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

Basic description and documentation

Bo Sundman, September 24, 2016

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) and the Application Software Interface (OCASI).

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 [15Sun1].

This documentation describes the datastructures and software for the calculation of diagrams, for examples using OC and in particular the step, map and plot commands see the user guide or the examples.

Contents

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. Binary phase diagrams can be found in drawn collections collections and there are about 1000 binary systems, ternary and higher order systems that has been assessed thermodynamically, although many of them are not using compatible models and some are not of very high quality.

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 with a single variable condition, for example T, are more useful in order to understand a system at a particular composition.

At equilibrium a multicomponent thermodynamic system may consist of several phases. The Gibbs energy of each phase is modeled independently in the thermodynamic software package as a function of T, P and the phase composition. The equilibrium of a system if found by minimizing the Gibbs energy for the given set of conditions using the thermodynamic models. 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 of the phase changes and also many other properties.

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 conditons. This is achieved by the HMS package of OC described in the paper by Sundman et al.[15Sun2] and in the HMS documentation which is part of the OC software. For a general introduction to equilibrium calculations see the book by Hillert[09Hil] or Lukas et al.[07Luk]. A general presentation of to OpenCalphad project is given in [15Sun1] and the minimization algorithm in [15Sun2].

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 used to follow the lines in the phase diagram.

It is also possible to prescribe that a phase should not be allowed to be stable which makes it possible to calculate metastable phase 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, others may be found in the HMS documentation.

1.2 Graphics driver

The OC software generates a data file for the GNUPLOT[GNUPLOT] 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.

When OC was tested on a MAC the X11 terminal worked well but it did not support the "pause mouse" command for pausing the program after plotting. However it managed to spawn the diagrams so one could look at several figures at the same time, a feature I have not found out how to do on Windows.

1.3 Parallel calculations

Each line in a diagram can be calculated independently and there are provisions, not yet implemented, to do this 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 an 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 show.

3 Phase diagrams

A phase diagram has at least two axis and the lines (or surfaces in 3D and hyper-surfaces 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 of limited interest in general 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

This section will be limited to phase diagrams calculated and plotted in 2D, the most common presentation. Along the lines in such a phase diagram we thus have zero amount of one or more phases and the lines has been called "zero phase fraction" (ZPF) lines by Moral[84Mor].

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[15Sun2] 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 axis condition, except one, 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 and new lines with a different phase set as fix with zero amout will exit from this node point. A line may also terminate at the limit of an axis variable. Whever 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 nodepoint 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 gridminimizer because the results from the previous calculation provides good start values for the next calculation when 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 may 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 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 chise of axis variables, see for example the Ag-Cu system in the macro-OC.pdf.

3.4 Solubility lines

For certain 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.

3.5 Phase diagrams with tie-lines in the plane

A special type of phase diagram has "tie-lines in the plane" and in such a case a single equilibrium calculation provides points on both lines separating this phase region and thess points are 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 single phase regions are separated by two-phase regions and the lines limiting the compositions of this two-phase equilibrium are in the same plane in the diagram.

3.6 Unary phase diagrams

For unary diagrams the only variables are T and P, with 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 it 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.

3.7 Binary phase diagrams

Also 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 limits of a single phase region. Thus one can apply many many relations between the lines, like the lever rule, which is not generally applicable in multicomponent systems.

3.8 Isothermal sections of ternary phase diagrams

This type of diagram reprented some special difficulties as the axis variables are two normallized 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).

There is still no way to plot isothermal sections in a triangular diagram.

It is possible but rather awkward to calculate for example a binary $T-x$ diagram using the entropy or enthalpy as one axis that is not rarely done as it is more complicated. If one is interested in this kind of diagram it is easier to calculate it with T as axis and then plot it with entalpy of the different phases, "HM(*)" as one axis.

3.9 Multicomponent phase diagrams

For multicomponent systems we normally do not have tie-lines in the plane, unless all conditions except one are potentials. But the rule is the same that the lines separate areas with different sets of stable phases. In multicomponent systems with two or more extensive variables as conditions, there is a simple rule that the number of stable phases always changes by unity for each line one cross. It can be +1 or -1 but never 2 or more. If there is an axis that is a potential, like T, and only a few elements, there is a small chance that one has an invariant equilibrium in the plane of the diagram and 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 other lines end at this line), but the phases stable on either side are not the same. This is the same as crossing an invariant line in a binary T-x diagram.

The reasoning here is to explain that it is quite easy to calculate a multicomponent phase diagram, we just follow all lines that represent zero amount of a phase for the given set of conditions. But it can be quite difficult to plot the diagram as will be discussed in this documentation in the next release of this software.

The mapping of a phase diagram is thus very similar to the stepping used for a property diagram. The second axis (or third axis etc) in the phase diagram has been replaced by a fixed phase and then we just step in the other axis keeping these phases fixed. 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 one wants to disappear because there are several ZPF lines meeting at such a point.

4 Basic data structures for STEP and MAP

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 claculation.

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 calle "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 equilibrum number for an axis change
     integer axandir,axchange
! more is 1 while following the line, 0 for last equilbrium, -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.

Also when the diagram is very large and one runs out of memory and must save intermediate results on a file, the equilibrium at all currently unfinished lines are stored in a node record, then the memory is cleared and mapping continues (not yet implemented).

```
  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 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 elesewhere.

```
  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 whic 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 all but one axis variables by a fix phase for phase diagram calculations. At present only two axos 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 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 line this 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 everyting. For phase diagrams it is possible that the node point where the line ends has already been creating comming from another line. In such a case no new node is created and instead one of the already existing exit lines are removed. This also avoids eternal loops when lines are conneced 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 line when the line changes direction. This can happe 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 a´xis and calculate a new equilibrium along a line.

```
  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 when the set of phases are changing.

```
  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 multicompnent 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 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 saveing equilibria

Called by map_setup to create a record where 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 reinitiated 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 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 nodepoint 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 handel 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 calculating an equilibrium by decreasing the step.

```
  subroutine map_halfstep(halfstep,axvalok,mapline,axarr,ceq)
! Used when an error calculating a normal step or a node point
! take back the last sucessfully 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 cucessfully 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 Lists stored equilibria

For the user interface there is one subrotine to list all node points and lines with calculated equilibra and another to amend these. For the amendment the user cal select to inactive (suppress) and line or activate an inactivated lines. During mapping a line may be 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 most diagrams

There are two subroutine to generate a graphical output from the step and map calculations. For a phase diagram the plot axis by default are the same as those used to generate the diagram, however for an extensive variable for a diagram with the tie-lines in the plane this is modified by including a wildcard for the phase, thus if the binary Ag-Cu diagrem (in macro map1.OCM) the calculated axis variable is x(cu) but the plotted axis is x(*,cu) meaning that the Cu content of stable phases are plotted.

For phase diagrams without tie-lines in the plane one will plot the same extensive variable as used for the calculation.

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

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

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

Inside ocplot2 the axis variable values selected are extracted from the caöculated equilibria and stored in two variables. Some information about invariant lines are also added. Then the ocplo2B 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 add labels to the diagram and it is possible to calculate the equilibrium set of phases for a point inside the same coordinates as for the diagram and use these as label.

```
  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

This subroutine is called when the user has calculated a diagram with two extensive variable as axis. The typical case for that is isothermal ternary sections, for example the macro map10.OCM. In that a case the axis variables are x(cr) and x(ni) and the plot axis 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 as specified that tie-lines should be plotted separate lines will be drawn in the two-phase region between pairs of Cr and Ni content from the stable phases.

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

The user may also add labels to the diagram and it is possible to calculate the equilibrium set of phases for a point inside the same coordinates as for the diagram and use these as label.

```
  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 phase 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 at a time along this axis. It is mainly used to calculate Gibbs energy curves 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 if the user asks for y(fcc#2,*) to be plotted and all composition sets of fcc has been calculated. Only those constituent fractions for a phase with an abbreviation that fits what the user specified will be plotted.

```
  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 that to the title. 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

[84Mor] J.E. Morall, Scr Met, 18, (1984), 407

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

[09Hil] M Hillert *Phase Equilibria, Phase Diagrams and Phase Transfromations* Cambr Univ Press (2009) 2nd ed.

[15Sun1] B Sundman, U Kattner, M Palumbo and S G Fries, OpenCalphad - a free thermodynamic software, Integrating Materials and Manufacturing Innovation, **4**:1 (2015), open access

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

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