Step Map and Plot (SMP) in Open Calphad (OC)

Basic description and documentation

Bo Sundman, January 3, 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 minimizer (HMS) and the software interface (OCTQ).

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

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 almost 1000 binary 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 modelled 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 generate any kind of property and phase diagrams.

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 welldefined equilibrium. But the user may have conditions on volume, V, enthalpy, H or the chemical potentials, MU of some components. In such cases the specified equilibrium may not exist.

2 Property diagrams

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 inrement until it has calculated equilibria to the maximum and to the minimum axis values. The calculated equilibria will be stored in a separate data array.

Whever 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 at 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.

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 as most real alloys have more then 3 components, a phase diagram for such a system cannot be drawn in full. This section will be limited to phase diagrams plotted in 2D, the most common presentation and thus along the lines in such a phase diagram we have zero amount of one or more phases, the lines can be called "zero phas fraction" lines.

To obtain a 2D presentation of a multicomponent phase diagram we must make various kinds of sections or projections. For a 2D presentation of a projection some lines which seem to intersect may not be in the same plane. Thus it may 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 for this an equilibrium along an axis specified for the phase diagram. In fact one can follow such a line using more than 2 axis as the method is to replace each axis condition, except one, with a fix phase.

While following such a line a new phase may become stable, or an existing phase disappear, and the phase diagram software will then generate a node point and new lines with a different phase with zero amount will exit from this node point. A line may also terminate at the limit of an axis variable.

When following a line the SMP package does not use the gridminimizer as the results from the previous calculation provies good start values for the next calculation when there is just a small change in one condition. But when calculating a node point the grid minimizer should be used to test that one has not walked into a miscivility gap. Also along a line one should test regularly at each 10'th or 20'th step that there is no problem by callinf the grid minimizer. 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 and 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 (normallized) extensive variables used as conditions. One should also realize that the

phase diagram is independent of the size of the system. We may also draw the phase diagram using other properties than those used as conditions in order to calculate the lines in the diagram.

A special type of phase diagram has "tie-lines in the plane" which means that the lines separating different phase regions are the result of a single equilibrium calculation. A tie-line is 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 limiting compositions of this two-phase equilibrium are in the same plane in the diagram.

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.

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

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 multicomponents with two or more exiensuve 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, but the phase stable are not the same. This is the same as crossing an invariant line in a binary T-x diagram.

The reasoning here is just to explain that it is quite easy to calculate a 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 them 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 in the phase diagram has been replaced by a fixed phase and then we just step in the other axis keeping this phase fixed. During the mapping we may have to change with 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.

4 Basic data structures for STEP and MAP

4.1 Dependencies

As the step and map procedures depend on the equilibrium calculation software many of these data structures, in particular the gtp_equilibrium_data record type. This is described in the GTP documentation, others may be found in the HMS documentation.

4.2 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
     integer number_of_equilibria,first,last,lineid,done,nfixphases
! 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 curren taxis 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
! problem is nonzero if map_problems has been called
     integer more,termerr,problems
! 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 valies 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
```

4.3 The node record

This record contain the equilibrium at a node points where several lines meet.

```
  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
```

4.4 The node record

This record contain a description of the axis set for the mapping of the phase diagram.

```
  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 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
```

4.5 The save record

This record contains calculated equilibria.

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

4.6 The graphics option record

The axis and other options for plotting the diagram are stored here.

```
  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 jpeg)
     integer rangedefaults(3)
     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
! 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
! many more options can easily be added when desired, linetypes etc
  end TYPE graphics_options
```

5 Subroutines and functions

5.1 The setup routine for step and mapping

This routine organizes the STEP or MAP procedure.

```
  subroutine map_setup(maptop,nax,axarr,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
! starteq is an equilibrium data record, if there are more start equilibria
! they are linked using the ceq%next index
    implicit none
    integer nax,nsp
    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

```
  subroutine map_startpoint(maptop,nax,axarr,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
!
    implicit none
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(map_node), pointer :: maptop
    integer nax
    integer inactive(*)
    type(map_axis), dimension(nax) :: axarr
```

5.3 Changes the variable used for mapping

If the slope of the axis variables changes this routine can change the variable used for the equilibrium calculation.

```
  subroutine map_replaceaxis(meqrec,axactive,ieq,nax,axarr,tmpline,&
       inactive,ceq)
! replace an axis condition with a fix phase
    implicit none
! 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
! tmpline is to transfer some line data to calling routine
! inactive is not really used.
    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 line to follow

```
  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 on the line

```
  subroutine map_store(mapline,axarr,nax,saveceq)
! store a calculated equilibrium
    implicit none
    integer nax
    type(map_line), pointer :: mapline
    type(map_axis), dimension(nax) :: axarr
    type(map_ceqresults), pointer :: saveceq
```

5.6 Storing the last equilibrium along a line

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

5.7 Changing the axis for next equilbrium calculation

During mapping it may sometimes be necessary to change the asix variable used for moving along the line.

```
  subroutine map_changeaxis(mapline,nyax,oldax,nax,axarr,axval,bytax,ceq)
! changes the axis with active condition to nyax
    type(map_line), pointer :: mapline
    type(gtp_equilibrium_data), pointer :: ceq
    type(map_axis), dimension(nax) :: axarr
    logical bytax
! nax is number of axis, nyax is new axis with active condition
    integer nyax,nax,oldax
! the value to set as condition on new axis
    double precision axval
  subroutine map_force_changeaxis(maptop,mapline,meqrec,nax,axarr,ceq)
! force change of axis with active condition.  Works only with 2 axis.
! (and for tie-line not in plane ??).  Similar to map_changeaxis ...
    implicit none
! number of axis, also in maptop record
    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
```

5.8 Calculating the next equilibrium along a line

This is the normal routine to 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 phase fix condition
! calculate the next equilibrium along a line.  New phases can appear.
! axis with active condition can change and the direction.
    implicit none
! number of axis, redundant as also in maptop record
    integer nax
    type(map_node), pointer :: maptop
    type(map_line), pointer :: mapline
    type(meq_setup) :: meqrec
! phr is included just for debugging to see phase amounts
    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.
    implicit none
    integer irem,iadd
! am am wondering if pointer is necessary??
    type(map_node), pointer :: maptop
    type(map_line), pointer :: mapline
! Note changes in meqrec is local, not copied to mapline%meqrec!!!
```

```
    type(meq_setup) :: meqrec
    type(map_axis), dimension(*) :: axarr
    type(gtp_equilibrium_data), pointer :: ceq
```

5.10 Creates a node with several lines

After calculating the node point this routine determines the sxits for new lines.

```
  subroutine map_newnode(mapline,meqrec,maptop,axval,lastax,axarr,phfix,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 it tie-lines in the plane all lines do not have to be calculated
! NOTE: meqrec not the same as mapline%meqrec !! ??
    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
! axval is axis value which was attemped to calculate,
    double precision axval
```

5.11 Reserves a record to save a calculated equilibrium

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

5.12 Finds a line to be calculated

Searches all node points for lines to be calculated.

```
  subroutine map_findline(maptop,axarr,mapfix,mapline)
! must be THREADPROTECTED
! Find a map_line record to be calculated from maptop
! already been found and if so eliminate a line record.  Otherwise
```

```
    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

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

```
  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
    integer nax
    type(map_axis), dimension(nax) :: axarr
    type(gtp_equilibrium_data), pointer :: ceq
```

5.15 Checks if the node point is an invariant equilibrium

A nodepoint from an invariant equilibria have more lines that exit than normal nodes.

```
  integer function invariant_equilibrium(lines,mapnode)
! Only called for tie-lines not in plane.  If tie-lines in plane then all
! nodes are invariants.
    integer lines
    type(map_node), pointer :: mapnode
```

5.16 Tries to handel problems when mapping

This tries various way to handle problems during STEP and MAP.

```
  subroutine map_problems(maptop,mapline,axarr,xxx,typ)
! jump here for different problems
```

```
    integer typ
    type(map_node), pointer :: maptop
    type(map_line), pointer :: mapline
    type(map_axis), dimension(*) :: axarr
    double precision xxx,yyy
```

5.17 Using smaller steps when there are convergence problems

This subroutine is used when there is a convergence problem calculating an 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 sucessfully calculated axis value and take smaller step
! possibly one should also restore the ceq record.
    implicit none
    integer halfstep
    double precision axvalok
    TYPE(gtp_equilibrium_data), pointer :: ceq
    TYPE(map_line), pointer :: mapline
    type(map_axis), dimension(*) :: axarr
```

5.18 Lists stored equilibria

```
  subroutine list_stored_equilibria(kou,axarr,maptop)
! list all nodes and lines from step/map
! maybe allow some delete/amend later ...
    integer kou
    type(map_node), pointer :: maptop
    type(map_axis), dimension(*) :: axarr
```

5.19 Plots calculated diagrams

```
  subroutine ocplot2(ndx,pltax,filename,maptop,axarr,graphopt,pform,ceq)
! 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
! pform is type of output (screen or postscript or jpeg)
    implicit none
    integer ndx
```

```
    character pltax(*)*(*),filename*(*),pform*(*)
    type(map_axis), dimension(*) :: axarr
    type(map_node), pointer :: maptop
    type(graphics_options) :: graphopt
```

5.20 Calculates equilibria for one phase at a time

Needed to calculate for example Gibbs energy curves.

```
  subroutine step_separate(maptop,noofaxis,axarr,starteq)
! calculates for each phase separately along an axis (like G curves)
! There can not be any changes of the stable phase ...
    implicit none
    integer noofaxis
    type(map_axis), dimension(noofaxis) :: axarr
    TYPE(gtp_equilibrium_data), pointer :: starteq
    TYPE(map_node), pointer :: maptop
```

6 Summary

That is all!

References

[15Sun2] B Sundman et al, Comp. Mat. Sci (2015)