The Application Software Interface for the Open Calphad software with some examples

Bo Sundman, Matthias Stratman, Ursula R Kattner,
Mauro Palumbo and Suzana G Fries

INSTN CEA Saclay, France, NIST USA, SISSA, Trieste, Italy and Ruhr University Bochum, Germany

TMS February 2016

Background

The Calphad technique has been very successful providing important information about the stable state of inorganic materials including gases, alloys, semiconductors, ceramics, slags, nuclear materials etc. An additional advantage is that the same data can also be used to provide essential information for simulation of phase transformations.

Background

The Calphad technique has been very successful providing important information about the stable state of inorganic materials including gases, alloys, semiconductors, ceramics, slags, nuclear materials etc. An additional advantage is that the same data can also be used to provide essential information for simulation of phase transformations.
In 2012 a few scientists took the initiative to start developing an open Calphad software for thermodynamic applications, simply called Open Calphad, OC, to provide a free software for multicomponent equilibrium calculations.

Background

The Calphad technique has been very successful providing important information about the stable state of inorganic materials including gases, alloys, semiconductors, ceramics, slags, nuclear materials etc. An additional advantage is that the same data can also be used to provide essential information for simulation of phase transformations.
In 2012 a few scientists took the initiative to start developing an open Calphad software for thermodynamic applications, simply called Open Calphad, OC, to provide a free software for multicomponent equilibrium calculations.
In this presentation I will provide an short description of the OC Application Software Interface (OCASI) and some examples how this can be used.

Background

The Calphad technique has been very successful providing important information about the stable state of inorganic materials including gases, alloys, semiconductors, ceramics, slags, nuclear materials etc. An additional advantage is that the same data can also be used to provide essential information for simulation of phase transformations.
In 2012 a few scientists took the initiative to start developing an open Calphad software for thermodynamic applications, simply called Open Calphad, OC, to provide a free software for multicomponent equilibrium calculations.
In this presentation I will provide an short description of the OC Application Software Interface (OCASI) and some examples how this can be used.
In addition to be free OC can also be run in parallel using the OpenMP library which can reduce the calculation times for a simulation more than one order of magnitude.

Content

- Data structures and algorithms

Content

- Data structures and algorithms
- Two examples
  - Phase field simulation of diffusion in a 4 component steel,

Content

- Data structures and algorithms
- Two examples
  - Phase field simulation of diffusion in a 4 component steel,
  - Simulation of solidification and homogenization of a 9 component aluminium alloy,

Content

- Data structures and algorithms
- Two examples
  - Phase field simulation of diffusion in a 4 component steel,
  - Simulation of solidification and homogenization of a 9 component aluminium alloy,
- OCASI software interface to application programs
  - initialization and reading a database

Content

- Data structures and algorithms
- Two examples
  - Phase field simulation of diffusion in a 4 component steel,
  - Simulation of solidification and homogenization of a 9 component aluminium alloy,
- OCASI software interface to application programs
  - initialization and reading a database
  - phases and phase tuples

Content

- Data structures and algorithms
- Two examples
  - Phase field simulation of diffusion in a 4 component steel,
  - Simulation of solidification and homogenization of a 9 component aluminium alloy,
- OCASI software interface to application programs
  - initialization and reading a database
  - phases and phase tuples
  - state variables and conditions

Content

- Data structures and algorithms
- Two examples
  - Phase field simulation of diffusion in a 4 component steel,
  - Simulation of solidification and homogenization of a 9 component aluminium alloy,
- OCASI software interface to application programs
  - initialization and reading a database
  - phases and phase tuples
  - state variables and conditions
  - extracting results

Content

- Data structures and algorithms
- Two examples
  - Phase field simulation of diffusion in a 4 component steel,
  - Simulation of solidification and homogenization of a 9 component aluminium alloy,
- OCASI software interface to application programs
  - initialization and reading a database
  - phases and phase tuples
  - state variables and conditions
  - extracting results
  - C++ interface

Content

- Data structures and algorithms
- Two examples
  - Phase field simulation of diffusion in a 4 component steel,
  - Simulation of solidification and homogenization of a 9 component aluminium alloy,
- OCASI software interface to application programs
  - initialization and reading a database
  - phases and phase tuples
  - state variables and conditions
  - extracting results
  - C++ interface
- Summary and availability

Thermodynamic data

In the Calphad technique the Gibbs energy for each phase is modeled as a function of T, P and its constitution.

$$G_M^\alpha = G_M^\alpha(T, P, y_i)$$

Thermodynamic data

In the Calphad technique the Gibbs energy for each phase is modeled as a function of T, P and its constitution.

$$G_M^\alpha = G_M^\alpha(T, P, y_i)$$

Each phase can have a different model with sublattices and molecules, ions or vacancies as constituents. The thermodynamic databases contain model parameters for the Gibbs energy of formation of many different phases like compounds, endmembers of solution phases, interactions etc.

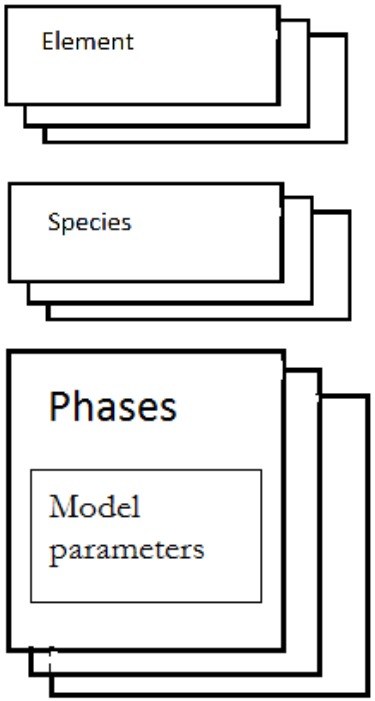
Thermodynamic data

In the Calphad technique the Gibbs energy for each phase is modeled as a function of T, P and its constitution.

$$G_M^\alpha = G_M^\alpha(T, P, y_i)$$

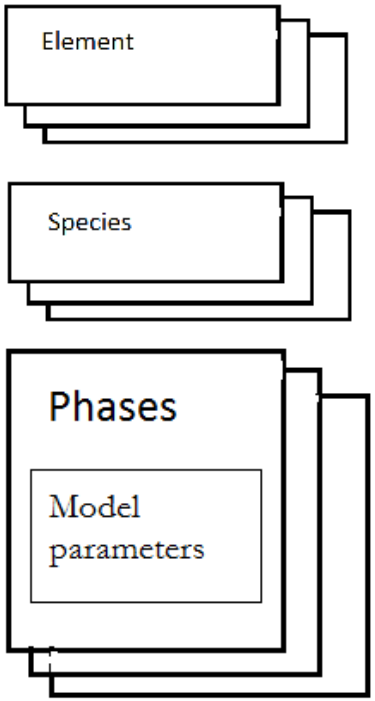
Each phase can have a different model with sublattices and molecules, ions or vacancies as constituents. The thermodynamic databases contain model parameters for the Gibbs energy of formation of many different phases like compounds, endmembers of solution phases, interactions etc. The equilibrium state is found by minimizing the Gibbs energy varying the set of stable phases and their constitutions for the given set of conditions.

Static and dynamic memory



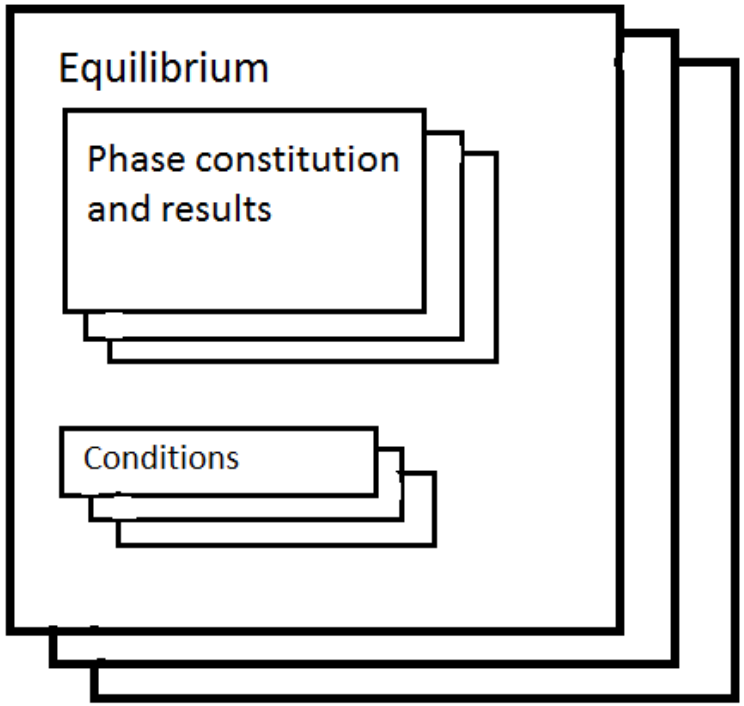
The data for elements, species and model parameters are independent on the external conditions on the system

Static and dynamic memory



The data for elements, species and model parameters are independent on the external conditions on the system

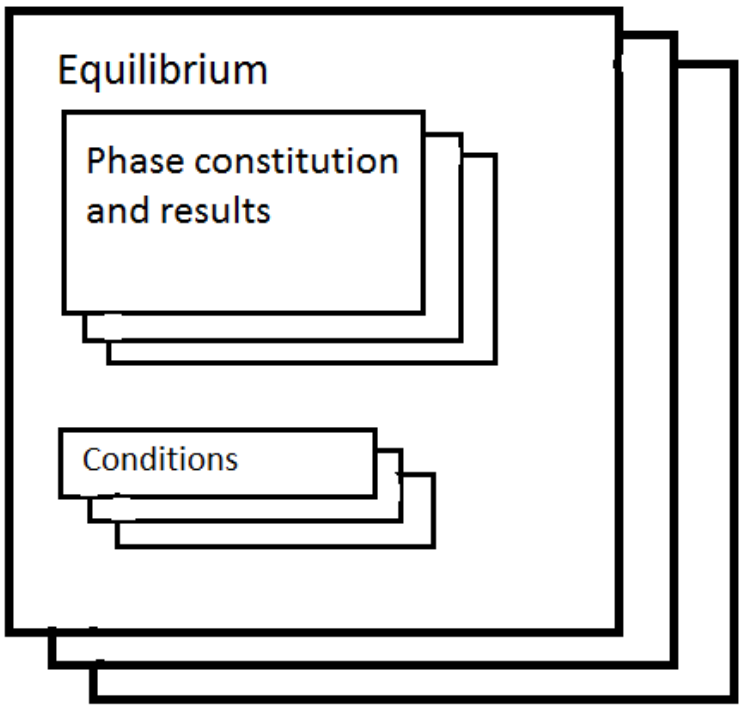
Dynamic data



The amount and constitution of the phase varies with the external conditions like temperature and composition and are stored in the equilibrium record.

Parallel execution

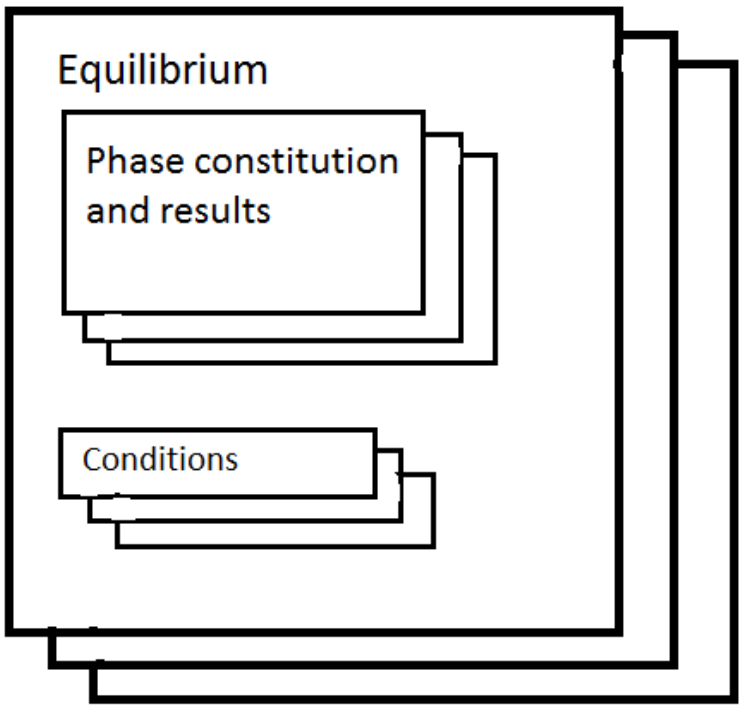
Dynamic data



In addition to the dynamic phase data the current conditions are also stored in the equilibrium record. In simple cases there is just one equilibrium record but each is independent and several can be executed in parallel using OpenMP.

Parallel execution

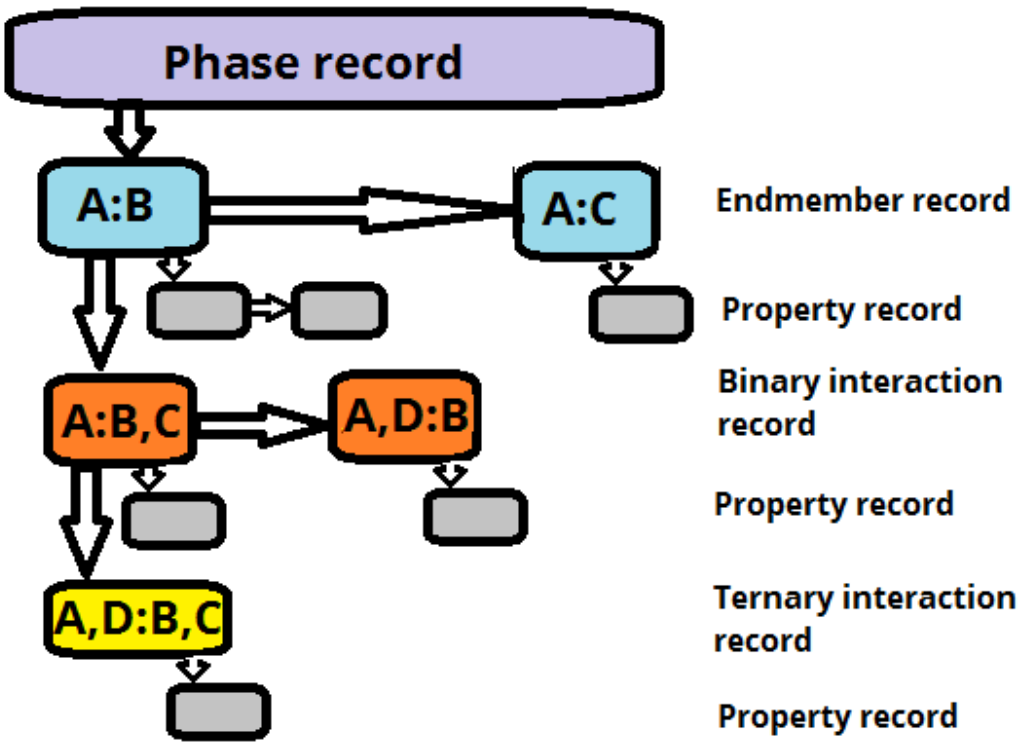
Dynamic data



In addition to the dynamic phase data the current conditions are also stored in the equilibrium record. In simple cases there is just one equilibrium record but each is independent and several can be executed in parallel using OpenMP. This is useful for application software like assessment of model parameters using experimental and theoretical data and for phase field simulations when each gridpoint can have its own equilibrium record with a local set of conditions.

Separation of static and dynamic data

The Gibbs energy for a phase the with the model $(A,C)_a(B,D)_b$ may be:



This represent the equation

$$G = y_A y_B({}^\circ G_{A:B} + y_C(L_{A:B,C} + y_D L_{A,D:B,C}) + y_D L_{A,D:B}) + y_A y_C \, {}^\circ G_{A:C}$$

The model parameters are in the static data area and the constituent fractions, y_i and the results, G are in the dynamic data structure.

Software, modules

The OC software is modularized in the following way.

- Module with different thermodynamic models to calculate the molar Gibbs energy and partial derivatives with respect to T, P, Y for each phase in the system.

Software, modules

The OC software is modularized in the following way.

- Module with different thermodynamic models to calculate the molar Gibbs energy and partial derivatives with respect to T, P, Y for each phase in the system.
- Module for single equilibrium calculation for the flexible external conditions on T, P, overall composition, chemical potentials, specification of fix phases etc. This module finds the equilibrium for a given set of conditions varing the amount of the phases and their constitutions.

Software, modules

The OC software is modularized in the following way.

- Module with different thermodynamic models to calculate the molar Gibbs energy and partial derivatives with respect to T, P, Y for each phase in the system.
- Module for single equilibrium calculation for the flexible external conditions on T, P, overall composition, chemical potentials, specification of fix phases etc. This module finds the equilibrium for a given set of conditions varing the amount of the phases and their constitutions.
- Module for calculation and plotting of diagrams.
- Command line user interface.
- A module for assessment of model parameters using experimental and theoretical data.

Software, modules

The OC software is modularized in the following way.

- Module with different thermodynamic models to calculate the molar Gibbs energy and partial derivatives with respect to T, P, Y for each phase in the system.
- Module for single equilibrium calculation for the flexible external conditions on T, P, overall composition, chemical potentials, specification of fix phases etc. This module finds the equilibrium for a given set of conditions varing the amount of the phases and their constitutions.
- Module for calculation and plotting of diagrams.
- Command line user interface.
- A module for assessment of model parameters using experimental and theoretical data.
- Application software interface for using OC together with kinetic and other data.

Calculating the equilibrium state

The equilibrium at fixed temperature, pressure and composition is given by a minimum in the total Gibbs energy of the system. The thermodynamic models describe the molar Gibbs energy, G_M^{φ} and the total energy for the system is:

$$G(T, P, N_i) = \sum_{\varphi} \aleph^{\varphi} G_M^{\varphi}(T, P, y_i)$$

where $\aleph^{\varphi}$ is the amount of the stable phases.

Calculating the equilibrium state

The equilibrium at fixed temperature, pressure and composition is given by a minimum in the total Gibbs energy of the system. The thermodynamic models describe the molar Gibbs energy, G_M^{φ} and the total energy for the system is:

$$G(T, P, N_i) = \sum_{\varphi} \aleph^{\varphi} G_M^{\varphi}(T, P, y_i)$$

where $\aleph^{\varphi}$ is the amount of the stable phases.
If one has conditions like volume, chemical potentials or fixed phases one can modify the Gibbs energy by the use of Lagrange multipliers.

Calculating the equilibrium state

The equilibrium at fixed temperature, pressure and composition is given by a minimum in the total Gibbs energy of the system. The thermodynamic models describe the molar Gibbs energy, G_M^{φ} and the total energy for the system is:

$$G(T, P, N_i) = \sum_{\varphi} \aleph^{\varphi} G_M^{\varphi}(T, P, y_i)$$

where $\aleph^{\varphi}$ is the amount of the stable phases.
If one has conditions like volume, chemical potentials or fixed phases one can modify the Gibbs energy by the use of Lagrange multipliers.
The minimization procedure in OC is based on a proposal by Hillert in 1981 and its implementation in OC was published last year.
To ensure we find the global equilibrium there is an initial, optional, grid minimizer to calculate start values.

Application software interface, OCASI

The OC software interface follows the TQ standard proposed in 1994 and which is used by Thermo-Calc and ChemApp in slightly different forms. The OC variant is called OCASI.

- The OC software can be used by applications written in Fortran or C++ using the iso-C standard.

Application software interface, OCASI

The OC software interface follows the TQ standard proposed in 1994 and which is used by Thermo-Calc and ChemApp in slightly different forms. The OC variant is called OCASI.

- The OC software can be used by applications written in Fortran or C++ using the iso-C standard.
- The application program can:
  - read from a database,

Application software interface, OCASI

The OC software interface follows the TQ standard proposed in 1994 and which is used by Thermo-Calc and ChemApp in slightly different forms. The OC variant is called OCASI.

- The OC software can be used by applications written in Fortran or C++ using the iso-C standard.
- The application program can:
  - read from a database,
  - set conditions in a very flexible way,

Application software interface, OCASI

The OC software interface follows the TQ standard proposed in 1994 and which is used by Thermo-Calc and ChemApp in slightly different forms. The OC variant is called OCASI.

- The OC software can be used by applications written in Fortran or C++ using the iso-C standard.
- The application program can:
  - read from a database,
  - set conditions in a very flexible way,
  - calculate several equilibria in parallel using the OpenMP library,

Application software interface, OCASI

The OC software interface follows the TQ standard proposed in 1994 and which is used by Thermo-Calc and ChemApp in slightly different forms. The OC variant is called OCASI.

- The OC software can be used by applications written in Fortran or C++ using the iso-C standard.
- The application program can:
  - read from a database,
  - set conditions in a very flexible way,
  - calculate several equilibria in parallel using the OpenMP library,
  - retrieve calculated results.

Application software interface, OCASI

The OC software interface follows the TQ standard proposed in 1994 and which is used by Thermo-Calc and ChemApp in slightly different forms. The OC variant is called OCASI.

- The OC software can be used by applications written in Fortran or C++ using the iso-C standard.
- The application program can:
  - read from a database,
  - set conditions in a very flexible way,
  - calculate several equilibria in parallel using the OpenMP library,
  - retrieve calculated results.
- A number of applications programs are provided with the source code.

Example 1 of use of OCASI, simulating uphill diffusion

At Ruhr University in Bochum, Germany, the ICAMS group has implemented OC in their free software for phase field simulations, Open Phase. As a test they have simulated the classical Darken experiment for "uphill diffusion" of carbon in two iron bars with almost the same carbon content but different Si and Mn content.

mass%	C	Si	Mn	Fe
left bar	0.49	3.8	0.2	rest
right bar	0.58	0.2	6.4	rest

Example 1 of use of OCASI, simulating uphill diffusion

At Ruhr University in Bochum, Germany, the ICAMS group has implemented OC in their free software for phase field simulations, Open Phase. As a test they have simulated the classical Darken experiment for "uphill diffusion" of carbon in two iron bars with almost the same carbon content but different Si and Mn content.

mass%	C	Si	Mn	Fe
left bar	0.49	3.8	0.2	rest
right bar	0.58	0.2	6.4	rest

Darken joined these steels together and heat treated them during 10 days at 1323 K. Carbon is a fast diffusing element but the Si and Mn will not diffuse very far at even at that temperature.

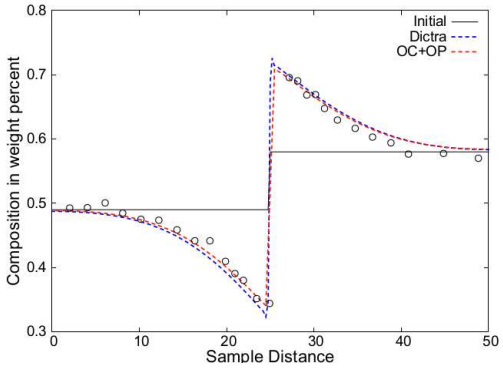
Example 1 of use of OCASI, simulating uphill diffusion

At Ruhr University in Bochum, Germany, the ICAMS group has implemented OC in their free software for phase field simulations, Open Phase. As a test they have simulated the classical Darken experiment for "uphill diffusion" of carbon in two iron bars with almost the same carbon content but different Si and Mn content.

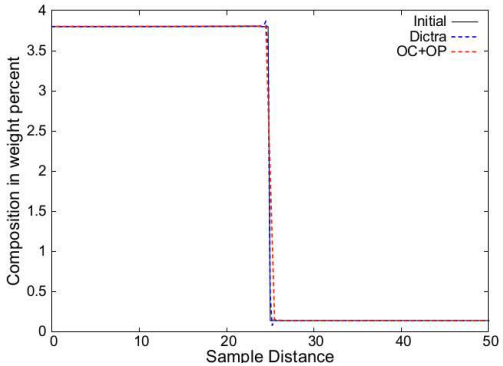
mass%	C	Si	Mn	Fe
left bar	0.49	3.8	0.2	rest
right bar	0.58	0.2	6.4	rest

Darken joined these steels together and heat treated them during 10 days at 1323 K. Carbon is a fast diffusing element but the Si and Mn will not diffuse very far at even at that temperature.
Si and Mn has opposite effects on the carbon activity, it is increased by Si and decreased by Mn.

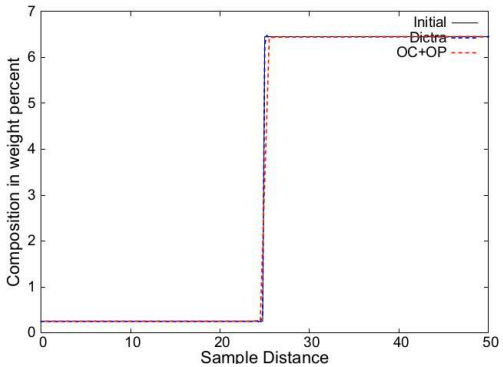
Example 1 of use of OCASI, simulating uphill diffusion



C



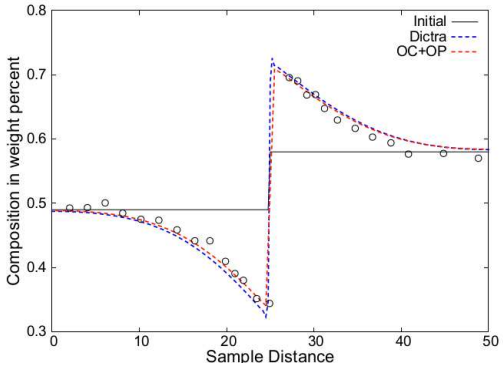
Si



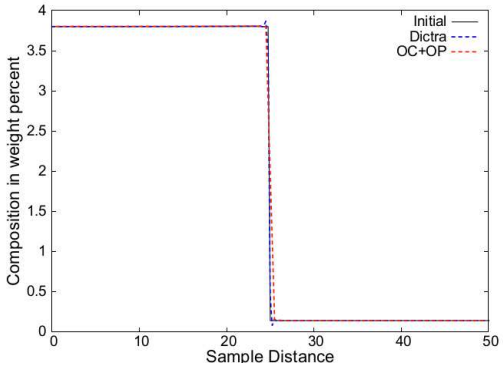
Mn

These figures show the simulated composition profiles across the joint compared with experimental data.

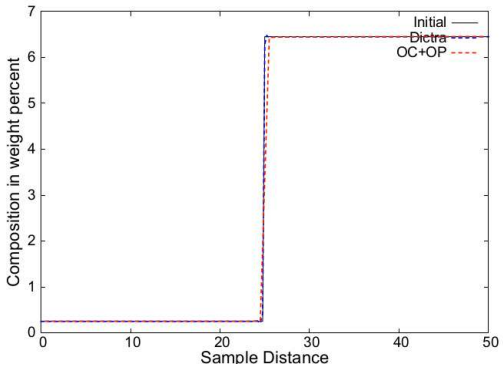
Example 1 of use of OCASI, simulating uphill diffusion



C



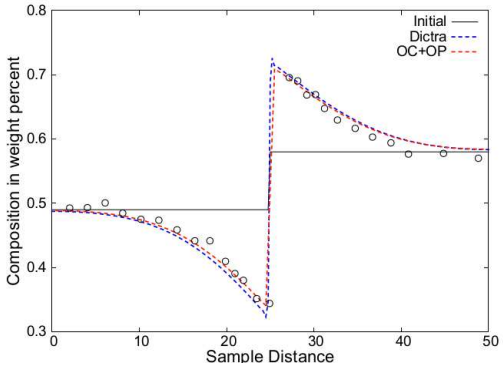
Si



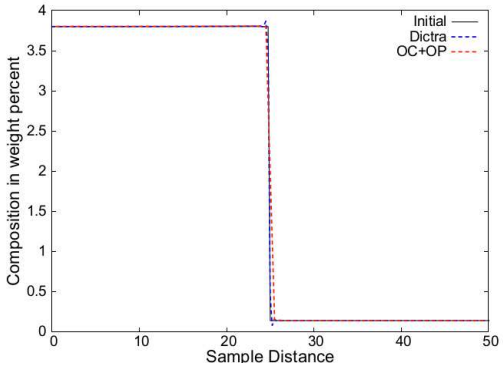
Mn

These figures show the simulated composition profiles across the joint compared with experimental data. The Carbon diffusion has increased the C content in the low Si steel because the carbon activity is lower there. Darken wanted to show that the diffusion is activity controlled.

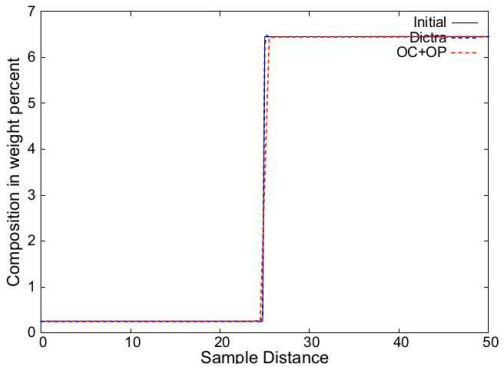
Example 1 of use of OCASI, simulating uphill diffusion



C



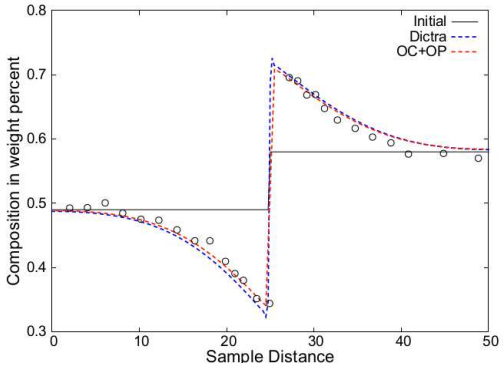
Si



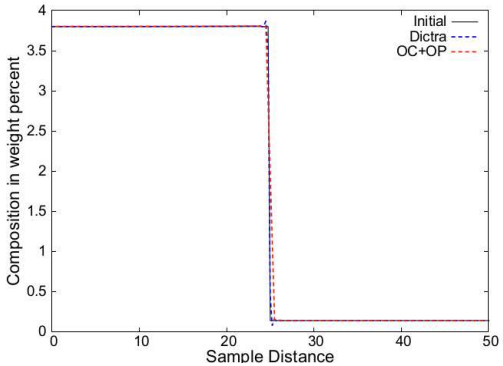
Mn

These figures show the simulated composition profiles across the joint compared with experimental data. The Carbon diffusion has increased the C content in the low Si steel because the carbon activity is lower there. Darken wanted to show that the diffusion is activity controlled. Note there is no fitting made to the experimental data, the curves are calculated using independently assessed thermodynamic and kinetic data.

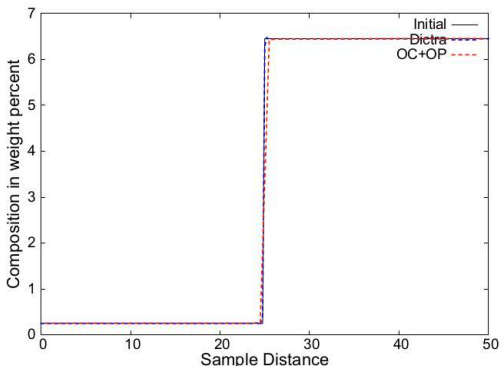
Example 1 of use of OCASI, simulating uphill diffusion



C



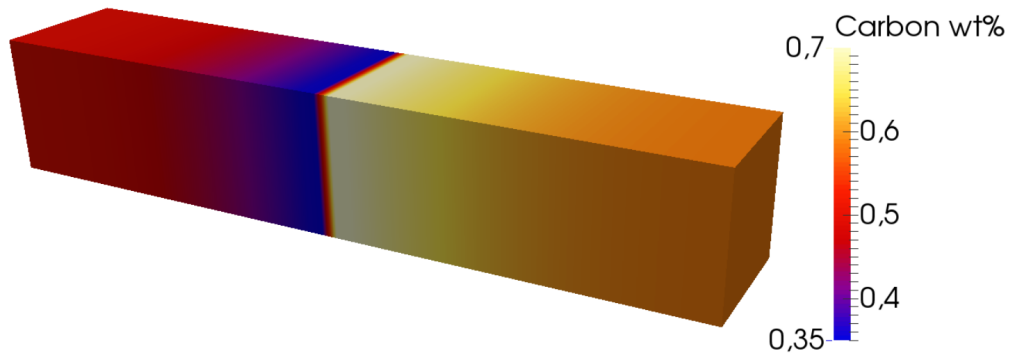
Si



Mn

These figures show the simulated composition profiles across the joint compared with experimental data. The Carbon diffusion has increased the C content in the low Si steel because the carbon activity is lower there. Darken wanted to show that the diffusion is activity controlled. Note there is no fitting made to the experimental data, the curves are calculated using independently assessed thermodynamic and kinetic data.

This is the simulated volume, the color show the C concentration.



Example 2, solidification and homogenization

An aluminium company in France has converted a software they used for simulating solidification and homogenization to use OCASI.

Example 2, solidification and homogenization

An aluminium company in France has converted a software they used for simulating solidification and homogenization to use OCASI.
For an alloy with the composition

Si	Fe	Cu	Mn	Mg	Zn	Ti	Zr	Al
0.06	0.06	1.8	0.1	2.3	8.1	0.06	0.06	rest

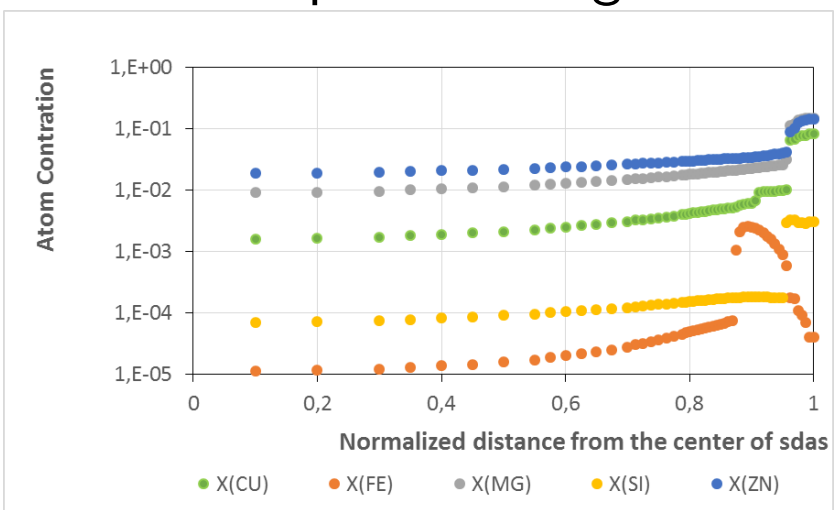
Their own software took about 3.5 days to simulate the solidification and several subsequent homogenization steps.

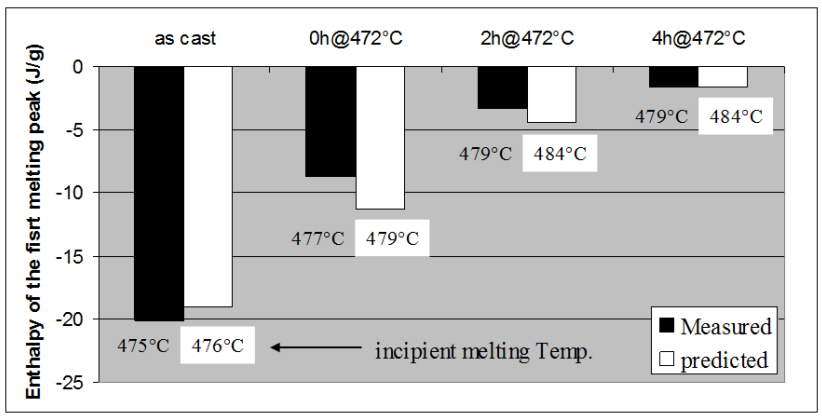
Example 2, solidification and homogenization

An aluminium company in France has converted a software they used for simulating solidification and homogenization to use OCASI.
For an alloy with the composition

Si	Fe	Cu	Mn	Mg	Zn	Ti	Zr	Al
0.06	0.06	1.8	0.1	2.3	8.1	0.06	0.06	rest

Their own software took about 3.5 days to simulate the solidification and several subsequent homogenization steps.





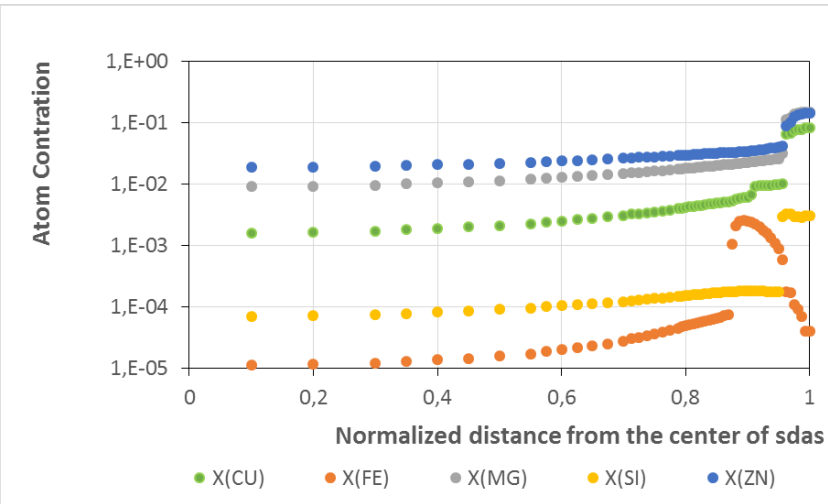
The left hand figure show the concentration profiles across a dendrite after solidification, the right the heat evolved during homogenization.

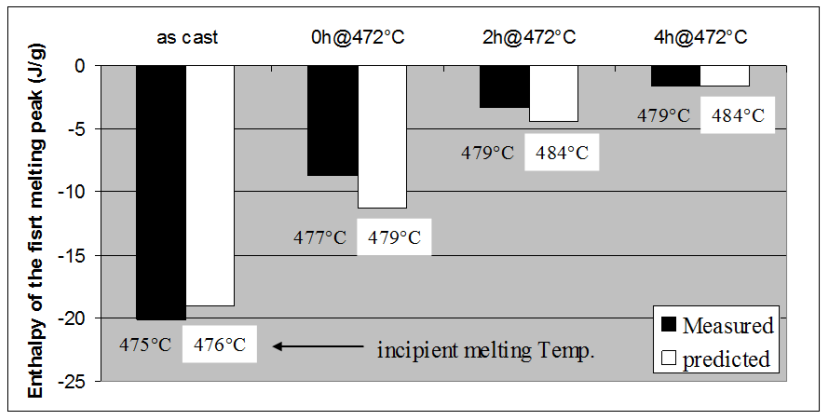
Example 2, solidification and homogenization

An aluminium company in France has converted a software they used for simulating solidification and homogenization to use OCASI.
For an alloy with the composition

Si	Fe	Cu	Mn	Mg	Zn	Ti	Zr	Al
0.06	0.06	1.8	0.1	2.3	8.1	0.06	0.06	rest

Their own software took about 3.5 days to simulate the solidification and several subsequent homogenization steps.





After implementing OCASI the same simulation took 3.5 hours using 12 CPUs in parallel, a gain of time with a factor of 24.

Parallellization

Other tests with parallel calculations, also with OC itself, seems to indicate that there is practically no loss of time due to overhead and no memory leaks.

Parallellization

Other tests with parallel calculations, also with OC itself, seems to indicate that there is practically no loss of time due to overhead and no memory leaks.
The examples show the calculation of several equilibria in parallel. It is also possible to speed up the calculation of a single equilibrium by parallelizing parts of this, for example the grid minimizer and inverting the phase matrices.

Parallellization

Other tests with parallel calculations, also with OC itself, seems to indicate that there is practically no loss of time due to overhead and no memory leaks.
The examples show the calculation of several equilibria in parallel. It is also possible to speed up the calculation of a single equilibrium by parallelizing parts of this, for example the grid minimizer and inverting the phase matrices.
As far as I know OC is the only thermodynamic software which can offer equilibrium calculations in parallel using the OpenMP library as a standard feature.

OCASI library, how to use?

An application program must first initialize the data structures before calling any routine in OCASI.

call tqini(n,ceq)

where n provides some dimensioning and ceq is a pointer initialized to point to an equilibrium data structure. This must be passed to all other routines. With another subroutine call one can create additional such pointers each representing a equilibrium with an independent set of conditions.

OCASI library, how to use?

An application program must first initialize the data structures before calling any routine in OCASI.
call tqini(n,ceq)
where n provides some dimensioning and ceq is a pointer initialized to point to an equilibrium data structure. This must be passed to all other routines. With another subroutine call one can create additional such pointers each representing a equilibrium with an independent set of conditions.

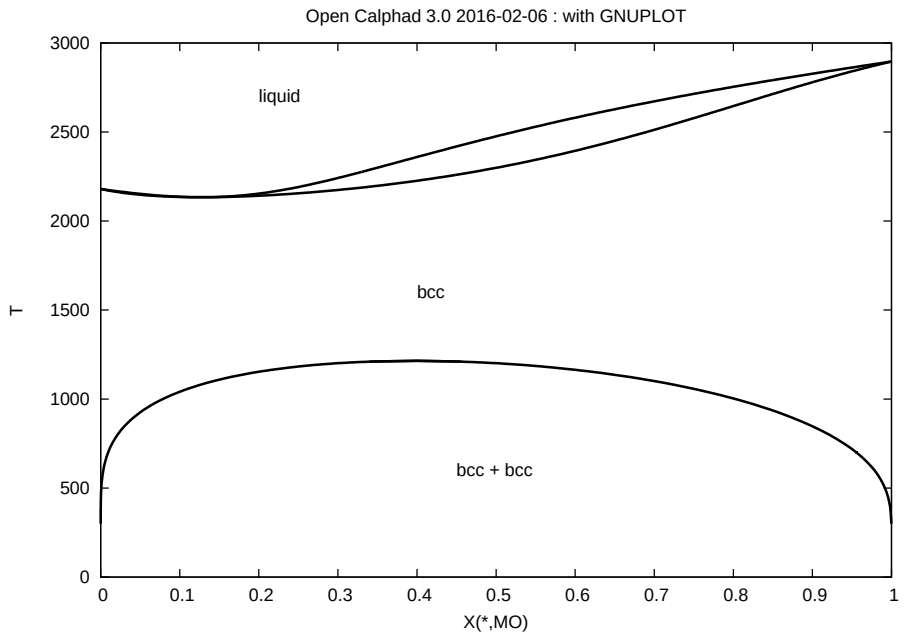
The second step is to read the thermodynamic data from a database
call tqrpfil(database,nel,elnames,ceq)
just passing the database name, the number and names of the elements to select and the equilibrium record pointer. The database information is stored in the static data structure.

Phases and phase tuples

A system has normally several phases and handling phases can be quite complicated, partly because there is no standard for phase names.

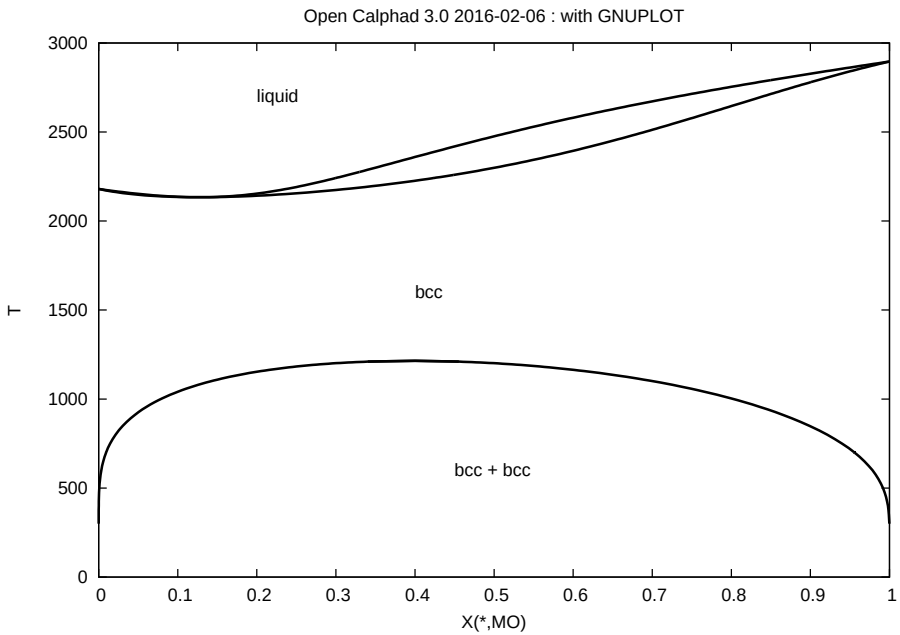
Phases and phase tuples

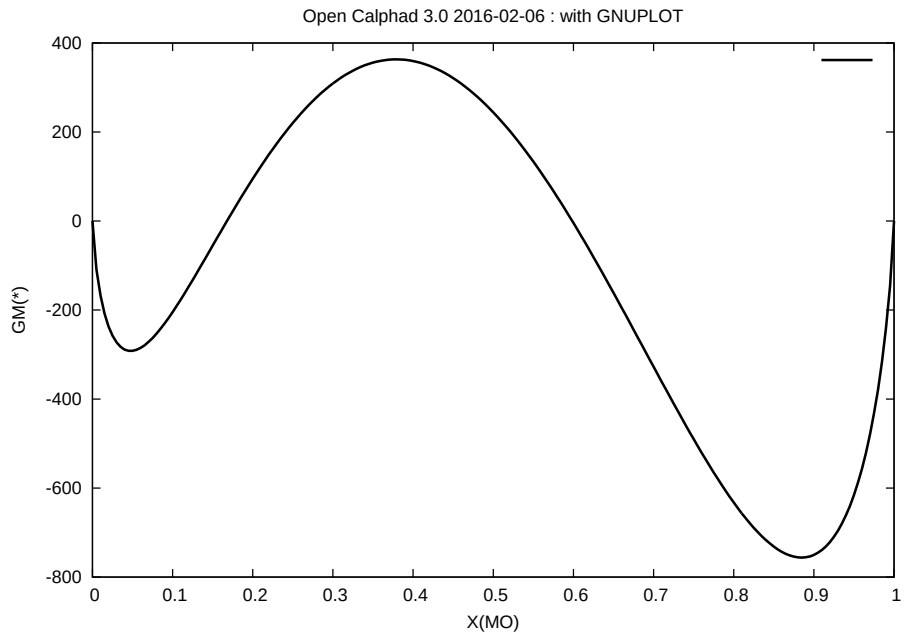
A system has normally several phases and handling phases can be quite complicated, partly because there is no standard for phase names. and because a phase can exist with two or more compositions at a miscibility gap like in Cr-Mo.



Phases and phase tuples

A system has normally several phases and handling phases can be quite complicated, partly because there is no standard for phase names. and because a phase can exist with two or more compositions at a miscibility gap like in Cr-Mo.

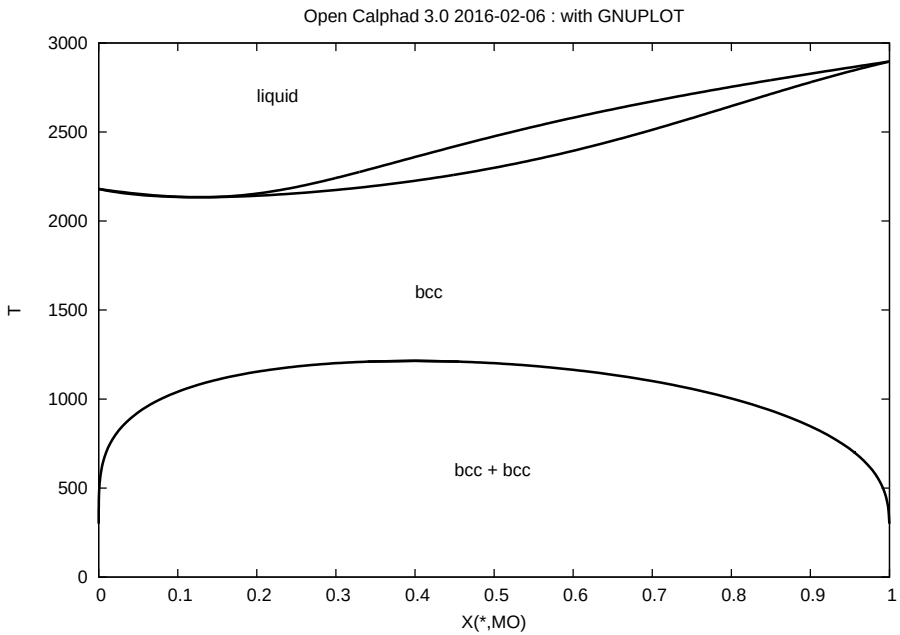


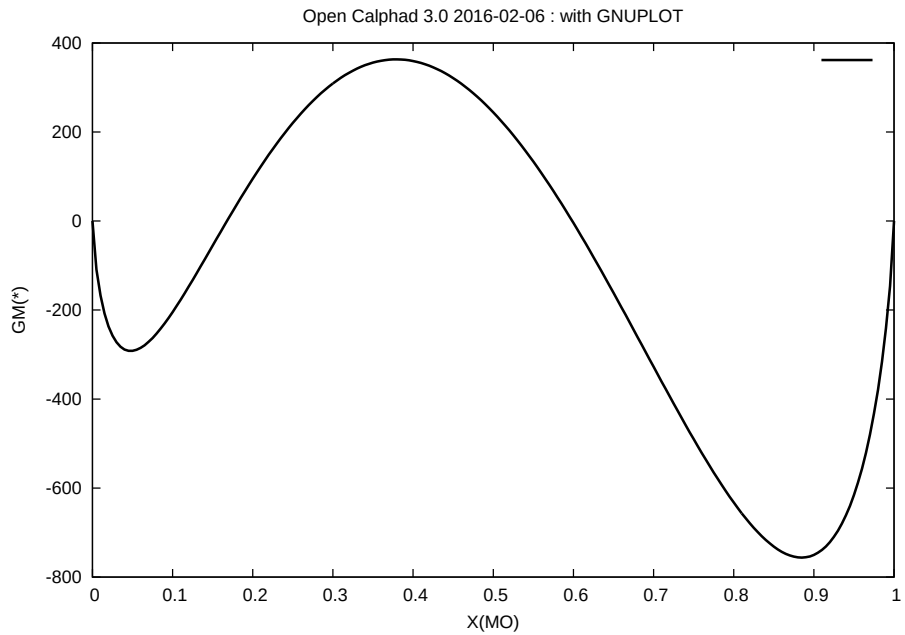


The right hand figure shows the Gibbs energy curve for bcc at 900 K with a convex part. As this is unstable the phase separates in two phases with the same structure but with different compositions.

Phases and phase tuples

A system has normally several phases and handling phases can be quite complicated, partly because there is no standard for phase names. and because a phase can exist with two or more compositions at a miscibility gap like in Cr-Mo.





The right hand figure shows the Gibbs energy curve for bcc at 900 K with a convex part. As this is unstable the phase separates in two phases with the same structure but with different compositions. In OCASI these are treated as phase tuples with two indices, one for the phase, the other for the composition set.

Phases and phase tuples

After reading the database there is one phase tuple for each phase. Composition sets (phase tuples) can be created explicity or automatically by the grid minimizer in OC when it discovers a stable miscibility gap at the equilibrium calculation.

Phases and phase tuples

After reading the database there is one phase tuple for each phase. Composition sets (phase tuples) can be created explicity or automatically by the grid minimizer in OC when it discovers a stable miscibility gap at the equilibrium calculation. This means the number of tuples may change after each calculation using the grid minimizer. To find the current number of tuples use

call tqgnp(nuberoftup,ceq)

Phases and phase tuples

After reading the database there is one phase tuple for each phase. Composition sets (phase tuples) can be created explicity or automatically by the grid minimizer in OC when it discovers a stable miscibility gap at the equilibrium calculation. This means the number of tuples may change after each calculation using the grid minimizer. To find the current number of tuples use

call tqgnp(nuberoftup,ceq)

The name of the phase is modified when there are several composition sets. The set number will be appended to the original phase name like "BCC#2" for the second set. The name of phase tuple "ptup", including composition set number is returned by

call tqgpn(ptup,phasename,ceq)

Phases and phase tuples

After reading the database there is one phase tuple for each phase. Composition sets (phase tuples) can be created explicity or automatically by the grid minimizer in OC when it discovers a stable miscibility gap at the equilibrium calculation. This means the number of tuples may change after each calculation using the grid minimizer. To find the current number of tuples use

call tqgnp(nuberoftup,ceq)

The name of the phase is modified when there are several composition sets. The set number will be appended to the original phase name like "BCC#2" for the second set. The name of phase tuple "ptup", including composition set number is returned by

call tqgpn(ptup,phasename,ceq)

Each phase tuple has a complete set of dynamic data in the equilibrium record but all phase tuples for the same phase has the same thermodynamic description, the static memory does not change.

OCASI library, state variable symbols

State variables are important as they are used to set conditions and extract results. All state variables are identified by their symbols and most state variables have their "normal" symbols like T, P, V, N and X.

- N(component index) is the total number of moles of a component,
- X(component index) is the overall mole fractions of a component,
- X(phase index,component index) is the mole fractions of a phase,

OCASI library, state variable symbols

State variables are important as they are used to set conditions and extract results. All state variables are identified by their symbols and most state variables have their "normal" symbols like T, P, V, N and X.

- N(component index) is the total number of moles of a component,
- X(component index) is the overall mole fractions of a component,
- X(phase index,component index) is the mole fractions of a phase,
- W%(component index) is the mass per cent of a component.

OCASI library, state variable symbols

State variables are important as they are used to set conditions and extract results. All state variables are identified by their symbols and most state variables have their "normal" symbols like T, P, V, N and X.

- N(component index) is the total number of moles of a component,
- X(component index) is the overall mole fractions of a component,
- X(phase index,component index) is the mole fractions of a phase,
- W%(component index) is the mass per cent of a component.
- MU(component index) is the chemical potential,

OCASI library, state variable symbols

State variables are important as they are used to set conditions and extract results. All state variables are identified by their symbols and most state variables have their "normal" symbols like T, P, V, N and X.

- N(component index) is the total number of moles of a component,
- X(component index) is the overall mole fractions of a component,
- X(phase index,component index) is the mole fractions of a phase,
- W%(component index) is the mass per cent of a component.
- MU(component index) is the chemical potential,
- DGM(phase index) is the driving force of a phase.

OCASI library, state variable symbols

State variables are important as they are used to set conditions and extract results. All state variables are identified by their symbols and most state variables have their "normal" symbols like T, P, V, N and X.

- N(component index) is the total number of moles of a component,
- X(component index) is the overall mole fractions of a component,
- X(phase index,component index) is the mole fractions of a phase,
- W%(component index) is the mass per cent of a component.
- MU(component index) is the chemical potential,
- DGM(phase index) is the driving force of a phase.
- NP(phase index) is the amount of a phase in moles of components,

OCASI library, state variable symbols

State variables are important as they are used to set conditions and extract results. All state variables are identified by their symbols and most state variables have their "normal" symbols like T, P, V, N and X.

- N(component index) is the total number of moles of a component,
- X(component index) is the overall mole fractions of a component,
- X(phase index,component index) is the mole fractions of a phase,
- W%(component index) is the mass per cent of a component.
- MU(component index) is the chemical potential,
- DGM(phase index) is the driving force of a phase.
- NP(phase index) is the amount of a phase in moles of components,
- N is the total amount of moles of in the system.

OCASI library, state variable symbols

State variables are important as they are used to set conditions and extract results. All state variables are identified by their symbols and most state variables have their "normal" symbols like T, P, V, N and X.

- N(component index) is the total number of moles of a component,
- X(component index) is the overall mole fractions of a component,
- X(phase index,component index) is the mole fractions of a phase,
- W%(component index) is the mass per cent of a component.
- MU(component index) is the chemical potential,
- DGM(phase index) is the driving force of a phase.
- NP(phase index) is the amount of a phase in moles of components,
- N is the total amount of moles of in the system.

These are just a few of the available state variables in OC.

OCASI library, setting conditions

To set a condition this subroutine is used.
call tqsetc(stv,n1,n2,value,cnum,ceq)
stv is a text with the state variable symbol like T or X or MU.
n1 and n2 are zero if not needed (like for T) but can be used for phase or component indices. Value is the value the state variable should have.
cnum is returned as an index of the condition.

Examples:
call tqsetc('T ', 0, 0, 1000, cnum, ceq)
call tqsetc('X ', 1, 0, 0.1, cnum, ceq)

According to Gibbs phase rule one must set as many condition as one has components plus 2 (for the potentials T and P).

OCASI library, setting conditions

To set a condition this subroutine is used.
call tqsetc(stv,n1,n2,value,cnum,ceq)
stv is a text with the state variable symbol like T or X or MU.
n1 and n2 are zero if not needed (like for T) but can be used for phase or component indices. Value is the value the state variable should have.
cnum is returned as an index of the condition.

Examples:
call tqsetc('T ', 0, 0, 1000, cnum, ceq)
call tqsetc('X ', 1, 0, 0.1, cnum, ceq)

According to Gibbs phase rule one must set as many condition as one has components plus 2 (for the potentials T and P). The conditions can be of many different kinds (not just the overall composition) for example specifying which phases that should be stable or the mole fraction of a component in a specific phase.

OCASI library, calculating equilibrium

In order to calculate the equilibrium the degrees of freedoms must be zero, i.e. the number of conditions must be equal to the number of components plus 2. The equilibrium is calculated with this call:
call tqce(target, n1, n2, value, ceq)

The **target** is an optional state variable name (with indices n1 and n2) and **value** is the desired value of this.
If you specify a target you should have one degree of freedom in the system and the software will try to adjust the system to reach the target value. This feature is mainly interesting when simulating a chemical process.

OCASI library, calculating equilibrium

In order to calculate the equilibrium the degrees of freedoms must be zero, i.e. the number of conditions must be equal to the number of components plus 2. The equilibrium is calculated with this call:
call tqce(target, n1, n2, value, ceq)

The **target** is an optional state variable name (with indices n1 and n2) and **value** is the desired value of this.
If you specify a target you should have one degree of freedom in the system and the software will try to adjust the system to reach the target value. This feature is mainly interesting when simulating a chemical process.
If no target is specified a normal equilibrium calculation is made. There is an option to calculate the equilibrium with or without the grid minimizer.

OCASI library, extracting information

After a successful calculation you can extract values of all kinds of state variables with

call tqgetv(stv, n1, n2, n3, values, ceq)

stv, n1 and n2 has the same meaning as when setting conditions. It is possible to use -1 to indicate that you want all values of a state variable, for example all mole fractions.

OCASI library, extracting information

After a successful calculation you can extract values of all kinds of state variables with

call tqgetv(stv, n1, n2, n3, values, ceq)

stv, n1 and n2 has the same meaning as when setting conditions. It is possible to use -1 to indicate that you want all values of a state variable, for example all mole fractions.

n3 is the dimension of values, set on return to number of values.

OCASI library, extracting information

After a successful calculation you can extract values of all kinds of state variables with
call tqgetv(stv, n1, n2, n3, values, ceq)
stv, n1 and n2 has the same meaning as when setting conditions. It is possible to use -1 to indicate that you want all values of a state variable, for example all mole fractions.
n3 is the dimension of values, set on return to number of values.

The amount of all phases:
call tqgetv('NP ', -1, 0, n3, values, ceq)

OCASI library, extracting information

After a successful calculation you can extract values of all kinds of state variables with
call tqgetv(stv, n1, n2, n3, values, ceq)
stv, n1 and n2 has the same meaning as when setting conditions. It is possible to use -1 to indicate that you want all values of a state variable, for example all mole fractions.
n3 is the dimension of values, set on return to number of values.

The amount of all phases:
call tqgetv('NP ', -1, 0, n3, values, ceq)
All mole fractions of phase with index 1:
call tqgetv('X ', 1, -1, n3, values, ceq)

OCASI library, extracting information

After a successful calculation you can extract values of all kinds of state variables with
call tqgetv(stv, n1, n2, n3, values, ceq)
stv, n1 and n2 has the same meaning as when setting conditions. It is possible to use -1 to indicate that you want all values of a state variable, for example all mole fractions.
n3 is the dimension of values, set on return to number of values.

The amount of all phases:
call tqgetv('NP ', -1, 0, n3, values, ceq)
All mole fractions of phase with index 1:
call tqgetv('X ', 1, -1, n3, values, ceq)
Other modeled properties like the mobility of Fe in the BCC phase can be retrieved by the same subroutine:
call tqgetv('MQ&FE(BCC) ', 0, 0, n3, values, ceq)

Dynamic data: a phase data record in ceq

```
  TYPE gtp_phase_varres
! Data here are for a composition set of a phase in an equilibrium record, ceq
! phlink: is index of phase record (static data) for this phase_varres record
! phstate: indicate state: fix/stable/entered/unknown/dormant/suspended
! phtupx: phase tuple index
     integer phlink,phstate,phtupx
....
! yfr: the constituent fraction array
     double precision, dimension(:), allocatable :: yfr
...
! prefix and suffix are added to the name for composition sets 2 and higher
     character*4 prefix,suffix
...
! arrays for storing calculated results for each phase (composition set)
! amfu: is amount formula units of the composition set (calculated result)
! netcharge: is net charge of phase (must be zero for stable phases)
! dgm: driving force
     double precision amfu,netcharge,dgm
!   last index 1=G, 2=TC, 3=BMAG. Properties defined in the gtp_propid record
! gval(*,1): is G, G.T, G.P, G.T.T, G.T.P and G.P.P
! dgval(1,j,1): is first derivatives of G wrt fractions j
! dgval(2,j,1): is second derivatives of G wrt fractions j and T
! dgval(3,j,1): is second derivatives of G wrt fractions j and P
! d2gval(ixsym(i,j),1): is second derivatives of G wrt fractions i and j
     double precision, dimension(:,:), allocatable :: gval
     double precision, dimension(:,:,:), allocatable :: dgval
     double precision, dimension(:,:), allocatable :: d2gval
  END TYPE gtp_phase_varres
! an array of these records is allocated inside each equilibrium record, ceq
```

Dynamic data: iso-C binding and ceq

The fortran TYPE record can be summarized for C++ software by the ISO C interface as:

```
typedef struct {
int phlink, phstate, phtupx;
        ...
char prefix[4], suffix[4];
        ...
double *yfr;
        ...
        double amfu, netcharge, dgm;
double **gval;
double ***dgval;
double **d2gval;
} gtp_phase_varres;
```

Dynamic data: iso-C binding and ceq

The fortran TYPE record can be summarized for C++ software by the ISO C interface as:

```
typedef struct {
int phlink, phstate, phtupx;
        ...
char prefix[4], suffix[4];
        ...
double *yfr;
        ...
        double amfu, netcharge, dgm;
double **gval;
double ***dgval;
double **d2gval;
} gtp_phase_varres;
```

Using the items in this struct a C++ program can access variables inside the dynamic data structure in OCASI.

Dynamic data: iso-C binding and ceq

The fortran TYPE record can be summarized for C++ software by the ISO C interface as:

```
typedef struct {
int phlink, phstate, phtupx;
        ...
char prefix[4], suffix[4];
        ...
double *yfr;
        ...
        double amfu, netcharge, dgm;
double **gval;
double ***dgval;
double **d2gval;
} gtp_phase_varres;
```

Using the items in this struct a C++ program can access variables inside the dynamic data structure in OCASI.
The first and second derivatives of the Gibbs energy are calculated analytically by the model package and used in the minimization algorithm to handle conditions of different types, improve speed of convergence and stability.

OCASI library, using the Fortran data structure

Via the "ceq" pointer to the equilibrium record it is possible to access data directly inside the OC equilibrium record, without the use of subroutines and state variables.
For example Gibbs energy for a phase_varres record with index lokcs is
gm=ceq%phase_varres(lokcs)%gval(1,1)

OCASI library, using the Fortran data structure

Via the "ceq" pointer to the equilibrium record it is possible to access data directly inside the OC equilibrium record, without the use of subroutines and state variables.
For example Gibbs energy for a phase_varres record with index lokcs is
gm=ceq%phase_varres(lokcs)%gval(1,1)
Another example is the chemical potentials of component i:
mu(i)=ceq%complist(i)%chempot(i)

OCASI library, using the Fortran data structure

Via the "ceq" pointer to the equilibrium record it is possible to access data directly inside the OC equilibrium record, without the use of subroutines and state variables.
For example Gibbs energy for a phase_varres record with index lokcs is
gm=ceq%phase_varres(lokcs)%gval(1,1)
Another example is the chemical potentials of component i:
mu(i)=ceq%complist(i)%chempot(i)
Note that the partial Gibbs energy of a constituent is not the chemical potential.

OCASI library, using the Fortran data structure

Via the "ceq" pointer to the equilibrium record it is possible to access data directly inside the OC equilibrium record, without the use of subroutines and state variables.
For example Gibbs energy for a phase_varres record with index lokcs is
gm=ceq%phase_varres(lokcs)%gval(1,1)
Another example is the chemical potentials of component i:
mu(i)=ceq%complist(i)%chempot(i)
Note that the partial Gibbs energy of a constituent is not the chemical potential. To obtain the chemical potentials from partials one must calculate an expression like this for a stable phase:

$$\mu_i = \left(\frac{\partial G}{\partial N_i}\right)_{T,P,N_7 j\neq i} = G_M + \left(\frac{\partial G_M}{\partial x_i}\right)_{T,P,x_{j\neq i}} - \sum_j x_j \left(\frac{\partial G_M}{\partial x_j}\right)_{T,P,x_{k\neq j}}$$

OCASI library, using the Fortran data structure

Via the "ceq" pointer to the equilibrium record it is possible to access data directly inside the OC equilibrium record, without the use of subroutines and state variables.
For example Gibbs energy for a phase_varres record with index lokcs is
gm=ceq%phase_varres(lokcs)%gval(1,1)
Another example is the chemical potentials of component i:
mu(i)=ceq%complist(i)%chempot(i)
Note that the partial Gibbs energy of a constituent is not the chemical potential. To obtain the chemical potentials from partials one must calculate an expression like this for a stable phase:

$$\mu_i = \left(\frac{\partial G}{\partial N_i}\right)_{T,P,N_7 j\neq i} = G_M + \left(\frac{\partial G_M}{\partial x_i}\right)_{T,P,x_{j\neq i}} - \sum_j x_j \left(\frac{\partial G_M}{\partial x_j}\right)_{T,P,x_{k\neq j}}$$

The application software must never change a value inside the OC data structure. Setting conditions, constitutions etc. must always be made via calls as additional variables may need to be updated.

OCASI library, iso-C binding and ceq

Using OC for programs written in C++ is easy using the iso-C standard for transfer of data between C++ and Fortran. As the passing of arguments is different in C++ and Fortran one must have a separate set of routines, for example

- **c_tqini(n,c_ceq) bind(c, name=c_tqini)** to initiate OCASI
- **c_tqsetc(statvar, ... , c_ceq) ...** to set a condition
- **c_tqce(mtarget, ... , c_ceq) ...** to calculate a equilibrium
- **c_tqgetv(statvar, ... , c_ceq) ...** to get a state variable value

Both the application examples shown were written in C++.

OCASI library, iso-C binding and ceq

Using OC for programs written in C++ is easy using the iso-C standard for transfer of data between C++ and Fortran. As the passing of arguments is different in C++ and Fortran one must have a separate set of routines, for example

- **c_tqini(n,c_ceq) bind(c, name=c_tqini)** to initiate OCASI
- **c_tqsetc(statvar, ... , c_ceq) ...** to set a condition
- **c_tqce(mtarget, ... , c_ceq) ...** to calculate a equilibrium
- **c_tqgetv(statvar, ... , c_ceq) ...** to get a state variable value

Both the application examples shown were written in C++.
Based on Fortran and C++ compatibility, it is also possible to interface OCASI functionalities in the Java language thanks to the Java Native Interface. Or other popular software languages.
But Fortran is still a useful language for number crunching.

Heat capacity, the dot derivative

The heat capacity is not available as state variable and to obtain it OC has implemented the "dot derivative" feature developed by Bo Jansson in the Thermo-Calc software. The symbol "H.T" is

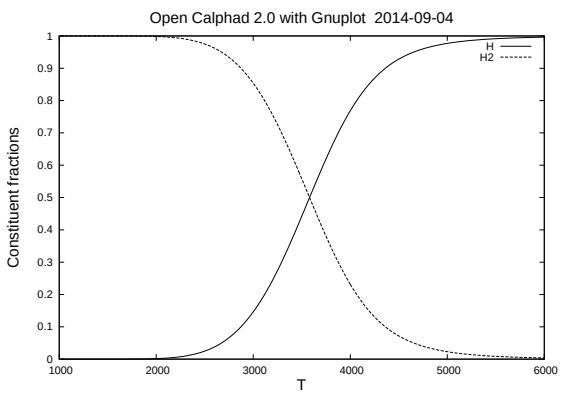
$$H.T = \left(\frac{\partial H}{\partial T}\right)_{P,N_A} = C_P$$

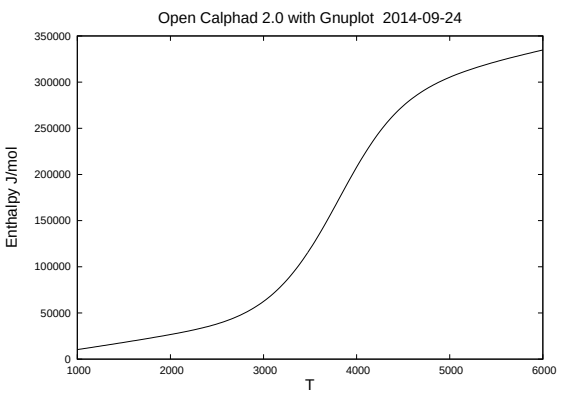
Heat capacity, the dot derivative

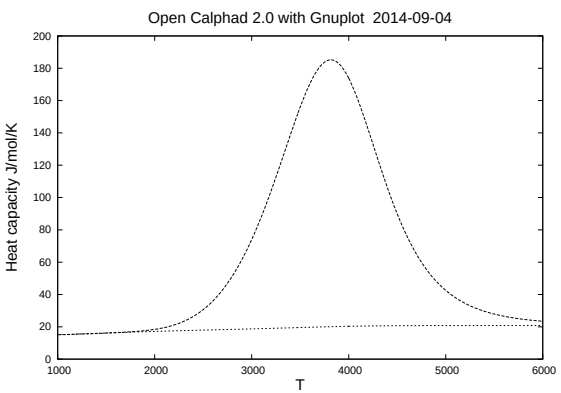
The heat capacity is not available as state variable and to obtain it OC has implemented the "dot derivative" feature developed by Bo Jansson in the Thermo-Calc software. The symbol "H.T" is

$$H.T = \left(\frac{\partial H}{\partial T}\right)_{P,N_A} = C_P$$

The heat capacity can be complicated to calculate when phases have internal degrees of freedoms as shown for a pure H gas with H_1 and H_2:





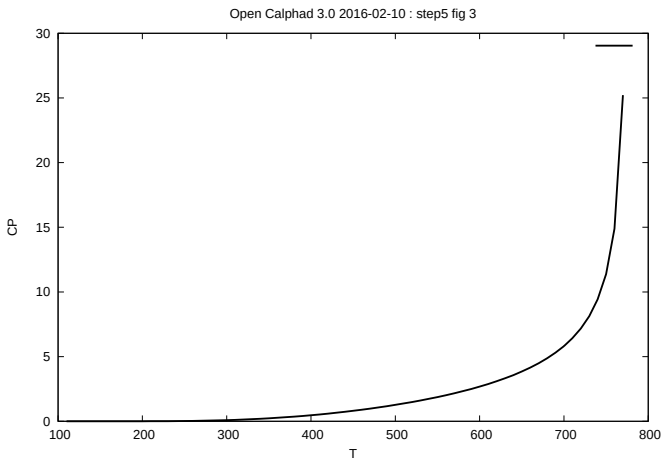


Heat capacity, the dot derivative

The heat capacity is not available as state variable and to obtain it OC has implemented the "dot derivative" feature developed by Bo Jansson in the Thermo-Calc software. The symbol "H.T" is

$$H.T = \left(\frac{\partial H}{\partial T}\right)_{P,N_A} = C_P$$

Order/disorder transitions and variable fractions of defects in solid phases have an influence on the heat capacity also. This is the heat capacity curve for the $L1_2$/A1 transition for $FeNI_3$



Heat capacity, the dot derivative

The heat capacity is not available as state variable and to obtain it OC has implemented the "dot derivative" feature developed by Bo Jansson in the Thermo-Calc software. The symbol "H.T" is

$$H.T = \left(\frac{\partial H}{\partial T}\right)_{P,N_A} = C_P$$

The "dot derivative" in OC (and TC) is calculated using the second derivatives of G from the last equilibrium calculation, there is no need of any numerical derivation.
The second derivatives are also used to calculate the stability function and needed to transform mobility data to diffusion coefficients.

Other use of dot derivatives

Dot derivatives can also be interesting to calculate properties important for simulations like the slope of the liquidus surface

$$X(\mathrm{Liquid}, \mathrm{Cr}).T = \frac{\partial X_{\mathrm{Cr}}^{\mathrm{Liquid}}}{\partial T}$$

gives the change of the Cr content in the liquid with T (if there are also other phases stable).

Other use of dot derivatives

Dot derivatives can also be interesting to calculate properties important for simulations like the slope of the liquidus surface

$$X(\mathrm{Liquid}, \mathrm{Cr}).T = \frac{\partial X_{\mathrm{Cr}}^{\mathrm{Liquid}}}{\partial T}$$

gives the change of the Cr content in the liquid with T (if there are also other phases stable).
Without this facility one would have to calculate the slope by taking the difference between two equilibrium calculations with a small temperature difference.

Summary

- The OC software is free with a GNU GPL license and the source code, documentation and many examples are available at http://www.opencalphad.org or the opencalphad repository at http://www.github.com

Summary

- The OC software is free with a GNU GPL license and the source code, documentation and many examples are available at http://www.opencalphad.org or the opencalphad repository at http://www.github.com
- The code is written in the new Fortran (2008) standard. A Fortran compiler like GNU Fortran 4.8 or later is needed. It has been tested on Windows, Linux and MacOS.

Summary

- The OC software is free with a GNU GPL license and the source code, documentation and many examples are available at http://www.opencalphad.org or the opencalphad repository at http://www.github.com
- The code is written in the new Fortran (2008) standard. A Fortran compiler like GNU Fortran 4.8 or later is needed. It has been tested on Windows, Linux and MacOS.
- The user interface is a command interface with macro facilities.

Summary

- The OC software is free with a GNU GPL license and the source code, documentation and many examples are available at http://www.opencalphad.org or the opencalphad repository at http://www.github.com
- The code is written in the new Fortran (2008) standard. A Fortran compiler like GNU Fortran 4.8 or later is needed. It has been tested on Windows, Linux and MacOS.
- The user interface is a command interface with macro facilities.
- The data structures and the algorithms are published and extensively documented.

Summary

- The OC software is free with a GNU GPL license and the source code, documentation and many examples are available at http://www.opencalphad.org or the opencalphad repository at http://www.github.com
- The code is written in the new Fortran (2008) standard. A Fortran compiler like GNU Fortran 4.8 or later is needed. It has been tested on Windows, Linux and MacOS.
- The user interface is a command interface with macro facilities.
- The data structures and the algorithms are published and extensively documented.
- An preliminary OCASI software interface has been tested and some examples are provided to show how to use it.

Summary

- The OC software is free with a GNU GPL license and the source code, documentation and many examples are available at http://www.opencalphad.org or the opencalphad repository at http://www.github.com
- The code is written in the new Fortran (2008) standard. A Fortran compiler like GNU Fortran 4.8 or later is needed. It has been tested on Windows, Linux and MacOS.
- The user interface is a command interface with macro facilities.
- The data structures and the algorithms are published and extensively documented.
- An preliminary OCASI software interface has been tested and some examples are provided to show how to use it.
- **OC is fully parallelized using the OpenMP library.**

Summary

- The OC software is free with a GNU GPL license and the source code, documentation and many examples are available at http://www.opencalphad.org or the opencalphad repository at http://www.github.com
- The code is written in the new Fortran (2008) standard. A Fortran compiler like GNU Fortran 4.8 or later is needed. It has been tested on Windows, Linux and MacOS.
- The user interface is a command interface with macro facilities.
- The data structures and the algorithms are published and extensively documented.
- An preliminary OCASI software interface has been tested and some examples are provided to show how to use it.
- **OC is fully parallelized using the OpenMP library.**
- Using OC inside a commercial software requires another kind of license.

Summary -

- OC has no high quality databases.

Summary -

- OC has no high quality databases.
- OC supports the development of free databases.
- and OC is trying to revise the SGTE unary database.

Summary -

- OC has no high quality databases.
- OC supports the development of free databases.
- and OC is trying to revise the SGTE unary database.
- There are still a number of unimplemented features.

Summary -

- OC has no high quality databases.
- OC supports the development of free databases.
- and OC is trying to revise the SGTE unary database.
- There are still a number of unimplemented features.
- Many models are on the waiting list.

Summary -

- OC has no high quality databases.
- OC supports the development of free databases.
- and OC is trying to revise the SGTE unary database.
- There are still a number of unimplemented features.
- Many models are on the waiting list.
- OC has no organization providing support.

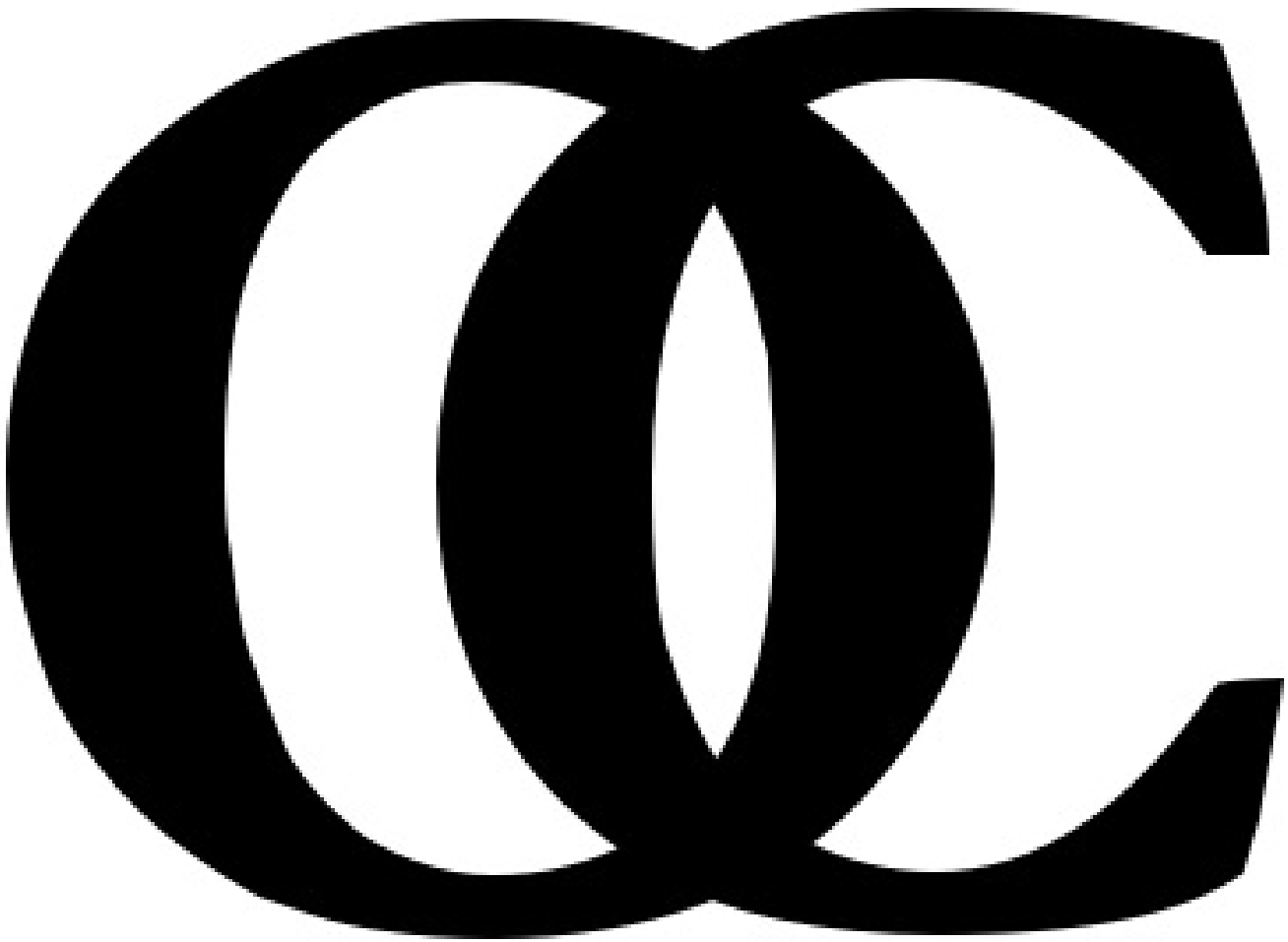
Summary -

- OC has no high quality databases.
- OC supports the development of free databases.
- and OC is trying to revise the SGTE unary database.
- There are still a number of unimplemented features.
- Many models are on the waiting list.
- OC has no organization providing support.

But you can fix things yourself!

Welcome to develop and use:

websites: http://www.opencalphad.org or the opencalphad repository at http://www.github.com



Welcome to develop and use:

websites: http://www.opencalphad.org or the opencalphad repository at http://www.github.com

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

Welcome to develop and use:

websites: http://www.opencalphad.org or the opencalphad repository at http://www.github.com

- Sundman B, Kattner U R, Palumbo M and Fries S G, **OpenCalphad - a free thermodynamic software**, Integrating Materials and Manufacturing Innovation, **4**:1 (2015)
- Sundman B, Lu X-G and Ohtani H, Comp Mat Sci, **101** (2015) 127-137

Welcome to develop and use:

websites: http://www.opencalphad.org or the opencalphad repository at http://www.github.com

- Sundman B, Kattner U R, Palumbo M and Fries S G, **OpenCalphad - a free thermodynamic software**, Integrating Materials and Manufacturing Innovation, **4**:1 (2015)
- Sundman B, Lu X-G and Ohtani H, Comp Mat Sci, **101** (2015) 127-137
- Eriksson G, Spencer P and Sippola H, 2nd Colloquium on Process Simulations, pp 115-126, June 1995, HUT, Espoo, Finland.

Welcome to develop and use:

websites: http://www.opencalphad.org or the opencalphad repository at http://www.github.com

- Sundman B, Kattner U R, Palumbo M and Fries S G, **OpenCalphad - a free thermodynamic software**, Integrating Materials and Manufacturing Innovation, **4**:1 (2015)
- Sundman B, Lu X-G and Ohtani H, Comp Mat Sci, **101** (2015) 127-137
- Eriksson G, Spencer P and Sippola H, 2nd Colloquium on Process Simulations, pp 115-126, June 1995, HUT, Espoo, Finland.
- Zhang L, Stratmann M, Du Y, Sundman B and Steinbach I, Acta Materialia **88** (2015) 156-169

Welcome to develop and use:

websites: http://www.opencalphad.org or the opencalphad repository at http://www.github.com

- Sundman B, Kattner U R, Palumbo M and Fries S G, **OpenCalphad - a free thermodynamic software**, Integrating Materials and Manufacturing Innovation, **4**:1 (2015)
- Sundman B, Lu X-G and Ohtani H, Comp Mat Sci, **101** (2015) 127-137
- Eriksson G, Spencer P and Sippola H, 2nd Colloquium on Process Simulations, pp 115-126, June 1995, HUT, Espoo, Finland.
- Zhang L, Stratmann M, Du Y, Sundman B and Steinbach I, Acta Materialia **88** (2015) 156-169
- Sundman B, Stratmann M, Sigli C, Le Tellier R, Kattner U R, Palumbo M, Fries S G, submitted to Comp Phys Comm. (2016)

Thanks for listening

Time left?

Algorithm for equilibrium calculation

maybe??