF) lines

The explanation of phase diagrams will be limited to phase diagrams calculated and plotted in 2D, the most common presentation. Along the lines in such a 2D phase diagram we have zero amount of one or more phases and for this reason the lines has been called "zero phase fraction" (ZPF) lines by Moral[6]. All lines in a phase diagram are ZPF but they may also have other properties.

3.2 Projecting and sectioning a multidimensional diagram to 2D

To obtain a 2D presentation of a multicomponent phase diagram we must make various kinds of sections or projections. Thus a 2D presentation of a projection may have some lines which seem to intersect but can be in different planes. Thus it can be complicated to draw and understand a phase diagram but with an equilibrium software like that used in OC[2] it is easy to calculate an equilibrium with the amount of a specific phase equal to zero and then follow the line with the amout zero along an axis specified for the phase diagram. In fact one can follow such a line using more than 2 axis as the method to replace all, except one, of axis the conditions with a fix phase, can be repeated.

While following a ZPF line a new phase may become stable, or a stable phase disappear, and the phase diagram software will then generate a "node" point with this phase fix also. Two or more new lines with a different phase fix with zero amount will exit from this node point. A line may also terminate at the lower or upper limit of an axis variable. Whenever a node point is found OC will by default call the grid minimizer to check that it is indeed a global minimum. If it is not the whole line ending at the node point will be set as excluded and will not appear in a plot. All calculated lines, also those excluded, can be listed with the command *list lines* and one may change a line from included/excluded by the command *amend lines*

When following a line the SMP package does not use the grid-minimizer because the results from the previous calculation provides good start values for the next calculation

because there is just a small change in one condition. But, as already mentioned, when calculating a node point the grid minimizer is used to test that one has not stepped into a miscibility gap or that another metastable phase should be stable.

In a future version it will be possible to test regularly at regular intervals that the set of stable phases represent the global equilibrium by calling the grid minimizer. But in the current version this has not yet been implemented.

As explained in any thermodynamic textbook the Gibbs phase rule determines the number of conditions needed to determine an equilibrium

$$f = n + 2 - p \tag{1}$$

where f is the degrees of freedoms, n the number of components and p the number of stable phases. To calculate an equilibrium $f = 0$ is necessary. For a unary (single component) system that means 3 conditions, for a binary 4 etc.

The type of phase diagram that is calculated depends on the number of potentials and (normalized) extensive variables used as conditions. One should also realize that the phase diagram is independent of the size of the system. With OC and some other software we may also draw the phase diagram using other axis variables properties than those used as conditions in order to calculate the diagram.

3.3 Phase diagrams in 3D

Phase diagrams in 3D have no scientific value but can be useful to understand the multidimensionallity of real phase diagrams. As a phase diagram for a system with n components have $n + 2$ dimensions only unary phase diagrams can be represented completely in 3D. For all other systems even a 3D diagram is a section or projection of the real diagram and becomes very difficult to interpret.

3.4 Potential and extensive axis variables

A potential like T, P or the chemical potential, μ for a component have the same value in all phases at equilibrium. Extensive variables, like the amount of a component, is normally different in the different stable phases. A phase diagram may thus change considerably depending on the choice of axis variables, see for example the Ag-Cu system in the macro-OC.pdf.

3.5 Solubility lines

For certain phase diagram the ZPF lines are also solubility lines, i.e. they represent the composition of a single phase in equilibrium with another phase. For binary T-x diagrams and ternary isothermal diagrams that is the case.

3.6 Phase diagrams with tie-lines in the plane

A special type of phase diagrams have "tie-lines in the plane" and in such cases a single equilibrium calculation provides points on both lines separating this phase region. These points can be connected by a tie-line. The tie-line is thus a line in a two-phase region connecting the compositions of the two phases in equilibrium. The most common type of phase diagram is a binary T-composition diagram where the regions with a single stable phase are separated by two-phase regions and ternary isothermal diagrams. The lines connecting the compositions of this two-phase equilibrium, called tie-lines, are in the same plane in the diagram.

3.6.1 An alternative method to calculate diagrams with tie-lines in the plane

The method of having a phase fix with zero amount to follow the line of stability of this phase in the phase diagram creates some problems as one may have to change the fix phase as well as acitive axis during the mapping, for example to pass a congrent melting point or the top of a miscibility gap. Thus another method has been tested.

Instead of replacing an axis condition with a fix phase we follow the movement of the center of the tie-line changing the values of the condition on both axis. As a tie-line can reach across the whole diagram if the solubilities are low one must control the length of each step to obtain smooth solubility curves. This technique was used by Lukas[7] in his software. Contrary to his algorithm the one implemented in OC also keep track of the metastable phases and can find when one reaches an invariant equilibrium where a third phase becomes stable and OC will calculate this invariant and generate new lines exiting from this, in the same way as when a phase is kept fixed. The hope is that this method will avoid some of the problems calculating these diagrams.

The first tieline is provided by the user as a start point. When this has been calculated the algorithm will be if both axis are compositions (an isothermal section):

1. Calculate the equilibrium for the start point. Save the axis coordinates of the two phases in $x_1^\alpha, x_2^\alpha, x_1^\beta, x_2^\beta$
2. Create two "start tie-line" records with this equilibrium with two opposite directions, d and add to the list of lines to map.
3. Select a start tie-lines record to map. If there are none go to point 15
4. The middle point of the tie-line is:

$$\begin{aligned} x_1 &= 0.5(x_1^\alpha + x_1^\beta) \\ x_2 &= 0.5(x_2^\alpha + x_2^\beta) \end{aligned} \tag{2}$$

Set the values of the axis increment factors f_j to a small value and the axis directions $d_j = d$ where d is the direction set in the line record, either +1 or -1.

5. Set as conditions

$$\begin{aligned} w_1 &= x_1 + d_1 \cdot f_1 \cdot \Delta x_1 \\ w_2 &= x_2 + d_2 \cdot f_2 \cdot \Delta x_2 \end{aligned} \tag{3}$$

where Δx_j is the maximum increment allowed for axis j. Save the phase compositions in $z_1^\alpha, z_2^\alpha, z_1^\beta, z_2^\beta$ and also the middle point in z_j.

6. This is the looping point of the algorithm. Calculate the equilibrium. Many things can happen.

   (a) If the calculation does not converge, do something.

   (b) If a new phase wants to be stable calculate the node point equilibrium with the new phase set fix and with zero amount and release one of the axis conditions.

   (c) If this fails do something elsee.

   (d) If a new node is found terminate the line with the node point as the last equilibrium.

   (e) If the new node is already known (many diagram have loops) remove the "start tie-line" corresponding to the calculated line and go to point 3.

   (f) If the new node is unknown create two "start tie-lines" with the new phase stable together with one of those stable along the current line and go to point 3.

7. If we are still here calculate the middle point of the new tie-line as:

$$\begin{aligned} x_1 &= 0.5(x_1^\alpha + x_1^\beta) \\ x_2 &= 0.5(x_2^\alpha + x_2^\beta) \end{aligned} \tag{4}$$

8. The intention is that the compositions set as conditions, w_j, for the calculation of the new tie-line should be as close as possible to the center of the next tie-line. Calculate the real movement of the center of the tie-line:

$$\begin{aligned} \Delta u_1 &= w_1 - z_1 \\ \Delta u_2 &= w_2 - z_2 \end{aligned} \tag{5}$$

   If Δu_j is negative the sign of the increment is wrong, set $d_j = -d_j$.

   Set the axis increment factor $f_j = \min(\frac{|\Delta u_j|}{f_j \Delta x_j}, 2f_j)$ not to be stuck with very small increments.

9. Calculate the four increments in the phase composition as:

$$\begin{aligned} \Delta x_1^\alpha &= x_1^\alpha - z_1^\alpha \\ \Delta x_2^\alpha &= x_2^\alpha - z_2^\alpha \\ \Delta x_1^\beta &= x_1^\beta - z_1^\beta \\ \Delta x_2^\beta &= x_2^\beta - z_2^\beta \end{aligned} \tag{6}$$

   If any $|\Delta x_j^\gamma| > \Delta x_j$ then decrease $f_j = 0.1 f_j$

10. Save the compositions in $z_1^\alpha, z_2^\alpha, z_1^\beta, z_2^\beta$.

11. Calculate the new composition for conditions:

$$\begin{aligned} w_1 &= x_1 + d_1 \cdot f_1 \cdot \Delta x_1 \\ w_2 &= x_2 + d_2 \cdot f_2 \cdot \Delta x_2 \end{aligned} \tag{7}$$

12. if any w_j is negative set its value to 1.0D-6 and mark that this is the last calculation.

13. if $w_1 + w_2 > 1$ then divide both values with the sum and set that this is the last calculation.

14. Go to point 6

15. The phase diagram is finished!

Special problems can occur for example if the tie-line is between two phases with fix composition, then the two phase region is just a line between two three phase regions. That has to be checked when the equilibrium calculation at the first step fails.

3.7 Unary phase diagrams

For unary diagrams the only variables are T and P, or the corresponding extensive variables S and V. When plotted with T and P as variables there are only single phase regions in the phase diagram. The lines represent the two-phase regions where one phase disappear and another become stable at the same value of T and P. However if the diagram is plotted with V as one axis variable there are two-phase regions where the amount of the two phases varies with the axis variables.

At an invariant equilibrium in a unary system there are three phases stable and there are three lines which meet at such a point. The "triple point" in the H_2O system with gas/liquid/ice is a fix point on the temperature scale. As there are several forms of ice at various pressure there are many triple points in the pure H_2O system but only one involving gas and liquid.

The phase diagram for the unary system H_2O is also well known for its "critical point" where gas and liquid become a single phase. That means the line separating gas and liquid ends at a specific value of T and P. Below this point gas and liquid have different molar volumes but at the critical point they have the same. The phase above the critical point is sometimes known as fluid.

3.8 Binary phase diagrams

In binary phase diagrams the lines are "zero phase fraction" lines but as a binary system has tie-lines in the plane most lines also represent "solubility lines" of a single phase. Thus one can apply many many relations for the lines in a binary system, like the lever rule, which are not generally applicable in multicomponent phase diagrams.

The lines in a binary phase diagram that are not solubility lines are associated with the invariant equilibria.

3.9 Isothermal sections of ternary phase diagrams

This type of diagram represented some special difficulties to plot as the axis variables are two normalized state variables: the mole fractions of two of the components. Almost all other kinds of phase diagrams have one potential axis and one extensive (composition) axis.

It is now possible to calculate such diagrams with OC but usually they require more than one start point selected manually by the user.

It is possible, but rather awkward, to calculate for example a binary $T - x$ diagram using the entropy or enthalpy as one axis instead of T. If one is interested in this kind of diagram it is easier to calculate it with T as axis and then plot the diagram with enthalpy of the different phases, "HM(*)" as one axis.

3.10 Multicomponent phase diagrams

For multicomponent systems we normally do not have tie-lines in the plane, unless all conditions except one are potentials. But all lines in a multicomponent system are ZPF lines separating areas with different sets of stable phases. In multicomponent systems with two axis there is a simple rule that the number of stable phases always will change by +1 or - 1 when one cross a line from one region to another. If one of the axis is a potential, like T, and if there are only a few elements, there is a small chance that one has an invariant equilibrium in the plane of the diagram and this rule does not apply when crossing a line representing an invariant equilibrium. Such an invariant must be at a constant value of the potential, and there is no change in the number of stable phase crossing it (except at points where there are other lines ending at the invariant

line). But even it the number of phases are the same there will be at least two different phases stable on either side of the invariant line. When crossing an invariant line in a binary T-x diagram we have also the same number of phases on the two sides but only one is different.

The use of an algorithm that can follow ZPF lines means that it is quite easy to calculate a multicomponent phase diagram, we start from one ZPF line and generate new lines whenever the set of phases changes. There can be separate sets of ZPF lines that does not cross in the calculated region so several start points may be needed. But it can be quite difficult to plot a correct phase diagram as will be discussed later.

To follow a ZPF line we replace all but one of the axis conditions one by one with a fix phase and the mapping of a phase diagram is thus very similar to the step routine used for a property diagram. During the mapping we may have to change which axis variable that is used for stepping along the line. And of course take care to generate node points when a new phase wants to be stable or a stable one wants to disappear because there are several ZPF lines meeting at such a point.

4 Basic data structures for STEP and MAP

The following sections are not updated with the current software

During the STEP or MAP procedures all calculated equilibria along the lines and at the node points are saved in memory. It is thus possible to plot diagrams with other properties than those used for the calculation.

At present there is no way to save the calculated results on files for later use.

The records representing a calculated diagram are connected by pointers. There is a main node record called "maptop" from which all calculated data can be found. The maptop record can have pointers to other node records which are created when the set of stable phases changes. Each node record has pointers to two or more line records where the equilibria calculated along the line are stored.

It is possible to list this structure with the *list lines* command and one can remove individual lines from a plot using the *amend line* command.

4.1 The line record

This record keeps a linked list of equilibrium records calculated along a line.

```
  TYPE map_line
! This is record contains a list of calculated equilibria along a line
! These are pointers to map_node records at start and end of line.
     type(map_node), pointer :: start,end
```

```
! For threading this record must be separate from other threads
! This record is created by calling calceq7 the first equilibrium of line
      type(meq_setup) :: meqrec
! the active ceq record must be stored together with all others in order to be
! copied from the ceq record saved with the node record when line starts
      type(gtp_equilibrium_data), pointer :: lineceq
! this is the number of calculated equilibria for this line and the index
! of the first and last one stored.
! The stored equilibria has an internal next link.
! lineid is a sequential index of the lines. done is negative if done
! nfixphases are the number of fixed phases replacing axis conditions
! status kan be used to delete a line
      integer number_of_equilibria,first,last,lineid,done,nfixphases,status
! This is used during mapping to identify lines that have the same fixed phases
! if we have 3 or more axis there can be 2 or more fix phases along the line??
      type(gtp_phasetuple), dimension(:), allocatable :: linefixph
! Save the phase tuplet representing the phase fix at start node here
! If it wants to be stable at first step along a line change axis direction
!     type(gtp_phasetuple) :: nodfixph
! This is the phase index in the phr array
      integer nodfixph
! We must also save the number and set of stable phases and theit amounts
! as we will have different stable phases for different lines
      integer nstabph
      type(gtp_phasetuple), dimension(:), allocatable :: stableph
      double precision,     dimension(:), allocatable :: stablepham
! axandir is set when linenode is created to the axis and direction for first
! step from the node.  It can be changed to another axis and direction
! during map and indicate the current axis with active condition
! axchange remember the equilibrium number for an axis change
      integer axandir,axchange
! more is 1 while following the line, 0 for last equilibrium, -1 when finished
! termerr is zero unless line terminated with error, -1 means exit not used
! problem is nonzero if map_problems has been called
! lasterr is the last error occured calculating this line
      integer more,termerr,problems,lasterr
! firstinc is a value to add the the axis variable for the first equilibrium
! to avoid finding the node point again.  Evenvalue is the next value
! to calculate during a step calculation.  Both set when creating the node.
! At start nodes they are zero
      double precision firstinc,evenvalue
! During map the last axis values for ALL axis are stored here
      double precision, dimension(:), allocatable :: axvals
! If tie-lines in the plane we must also check the axis values for
! the other line as we may have to change the fix phase
      double precision, dimension(:), allocatable :: axvals2
```

```
! save previous values of axvals to handle axis changes ...
     double precision, dimension(:), allocatable :: axvalx
! factor to control length of step in axis with axtive condition
     double precision :: axfact
  end TYPE map_line
```

4.2 The node record

This record contain the equilibrium at a node points where several lines meet. When the calculation of a phase diagram starts the "start equilibrium" is used to find a line in the diagram, i.e. where a phase is stable with zero amount. This is stored in an initial node record. All other node records are linked from this record.

When the diagram is very large and one runs out of memory and must save intermediate results on a file, the equilibrium record at all currently unfinished lines are stored in a node record, with one exit line then the memory is cleared and mapping continues by search all node points for lines to calculate (not yet implemented though).

```
  TYPE map_node
! this record organizes the step/map results.  Whenever there is a
! change of the set of stable phases a node record is created and it
! can have links to several map_line records.  The map node record has a
! link to a gtp_equilibrium_data record (ceq) for the equilibrium at the node.
! This is copied to the map_line record when this is activated.
! In the map_line record an axis and direction to start is stored.
! NOTE all gtp_equilibrium_data (ceq) records are pointers to the global
! array as new composition sets may be created along any line.
! The node record is identified by the set of stable phases and the
! chemical potentials of the components.  One must be able to identify the
! node as one may fins the same node following different lines.
! locally stored linerecords for lines exiting the node
     type(map_line), dimension(:), allocatable :: linehead
! links to other nodes
! plotlink is used to overlay two or more map or step commands
     type(map_node), pointer :: first,next,previous,plotlink
! saved copy of the meqrec record used to calculate the node
     type(meq_setup) :: meqrec
! link to saved copy of the equilibrium record
     type(gtp_equilibrium_data), pointer :: nodeceq
! link to array of saved equilibrium record.   (only maptop?)
     type(map_ceqresults), pointer :: saveceq
! copy of nodeceq in saveceq (composition sets not updated but needed for plot)
     integer savednodeceq
! type_of_node not used?? should identify invariants when tie-line not in plane
! lines are number of line records
```

```
! noofstph is number of stable phases (copied from meqrec)
! tieline_inplane is 1 if so, 0 if step, -1 if no tie-lines (only maptop)
! number_ofaxis is the number of axis, 1=step;  (only maptop)
     integer type_of_node,lines,noofstph,tieline_inplane,number_ofaxis
! seqx is unique identifier for a map node
! seqy unique identifier for maplines, incremented for each line (only maptop)
     integer seqx,seqy
! nodefix is the phase held fix when calculating node (negative if removed)
     type(gtp_phasetuple) :: nodefix
! Value of T and P, copied from meqrec
     double precision, dimension(2) :: tpval
! chemical potentials, copied from meqrec
     double precision, dimension(:), allocatable :: chempots
! stable phase+compset, copied from meqrec (not used?)
     type(gtp_phasetuple), dimension(:), allocatable :: stable_phases
  end TYPE map_node
```

4.3 The axis record

This record contain a description of the axis set for the mapping of the phase diagram. It is usually created by user input in the user interface.

```
  TYPE map_axis
! description of the axis variables used for step/map
! The axis condition in bits and pieces
     integer nterm,istv,iref,iunit
     integer, dimension(:,:), allocatable :: indices
     type(gtp_state_variable), dimension(:), allocatable :: axcond
     double precision, dimension(:), allocatable :: coeffs
! the min, max and increment along the axis
     double precision axmin,axmax,axinc
! more must be initiated to 0, if nonzero replaced by a fixed phase
! seqz is the sequential index of the condition in the list (this is not
! changed if conditions are added (at the end) or deleted (active=1)
! we cannot use a pointer as that depend on the current equilibrium.
     integer more,seqz
! This is the last succesfully calculated axis value
     double precision lastaxval
  end TYPE map_axis
```

4.4 The save record

This record contains equilibria calculated along the lines. The indices to the saved equilibria are stored in the line record.

```
  TYPE map_ceqresults
! stores calculated equilibrium records
     integer size,free
     TYPE(gtp_equilibrium_data), dimension(:), allocatable :: savedceq
  end TYPE map_ceqresults
```

4.5 Record to store texts plotted on a diagram

The user can add texts that is plotted on the diagram. Each text to be shown on a diagram has a position and a text line and are stored in this record.

```
  TYPE graphics_textlabel
! To put labels on a graph we must store these in a list
     TYPE(graphics_textlabel), pointer :: nexttextlabel
     double precision xpos,ypos
     character*40 textline
  end type graphics_textlabel
!
```

4.6 The graphics option record

The axis and other options for plotting the diagram are stored here. It needs to be reorganized as some data is stored elsewhere.

```
  TYPE graphics_options
! setting options for the plotting, this replaces most arguments in the call
! to ocplot2(ndx,pltax,filename,maptop,axarr,form)
! ndx is mumber of plot axis, pltax is text with plotaxis variables
! filename is intermediary file (maybe not needed)
! maptop is map_node record with all results
! form is type of output (screen or postscript or gif)
     integer :: status=0,rangedefaults(3)=0,axistype(2)=0
     double precision, dimension(3) :: plotmin,plotmax
     double precision, dimension(3) :: dfltmin,dfltmax
! labeldefaults 0 if default, 1 if text provided in plotlabels
! linetype 0 is black full line, >100 is symbols
     integer :: labeldefaults(3),linetype,tielines=0
! label 1 is heading, 2 is x-axis text, 3 is y-axis text
     character*64, dimension(3) :: plotlabels
     logical gibbstriangle
! the set key command in GNUPLOT specifies where the line id is written
! it can be on/off, placed inside/outside, left/right/center, top/bottom/center,
! and some more options that may be implemented later ...
```

```
     character labelkey*24,appendfile*72
! text label to be written at a given position
     TYPE(graphics_textlabel), pointer :: firsttextlabel
! many more options can easily be added when desired, linetypes etc
  end TYPE graphics_options
```

5 Subroutines and functions

Many calls are made to routines in the HMS and GTP package to obtain results and handle phase information. Those described here are used only for the step, map and plotting of diagrams.

5.1 The setup routine for step and mapping

This routine organizes the STEP or MAP procedure.

This is the main routine to organize the STEP or MAP of a diagram. There is a separate routine, step_separate, described in section 5.23 for the command *step separate* because that calculates properties for each phase separately along a single axis.

This routine first calls map_startpoint in section 5.2 which converts the start equilibrium to a point where a phase if fixed for phase diagram calculations, and generates a map node for that point with two exit lines. For a step calculation it generates a map note with two exit lines at the start equilibrium. Then it follows all lines generated at node points until there are no left.

```
  subroutine map_setup(maptop,nax,axarr,seqxyz,starteq)
! main map/step routine
! maptop is the main map_node record which will return all calculated lines.
! nax is the number of axis (can be just one for STEP)
! axarr is an array of records specifying the axis for the step/map
! seqxyz are intial values for number of nodes and lines
! starteq is an equilibrium data record, if there are more start equilibria
! they are linked using the ceq%next index
    implicit none
    integer nax,seqxyz(*)
    type(map_axis), dimension(nax) :: axarr
    TYPE(gtp_equilibrium_data), pointer :: starteq
    TYPE(map_node), pointer :: maptop
```

5.2 Converts a start equilibrium to a start point along a line

For a step calculation this routine generates a node point with the start equilibrium with two exit lines.

For a map command to calculate a phase diagram it searches from the start equilibrium a point where the set of phases changes. It then generates a node point with two exit lines. It also selects an initial "active axis", i.e. the axis for which the condition are changed to follow the line.

```
  subroutine map_startpoint(maptop,nax,axarr,seqxyz,inactive,ceq)
! convert a start equilibrium to a start point replacing all but one axis
! conditions with fix phases.  The start equilibrium must be already
! calculated. ceq is a datastructure with all relevant data for the equilibrium
! A copy of ceq and the corresponing meqrec must be made and linked from maprec
! the axis conditions replaced by fix phases are inactive
! maptop is returned as a first nodepoint(although it is not a node)
! nax is number of axis, axarr records with axis information
! seqxyz is array with indices for numbering nodepoints and lines
! inactive is not really used (conditions replaced by fix phase)
! ceq is equilibrium record
 implicit none
 TYPE(gtp_equilibrium_data), pointer :: ceq
 TYPE(map_node), pointer :: maptop
 integer nax,seqxyz(*)
 integer inactive(*)
 type(map_axis), dimension(nax) :: axarr
```

5.3 Replace an axis variable by a fix phase

This routine replaces one axis variable by a fix phase for a phase diagram calculations. At present only two axis can be used but the algorithm can handle any number of axis and at least 3 axis will be implemented eventually.

```
  subroutine map_replaceaxis(meqrec,axactive,ieq,nax,axarr,tmpline,&
       inactive,ceq)
! replace an axis condition with a fix phase
! meqrec is equilibrium calculation record
! axactive is the axis with active condition, ieq is number of exiting lines
! ieq is the number of lines exiting from the startpoint
! nax is number of axis, axarr are description of the axis
! axarr is array with axis records
! tmpline is to transfer some line data to calling routine
! inactive is not really used.
```

```
! ceq is equilibrium record
    implicit none
    type(meq_setup), pointer :: meqrec
    integer nax,axactive,ieq
    type(map_axis), dimension(nax) :: axarr
    type(gtp_equilibrium_data), pointer :: ceq
    type(map_line), dimension(2) :: tmpline
    integer inactive(*)
```

5.4 Finding a ZPF line to follow

This subroutine is called by map_startpoint to vary the axis variables in order to find an equilibrium where the set of stable phases changes so it can replace one axis variable with a fix phase condition.

```
  subroutine map_startline(meqrec,axactive,ieq,nax,axarr,tmpline,ceq)
! find a phase to fix to replace an axis condition when we
! do not have tie-lines in the plane or when we
! have tie-lines in the plane but start in a single phase region
! meqrec is equilibrium record already initiated
! axactive is set to the axis with active condition
! ieq is the number of lines exiting from the startpoint
! nax is number of axis, axarr are description of the axis
! axarr are axis records
! tmpline is a line record ... not needed ... ??
    implicit none
    integer nax,axactive,ieq
    type(meq_setup), pointer :: meqrec
    type(map_line), dimension(2) :: tmpline
    type(map_axis), dimension(nax) :: axarr
    type(gtp_equilibrium_data), pointer :: ceq
```

5.5 Storing a calculated point along the line

After a successful equilibrium calculation along a ZPF line the calculate equilibrium is stored an the saveceq record by this routine. It calls reserve_saveceq in section 5.11 to find a location.

```
  subroutine map_store(mapline,axarr,nax,saveceq)
! store a calculated equilibrium
! mapline is line record
! axarr is array with axis records
! nax is number of axis
```

```
! saveceq is record for saved equilibria
    implicit none
    integer nax
    type(map_line), pointer :: mapline
    type(map_axis), dimension(nax) :: axarr
    type(map_ceqresults), pointer :: saveceq
```

5.6 Handling the last equilibrium along a line

When a line terminates, either at an axis end or when the set of stable phases changes, this routine handles everything. For phase diagrams it is possible that the node point where the line ends has already been created coming from another line. In such a case no new node is created but instead one of the already existing exit lines from this node is cancelled. To check that the node points are the same all chemical potentials are compared and must be identical and the set of stable phases must be the same. This also avoids eternal loops when lines are connected in a phase diagram.

```
  subroutine map_lineend(mapline,value,ceq)
! terminates gracefully a line at an axis limit or an error.
! mapline probably not needed except for testing
! value is last calculated axis value
! ceq is equilibrium record
    implicit none
    integer mode
    type(map_line), pointer :: mapline
    type(gtp_equilibrium_data), pointer :: ceq
    double precision value
```

5.7 Changing the active axis for next equilbrium calculation

During mapping it may sometimes be necessary to change the active axis variable used for moving along the ZPF line when the line changes direction. This can happen close to the top of a miscibility gap or at a congruent transformation. This routine tries to handle that.

The map_force_changeaxis is called when there are problems to calculate an equilibrium and tries to solve that by changing the active axis.

When the active axis changes the next few steps will be very short.

```
  subroutine map_changeaxis(mapline,nyax,oldax,nax,axarr,axval,bytax,ceq)
! changes the axis with active condition to nyax
! mapline is line record
! nyax is index of new active axis
```

```
! oldax is index of old active axis
! nax is number of axis (always 2?)
! axarr is array with axis records
! axval the value to set as condition on new axis
! bytax logical, if true ignore axval
! ceq is equilibrium record
    type(map_line), pointer :: mapline
    type(gtp_equilibrium_data), pointer :: ceq
    type(map_axis), dimension(nax) :: axarr
    logical bytax
    integer nyax,nax,oldax
    double precision axval
  subroutine map_force_changeaxis(maptop,mapline,meqrec,nax,axarr,axvalok,ceq)
! force change of axis with active condition.  Works only with 2 axis.
! (and for tie-line not in plane ??).  Calls map_changeaxis ...
! maptop is node record
! mapline is line record
! meqrec is equilibrium calculation record
! nax is number of axis, also in maptop record
! axarr is array with axis records
! axvalok is last successfully calculated axis value
! ceq is equilibrium record
    implicit none
    integer nax
    type(map_node), pointer :: maptop
    type(map_line), pointer :: mapline
    type(meq_setup) :: meqrec
    type(gtp_equilibrium_data), pointer :: ceq
    type(map_axis), dimension(*) :: axarr
    double precision axvalok
```

5.8 Calculating the next equilibrium along a line

This is the normal routine to take a step along the active axis and calculate a new equilibrium along a line. Several factors are used to control the step length.

```
  subroutine map_step(maptop,mapline,meqrec,phr,axvalok,nax,axarr,ceq)
! used also for map as mapping is stepping in one axis with fix phase condition
! calculate the next equilibrium along a line.  New phases can appear.
! axis with active condition can change and the direction.
! maptop is map node record
! mapline is line record
! phr is new array phase status (just for debugging)
! axvalok is last successfully calculated axis value
! nax number of axis, redundant as also in maptop record
```

```
! axarr is array with axis records
! ceq is equilibrium record
    implicit none
    integer nax
    type(map_node), pointer :: maptop
    type(map_line), pointer :: mapline
    type(meq_setup) :: meqrec
    type(meq_phase), dimension(*), target :: phr
    type(gtp_equilibrium_data), pointer :: ceq
    type(map_axis), dimension(*) :: axarr
    double precision axvalok
```

5.9 Calculating a node point

This is used to calculate a node point where the set of stable phases changes.

```
  subroutine map_calcnode(irem,iadd,maptop,mapline,meqrec,axarr,ceq)
! we have found a change in the set of stable phases.  check if this node
! already been found and if so eliminate a line record.  Otherwise
! create a new node record with line records and continue mapping one
! of these.
! irem and iadd are indices (in phr?) of phase that will disappear/appear
! maptop is map node record
! mapline is map line record
! meqrec is equilibrium calculation record, ! Note changes in meqrec is local,
!      not copied to mapline%meqrec!!!
! axarr is array with axis records
! ceq is equilibrium record
    implicit none
    integer irem,iadd
    type(map_node), pointer :: maptop
    type(map_line), pointer :: mapline
    type(meq_setup) :: meqrec
    type(map_axis), dimension(*) :: axarr
    type(gtp_equilibrium_data), pointer :: ceq
```

5.10 Creates a node with several exit lines

After calculating the node point this routine generates exit lines from the node point depending on the type of diagram.

For a diagram with tie-lines in the plane there are always three lines meeting at a node point. For an multicomponent isopleth there are normally two lines crossing at a node point, i.e. three new exit lines will be generated. But invariant equilibria in

isopleths will have twice as many exit lines as there are stable phases at the invariant. This has not yet been implemented.

```
  subroutine map_newnode(mapline,meqrec,maptop,axval,lastax,axarr,&
       phfix,haha,ceq)
! must be partially THREADPROTECTED
! first check if a node with this equilibrium already exists
! if not add a new node with appropriate lineheads and arrange all links
! Take care if tie-lines in the plane all lines do not have to be calculated
! NOTE: meqrec not the same as mapline%meqrec !! ??
! mapline is line record for current line
! meqrec has information about last calculated equilibrium
! maptop is node record
! axval is the axis value attemped to calculate when phase set wanted to change
! lastax is index of last active axis
! axarr are axis records
! phfix is phase which is set fix at node point
! haha is nonzero if the calculated equilibrium is invariant
! ceq is equilibrium record
    implicit none
    type(map_node), pointer :: maptop
    type(meq_setup) :: meqrec
    type(map_line), pointer :: mapline,nodexit
    type(map_axis), dimension(*) :: axarr
    type(gtp_equilibrium_data), pointer :: ceq
    integer phfix,lastax,haha
    double precision axval
```

5.11 Reserves a record to save a calculated equilibrium

This routine is called by the map_store routine in section 5.5

```
  subroutine reserve_saveceq(location,saveceq)
! must be THREADPROTECTED
! location index of reserved ceq record in saveceq
    implicit none
    integer location
    type(map_ceqresults), pointer :: saveceq
```

5.12 Finds a new line to be calculated

Searches all node points for the next line to be calculated. If there are none the step or map is finished.

```
  subroutine map_findline(maptop,axarr,mapfix,mapline)
! must be THREADPROTECTED
! Searches all node records from maptop for a map_line record to be calculated
! ?? already been found and if so eliminate a line record ??
! maptop map node record
! axarr array with axis records
! mapfix returned fixph record with phases to be ste as fixed for this line
! mapline returned mapline record for line to be calculated
    type(map_node), pointer :: maptop
    type(map_line), pointer :: mapline
    type(map_axis), dimension(*) :: axarr
    type(map_fixph), pointer :: mapfix
```

5.13 Creates records for saving equilibria

Called by map_setup to create a record where many calculated equilibria can be saved.

```
  subroutine create_saveceq(ceqres,size)
! creates an array of equilibrium records to save calculated lines for step
! and map
    type(map_ceqresults), pointer :: ceqres
    integer size
```

5.14 Delete results

Used when the system is re-initiated by a NEW command or a new map/step command without saving the previous.

```
  subroutine delete_mapresults(maptop)
! delete all saved results created by step or map
    TYPE(map_node), pointer :: maptop
```

5.15 Checks if the diagram has tie-lines in the plane

For a phase diagram with tie-lines in the plane the number of exit lines from a node point is 2 whereas for isopleths it is normally 3. This function returns different values depending on the type of diagram that is calculated.

```
  integer function tieline_inplane(nax,axarr,ceq)
! returns -1 if tielines are not in the plane (isopleth)
!          0 for step calculations (nax=1)
!          1 if tielines in the plane (binary T-X, ternary isopleths
```

```
!          set if more than one extensive variable is not axis variables
! nax number of axis
! axarr array with axis records
    integer nax
    type(map_axis), dimension(nax) :: axarr
    type(gtp_equilibrium_data), pointer :: ceq
```

5.16 Checks if the node point is an invariant equilibrium

A node point from an invariant equilibria have more lines that exit than normal nodes. For diagrams with tie-lines in the plane all node points are invariants.

```
  logical function inveq(phases,ceq)
! Only called for tie-lines not in plane.  If tie-lines in plane then all
! nodes are invariants.
! UNFINISHED
    integer phases
    type(gtp_equilibrium_data), pointer :: ceq
```

5.17 Tries to handle problems when mapping

This routine tries various way to handle problems calculating an equilibrium during STEP and MAP.

```
  subroutine map_problems(maptop,mapline,axarr,xxx,typ)
! jump here for different problems
! maptop map node record
! mapline current line record
! axarr array with axis records
! xxx current active axis value that caused problems to calculate
! typ indicates the type of problem
    integer typ
    type(map_node), pointer :: maptop
    type(map_line), pointer :: mapline
    type(map_axis), dimension(*) :: axarr
    double precision xxx
```

5.18 Using smaller steps when there are convergence problems

This subroutine is used when there is a convergence problem by decreasing the step from the last calculated equilibrium.

```
  subroutine map_halfstep(halfstep,axvalok,mapline,axarr,ceq)
! Used when an error calculating a normal step or a node point
! take back the last successfully calculated axis value and take smaller step
! possibly one should also restore the ceq record.
! halfstep number of times halfstep has been called for this problem
! axvalok last successfully calculated value of active axis
! mapline line record
! axarr array with axis records
! ceq equilibrium record
    implicit none
    integer halfstep
    double precision axvalok
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(map_line), pointer :: mapline
    type(map_axis), dimension(*) :: axarr
```

5.19 List or amend stored equilibria

For the user interface there is one subroutine to list all node points and lines with calculated equilibra and another to amend these. For the amendment the user can select to inactivate (suppress) and line or activate an inactivated line. During the mapping procedure a line may be automatically inactivated if the end point is found not to be a global equilibrium.

```
  subroutine list_stored_equilibria(kou,axarr,maptop)
! list all nodes and lines from step/map
! use amed to exclude/include lines
! kou output unit
! axarr array with axis records
! maptop map node record
    integer kou
    type(map_node), pointer :: maptop
    type(map_axis), dimension(*) :: axarr
  subroutine amend_stored_equilibria(axarr,maptop)
! allows amending inactive/acive status of all lines from step/map
    type(map_node), pointer :: maptop
    type(map_axis), dimension(*) :: axarr
```

5.20 Plotting diagrams

There are two subroutine to generate a graphical output from the step and map calculations. For a phase diagram the plot axis are by default the same as those used to generate the diagram, however, for an extensive variable when the diagram has tie-lines

in the plane this is modified by including a wildcard for the phase, thus for the binary Ag-Cu diagram (in macro map1.OCM) the calculated axis variable is x(cu) but the plotted axis is x(*,cu) meaning that lines with the Cu content in all stable phases are plotted.

For phase diagrams without tie-lines in the plane, the plotted extensive variable will be the same as used for the calculation because the other stable phases will have compositions outside the plane of calculation.

For step diagrams the default is the specified axis variable as horizontal axis and as vertical axis NP(*), meaning the amount in moles of the stable phases.

However, the user may select (almost) any state variable or symbol he likes as an axis variable. Some are less interesting but it is up to the user to decide and realize that.

The ocplot2 routine can only plot diagrams when one of the axis is a potential, like T. If both axis variables are extensive variables and the diagram has tie-lines in the plane the subroutine ocpolt3 is called, see section 5.21.

Inside ocplot2 the axis variable values selected are extracted from the calculated equilibria and stored in two variables. Some information about invariant lines are also added. Then the ocplot2B subroutine is called to generate a GNUPLOT file with the data to be plotted. The default plot unit is the screen but if the user selects a special graphical format like ACROBAT (pdf file) or POSTSCRIPT or GIF, an additional file using that format will be generated by GNUPLOT.

The user may also interactivly add labels to the diagram and it is possible to calculate the equilibrium set of phases for a point inside the diagram using the same axis as when calculating the diagram and use the phase names as label. A label may be interactivly moved and modified.

All lines in the phase diagram will be default be identified with the phase that is stable with zero amount along the line.

```
  subroutine ocplot2(ndx,pltax,filename,maptop,axarr,graphopt,pform,ceq)
! Main plotting routine, generates a GNUPLOT data file for a step/map calc
! NOTE for isothermal section ocplot3 is used (when 2 axis with wildcards)
! ndx is mumber of plot axis,
! pltax is text with plotaxis variables
! filename is the name of the GNUPLOT file
! maptop is map_node record with all results
! axarr is array of axis records
! graphopt is graphical option record
! pform is type of output (screen/acrobat/postscript/gif)
! ceq is equilibrium record
    implicit none
    integer ndx
    character pltax(*)*(*),filename*(*),pform*(*)
```

```
    type(map_axis), dimension(*) :: axarr
    type(map_node), pointer :: maptop
    type(graphics_options) :: graphopt
    TYPE(gtp_equilibrium_data), pointer :: ceq
  subroutine ocplot2B(np,nrv,nlinesep,linesep,pltax,xax,anpax,anpdim,anp,lid,&
       title,filename,graphopt,pform,conditions)
! called from icplot2 to generate the GNUPLOT file after extracting data
! np is number of columns (separate lines), if 1 no labelkey
! nrv is number of values to plot?
! nlinesep is the number of separate lines (index to linesep)
! linesep is the row when the line to plot finishes
! pltax
! xax array of values for single valued axis (T or mu etc)
! anpax=2 if axis with single value is column 2 and (multiple) values in
!         columns 3 and higher
! anp array of values for axis with multiple values (can be single values also)
! lid array with GNUPLOT line types for the different lines
! title Title of the plot
! filename GNUPLOT file name, (also used for pdf/ps/gif file)
! graphopt is graphical option record
! pform is output form (scree/acrobat/postscript/git)
! conditions is a character with the conditions for the diagram
    implicit none
    integer np,anpax,nlinesep
    integer ndx,nrv,linesep(*),anpdim
    character pltax(*)*(*),filename*(*),pform*(*),lid(*)*(*),title*(*)
    character conditions*(*)
    type(graphics_options) :: graphopt
    double precision xax(*),anp(anpdim,*)
    type(graphics_textlabel), pointer :: textlabel
```

5.21 Plotting ternary isothermal sections

The ocplot3 subroutine is called when the user has calculated a diagram with two extensive axis variables and with tie-lines in the plane. The typical case is isothermal ternary sections, for example the macro map10.OCM. In that case the axis variables for calculation are x(cr) and x(ni) and the default plot axis variables are x(*,cr) and x(*,ni). The ocplop3 routine will extract the values of Cr and Ni for the stable phases from each calculated equilibrium and draw a line for each phase. It will also detect endpoints that represent invariant equilibria and draw them. If the user has specified that tie-lines should be plotted separate lines will be drawn in the two-phase region between pairs of Cr and Ni phase content for the stable phases.

The GNUPLOT file will be written by ocplot3B who can also generate output as pdf, ps or gif files.

```
  subroutine ocplot3(ndx,pltax,filename,maptop,axarr,graphopt,pform,ceq)
! special to plot isothermal sections (two columns like x(*,cr) x(*,ni))
! ndx is mumber of plot axis,
! pltax is text with plotaxis variables
! filename is intermediary file (maybe not needed)
! maptop is map_node record with all results
! axarr are axis records
! graphopt is graphics record (should be extended to include more)
! pform is graphics form
! pform is type of output (screen or postscript or gif)
    implicit none
    integer ndx
    character pltax(*)*(*),filename*(*),pform*(*)
    type(map_axis), dimension(*) :: axarr
    type(map_node), pointer :: maptop
    type(graphics_options) :: graphopt
    TYPE(gtp_equilibrium_data), pointer :: ceq
  subroutine ocplot3B(same,nofinv,lineends,nx1,xval,ny1,yval,nz1,zval,plotkod,&
      pltax,lid,filename,graphopt,pform,conditions)
! called by ocplot3 to write the GNUPLOT file for two wildcard columns
! same is the number of lines to plot
! nofinv number of invariants
! lineends array with row numbers where ecah line ends
! nx1 first dimension of xval
! xval 2D matrix with values to plot on x axis
! ny1 first dimension of yval
! yval 2D matrix with values to plot on y axis
! nz1 first dimension of zval
! zval 2D matrix with third point of invariant triangles
! plotkod integer array indicating the type of line (-1 skip line)
! pltax text for axis
! lid array with GNUPLOT line types
! filename is intermediary file (maybe not needed)
! graphopt is graphics option record
! maptop is map_node record with all results
! pform is type of output (screen/acrobat/postscript/gir)
! conditions is a text with conditions for the calculation
    implicit none
    character pltax(*)*(*),filename*(*),pform*(*),lid(nx1,*)*(*),conditions*(*)
    type(graphics_options) :: graphopt
    integer same,plotkod(*),nx1,ny1,nz1,nofinv
    integer lineends(*)
    double precision xval(nx1,*),yval(ny1,*),zval(nz1,*)
```

5.22 Calculate a point in a diagram to list stable phases

This subroutine can be called to generate a text with the names of the stable phases at a point inside a calculated phase diagram,

```
  subroutine calc_diagram_point(axarr,pltax,xxx,xxy,line,ceq)
! calculates the equilibrium for axis coordinates xxx,xxy
! to obtain the set of stable phases
! axarr specifies calculation axis,
! pltax plot axis
! xxx and xxy are axis coordinates for calculating a point
! line is a character where the stable phases at the point is returned
! ceq is the current equilibrium, should be the default with axis conditions
! ONLY COORDINATES FOR CALCULATION AXIS ALLOWED
    implicit none
    type(map_axis), dimension(*) :: axarr
    double precision xxx,xxy
    character line*(*),pltax(*)*(*)
    TYPE(gtp_equilibrium_data), pointer :: ceq
```

5.23 Calculates equilibria for one phase at a time

This subroutine is used for the *step separate* command. The user defines one axis and the Gibbs energy for each non-suspended phase is calculated, one phase at a time, along this axis. It is mainly used to calculate and plot Gibbs energy curves as a function of composition for binary systems but the enthalpy or any other property can be plotted as a function of the axis variable.

```
  subroutine step_separate(maptop,noofaxis,axarr,seqxyz,starteq)
! calculates for each phase separately along an axis (like G curves)
! There can not be any changes of the stable phase ...
! maptop map node record
! noofaxis must be 1
! axarr array of axis records
! seqxyz indices for map and line records
! starteq equilibrium record for starting
    implicit none
    integer noofaxis,seqxyz(*)
    type(map_axis), dimension(noofaxis) :: axarr
    TYPE(gtp_equilibrium_data), pointer :: starteq
    TYPE(map_node), pointer :: maptop
```

5.24 Check an abbreviated phase name

This routine is used when plotting data generated by a step separate command, for example if the user gives the command *step sep* with a composition axis and he has several ordered forms of fcc, he may then plot y(fcc#2,*) versus the composition. Only those constituent fractions for a phase with an abbreviation that fits what the user specified as phase name will be plotted. See macro step3.OCM.

```
  logical function abbr_phname_same(full,short)
! return TRUE if short is a correct abbreviation of full
! This is used in macro step4 to plot fractions in different composition sets
    implicit none
    character*(*) full,short
```

5.25 Extract conditions for a diagram

This subroutine extracts the current set of condition for a plot abd adds them to the title of the diagram. Conditions that can be identified as axis variables are listed as such.

```
  subroutine get_plot_conditions(text,ndx,axarr,ceq)
! extacts the conditions from ceq and removes those that are axis variables
    implicit none
    character text*(*)
    integer ndx
    type(map_axis), dimension(ndx) :: axarr
    type(gtp_equilibrium_data), pointer :: ceq
```

6 Summary

That is all!

References

[1] B. Sundman, U. R. Kattner, M. Palumbo and S. G. Fries, OpenCalphad - a free thermodynamic software, Integrating Materials and Manufacturing Innovation, **4**:1 (2015), open access

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

[3] M. Hillert *Phase Equilibria, Phase Diagrams and Phase Transfromations* Cambr Univ Press (2009) 2nd ed.

[4] H. L. Lukas, S. G. Fries and B. Sundman, *Computational Thermodynamics, the CALPHAD method*, Cambr Univ Press, (2007)

[5] GNUPLOT graphics software, https://wikipedia.org/wiki/gnuplot

[6] J.E. Morall, Scr Met, 18, (1984), 407

[7] H. L. Lukas et al, Calphad (1982)