Available online at www.sciencedirect.com
ScienceDirect
Computer Coupling of Phase Diagrams and Thermochemistry 32 (2008) 129–134
www.elsevier.com/locate/calphad

A computational interface for thermodynamic calculations software MTDATA

Zhiheng Huang[a,*], Paul P. Conway[a], Rachel C. Thomson[b], Alan T. Dinsdale[c], Jim A.J. Robinson[c]

[a] *Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK*
[b] *Institute of Polymer Technology and Materials Engineering, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK*
[c] *NPL Materials Centre, National Physical Laboratory, Teddington, Middlesex, TW11 0LW, UK*

Received 4 May 2007; received in revised form 21 June 2007; accepted 7 July 2007
Available online 8 August 2007

Abstract

A novel computational interface between a thermodynamic calculations software MTDATA and a scientific computing software MATLAB is developed utilising the MATLAB interface to generic dynamic link libraries (DLLs). As such, runtine thermodynamic calculations can be performed in MATLAB environment with extended flexibility. In addition, combined thermodynamic-kinetic models developed in MATLAB can have a seamless link to MTDATA whenever thermodynamic parameters are required. Further extension of the interface to a multiphysics modelling software, COMSOL Multiphysics, is also introduced. The interface extends the capabilities of these three software packages and provides a methodology for sophisticated and even three dimensional combined thermodynamic-kinetic modelling. Applications of the interface in equilibrium thermodynamic calculation, nonequilibrium Scheil calculation, and diffusion in a multicomponent multiphase system are presented.

Keywords: Computational thermodynamics; Kinetic modelling; MTDATA; MATLAB; COMSOL Multiphysics

1. Introduction

A phase diagram is the most concise representation of the phase equilibria present in a system [1]. The development and maturity of computational thermodynamics provides phase diagram calculations with ease and makes them accessible to nonexpert users. A lot of useful information for a system can be derived from a thermodynamic calculation with an appropriate thermodynamic database. For example, the stable phases, their compositions, and the amount of the phases at any given temperature. However, the thermodynamic information alone is not sufficient in many different kinds of materials processing simulations, such as phase transformation, solidification, interfacial reactions and microstructure evolution. In such processes, the kinetics of the system, e.g. the microstructure as a function of time, is also of substantial interest. Kinetic models are developed to simulate the processes involving kinetics and, in general, thermodynamic parameters are incorporated in such models. Therefore, being able to freely access the functions of thermodynamic calculation software in a kinetic model is of practical importance. Thermodynamic calculation software packages such as MTDATA [2] and Thermo-Calc [3] provide application programming interfaces (API) which allow users to write their own programs and access to the built-in functions of the software. With this flexibility, the thermodynamic calculations can be done outside the thermodynamic calculation software whenever necessary. In particular, combining the advantages of a third-party software or environment, allows the capabilities of the original thermodynamic calculation software to be extended. Tanaka et al. [4], for example, used static linking to ChemApp to calculate the surface tension of Sn–Bi alloys. Strandlund et al. [5] developed an interface between Thermo-Calc and MATLAB [6] by the MEX (MATLAB Executable) file mechanism, in which mixed computer language programming is involved. In recent literature, there is increasing interest in incorporating thermodynamic calculations into phase field microstructure modelling techniques (e.g. [7]).

* Corresponding author. Tel.: +44 1509 227678; fax: +44 1509 227671.
E-mail addresses: z.huang@lboro.ac.uk (Z. Huang), p.p.conway@lboro.ac.uk (P.P. Conway), r.c.thomson@lboro.ac.uk (R.C. Thomson), alan.dinsdale@npl.co.uk (A.T. Dinsdale), jim.robinson@npl.co.uk (J.A.J. Robinson).

0364-5916/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.calphad.2007.07.003

This paper presents the development of a computational interface for MTDATA. MATLAB is selected as the third party environment because of its advantages in scientific computing. In addition, MATLAB has a seamless interface with COMSOL Multiphysics [8], which is a multiphysics modelling software based on the finite element method (FEM). As such, an interface between MTDATA and MATLAB has the potential to work with COMSOL Multiphysics. This paper demonstrates such an interface to enable a linkage between these three software packages.

2. Thermodynamic calculation software—MTDATA

The MTDATA software, developed and maintained by the National Physical Laboratory (NPL), provides a facility for retrieving thermodynamic data from databases, compiling data sets for the elements of interest, and assessing these data; calculating binary and ternary equilibrium phase diagrams, as well as equilibria in higher order multicomponent, multiphase systems; and also a facility for linking the minimization algorithms to external software or user code enabling, amongst others, the investigation of nonequilibrium processes. A number of modules are incorporated for manipulating and retrieving the data, for making various types of calculation, and plotting binary, ternary, multicomponent, and predominance area diagrams.

3. Scientific computing tools: MATLAB and COMSOL Multiphysics

MATLAB is a high-performance language for technical computing [6]. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include mathematics and computation; algorithm development; data acquisition; modelling, simulation, and prototyping; data analysis, exploration and visualization; scientific and engineering graphics; and application development, including graphical user interface building [6]. The MATLAB language is particularly well suited to designing predictive mathematical models and developing application-specific algorithms [9]. It provides easy access to a range of both elementary and advanced algorithms for numerical computing. These algorithms include matrix manipulation, operations for linear algebra, differential or partial differental equation solving, statistics, data analysis. MATLAB Toolboxes are add-ons that extend the capabilities of MATLAB with specialized functions. The most important advantage of using MATLAB is its capabilities in matrix and vector operations.

With all those features and advantages, MATLAB is a good environemt for the development of combined thermodynamic kinetic models. For some kinetic modelling in materials processing, partial differential equations (PDEs) are commonly encountered. Although MATLAB has a PDETOOL toolbox that can deal with some simpler PDEs based on the finite element method (FEM), there are some difficulties in solving complex PDEs in MATLAB, e.g. access to the boundary conditions at internal domain borders, periodic boundary conditions, and three-dimensional (3D) modelling. COMSOL Multiphysics is a powerful interactive environment for multiphysics modelling and solving all kind of scientific and engineering problems based on PDEs [8]. COMSOL has been developed from the PDETOOL toolbox in MATLAB, but it has now become an independent software package and offers substantial capabilities that the PDETOOL does not possess.

4. MATLAB interface to generic DLLs

A shared library contains a set of object files that implement functions and subroutines that can be linked with other object files to produce a complete executable program. In Windows, as in other modern operating systems, libraries may be either static or dynamic. The referenced object files from a static library are linked into the executable when it is built, but the routines in a dynamic library are not loaded until runtime. In Windows, static library files have the subscript ".lib", whereas dynamic libraries use the subscript ".dll". Executables that reference dynamic libraries are typically smaller than those that reference static libraries because the object files from the dynamic library are not present, but are loaded into memory at runtime directly from the DLL file.

The MATLAB Interface to Generic DLLs enables interaction with functions in dynamic link libraries directly from MATLAB [10]. This is the basis on which the novel interface between MTDATA and MATLAB in this paper is built. Using MATLAB 6.5.1 (R13SP1) and later versions, it is possible to access functions defined in Windows standard dynamic link libraries (.dll) through the MATLAB command line using the LOADLIBRARY function. This feature is available for Windows platforms only for releases R13SP1 through R14SP1, however, from release R14SP2, LOADLIBRARY is supported on both Windows and Linux.

5. A novel interface between MTDATA and MATLAB

Using the MATLAB interface to generic DLLs, it is possible to build an interface between MTDATA and MATLAB using the MTDATA DLL. To use the MATLAB LOADLIBRARY function, a header file which contains the declarations of all the functions and subroutines included in the MTDATA DLL needs to be constructed first. It is worth noting that the DLL for MTDATA is programmed in Fortran and there are a lot of differences in data types in Fortran and MATLAB languages. Therefore, the header file needs to be carefully constructed with reference to the MTDATA Application Interface Programming Guide [11], considering the differences of data types in the Fortran and MATLAB languages. Table 1 presents several examples of the differences in function declarations between the Fortran and MATLAB languages. A list of the header files containing part of the functions and subroutines in the MTDATA DLL can be found in Appendix.

The MTDATA DLL, i.e. "mtdata.dll", can be loaded into memory and made accessible to all applications by issuing the

Table 1
Declaration of subroutines and functions in Fortan and MATLAB

Fortran declaration	MATLAB declaration
DOUBLE PRECISION FUNCTION GAS_CONSTANT()	double GAS_CONSTANT()
DOUBLE PRECISION FUNCTION MOLES_OF_COMPONENTS_IN_PHASE(NPHA,IFLAG) INTEGER NPHA,IFLAG	double MOLES_OF_COMPONENT_IN_PHASE(int *,int *)
INTEGER FUNCTION ACT_PHASE_NO(PNAM) CHARACTER PNAM	int ACT_PHASE_NO(char *,int)
INTEGER FUNCTION EQUIL_PHASE_NO(PNAM) CHARACTER PNAM	int EQUIL_PHASE_NO(char *,int)
CHARACTER*(*) FUNCTION ACT_COMPONENT_NAME(NCOM) INTEGER NCOM	void ACT_COMPONENT_NAME(char *,int,int *)
SUBROUNTINE OPEN_MPI_FILE(CHINP,CHOUT,NERR) CHARACTER CHINP,CHOUT INTEGER NERR	void OPEN_MPI_FILE(char *,int,char *,int,int *)

following command in MATLAB after the header file has been properly constructed:

loadlibrary mtdata.dll mtdata.h alias mt

where "mtdata.h" is the header file for "mtdata.dll". It is noted that the library has been assigned an alias, i.e. "mt", which must be used in all further references to the library. The functions and subroutines included in the library can then be viewed after successful loading of the DLL by the following command:

libfunctionsview('mt')

where 'mt' is the alias of the DLL "mtdata.dll".

The procedures described up to this point are the preparation work, after which all of the functions and subroutines in the "mtdata.dll" DLL are accessible directly in the MATLAB environment. The "calllib" function should be used to call any of the functions from the library, for example, the following line calls the 'GAS_CONSTANT' function in the DLL.

calllib('mt','GAS_CONSTANT')

Compared with a static linking, the link between MTDATA and MATLAB through DLL results in several advantages, e.g. short program length, efficient usage of computer memory and an increased linking speed.

It is important to note the precision of the inputs in MATLAB when using the MTDATA–MATLAB interface. A detailed example is provided at the companion website of this paper [12].

6. An interface between MTDATA and COMSOL Multiphysics

Since its introduction, COMSOL Multiphysics [13] (hereinafter referred to as "COMSOL") has been fully integrated with MATLAB. In addition, any COMSOL model (".mph" file) can also be saved as a MATLAB m file (".m" file) and subsequently can run in the MATLAB environment once the link between COMSOL and MATLAB is enabled during installation. Furthermore, the COMSOL Script (COMSOL programming language) is compatible with MATLAB. Therefore, the novel interface established between MTDATA and MATLAB can be applied to COMSOL. However, it should be noted that the MTDATA DLL should only be loaded once in any COMSOL model. For example, if the DLL has already been loaded into the memory in the main program of a COMSOL model, the functions and subroutines in the DLL can be called directly in any of the functions of that COMSOL model and any attempt to load the DLL again in the functions and subroutines will result in linking errors. Fig. 1 shows a schematic diagram of the linking of the three software packages MTDATA, MATLAB and COMSOL.

7. Applications

This section provides several examples of using the interface to conduct equilibrium thermodynamic calculation, nonequlibrium Scheil calculation and diffusion analysis in a multicomponent multiphase system. The MATLAB codes for those three examples can be found at the companion webiste of this paper.

7.1. Equilibrium thermodynamic calculation

This is the first example to demonstrate how to use the interface between MTDATA and MATLAB[1] to conduct equlibrium thermodynamic calculations in MATLAB environment. The complete source code for this example, i.e. "equilibrium.m", can be found at the companion website of this paper. The major steps of this implementation are listed as below:

(1) Define or initialize the parameters that are required in the program;

[1] MTDATA version 4.73 and MATLAB 6.5 were used to implement the examples in this paper.

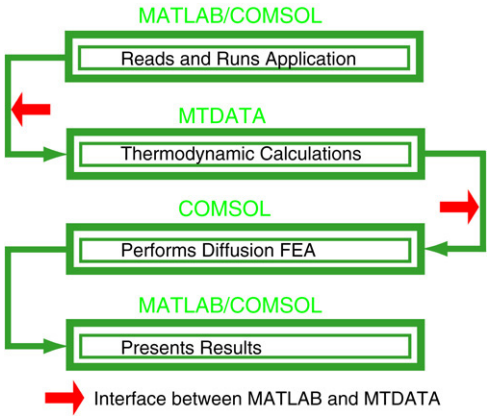

Fig. 1. A schematic diagram showing the mechanisms of the linking for the three software packages MTDATA, MATLAB and COMSOL.

(2) Input the thermodynamic datafile "*.MPI" for the thermodynamic calculation;
(3) Load the MTDATA DLL and initialize MTDATA;
(4) Open the MPI file and set the environment for MTDATA;
(5) Specify the conditions of the system, e.g. pressure, temperature, composition, and etc.;
(6) Conduct thermodynamic calculations at specified temperature(s);
(7) Assemble the results and display the output in MATLAB;
(8) Unload the MTDATA DLL.

7.2. Non-equilibrium Scheil calculation

Unlike the equilibrium calculations, Scheil calculations assume local equilibrium at the liquid/solid interface in which there is complete diffusion in the liquid and no diffusion at all in the solid [14], which simulates the worst case of micro segregation with the lowest possible final freezing temperature during cooling [15]. Fig. 2 illustrates how the Scheil model is implemented using the MTDATA–MATLAB interface. It is worth a mention that back diffusion is not considered in this program. The complete source code and a series of snapshots during the runtime of this example "scheil.m" is referred to the companion website of this paper. In addition to presenting the results in MATLAB, there are attempts to export the results from thermodynamic calculation to an Excel spreadsheet in "scheil.m", e.g. lines 373 to 394.

7.3. Diffusion in a multicomponent multiphase system

This example demonstrates the possibility to utilize the interface amongst the three software packages, i.e. MTDATA, MATLAB and COMSOL,[2] to solve the diffusion equations in a multicomponent and multiphase system using finite element method. The linking mechanisms amongst the three software

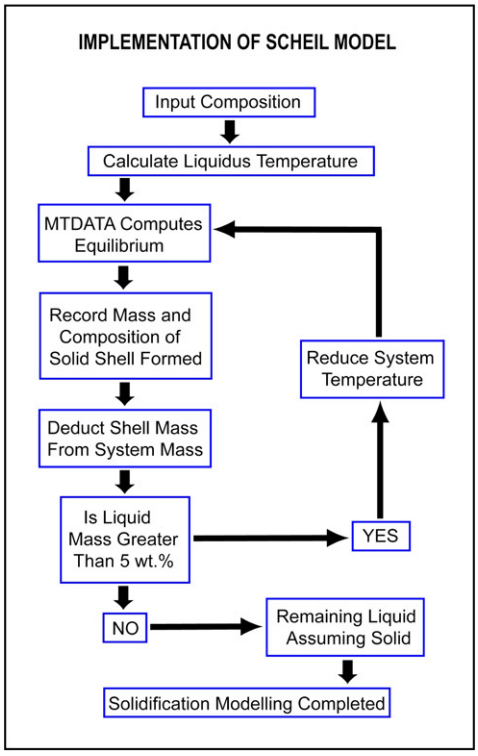

Fig. 2. Flow chart for implementing Scheil model in MATLAB.

packages is referred to in Fig. 1. The complete source code "test1.m" for this example can be found at the companion website of this paper.

In a multicomponent multiphase system, the flux J_i^V (through a surface of fixed coordinate in the volume-fixed frame of reference) of component i can be expressed as [16]:

$$J_i^V = \sum_k^N \left(\delta_{ik} - x_i \frac{V_k}{V_m}\right) c_k M_k \nabla \mu_k, \quad i = 1, 2, \ldots, N$$

and the diffusion equations in the system are [16]:

$$\frac{\partial x_i}{\partial t} = \nabla \sum_k^N \left(\delta_{ik} - x_i \frac{V_k}{V_m}\right) c_k M_k \nabla \mu_k, \quad i = 1, 2, \ldots, N$$

where N is the number of components in the system; δ_{ik} is the Kronecker Delta, i.e. $\delta = 0$ if $i \neq k$ and $\delta = 1$ if $i = k$; c_i, x_i are the concentration and mole fraction of component i ($c_i = x_i / V_m$) respectively; V_i, V_m are the molar volume of component i and diffusion matrix respectively; M_i is the mobility and μ_i are the chemical potential of component i.

This example studies the diffusion in a Cu–Sn binary system with four phases, i.e. liquid Sn phase, Cu_6Sn_5 phase, Cu_3Sn phase and the fcc-Cu phase. The finite element geometry and mesh are first generated in COMSOL. The diffusion equations are defined using the PDE general form of COMSOL. The chemical potential and molar volume are calculated through the interface between MTDATA and MATLAB. The diffusion equations are solved in COMSOL using FEM and the results are presented in MATLAB. Fig. 3 provides a snapshot of the FEM mesh construction after the thermodynamic calculation

[2] FEMLAB 3.0a was used in this paper. FEMLAB becomes COMSOL Multiphysics since Version 3.2.

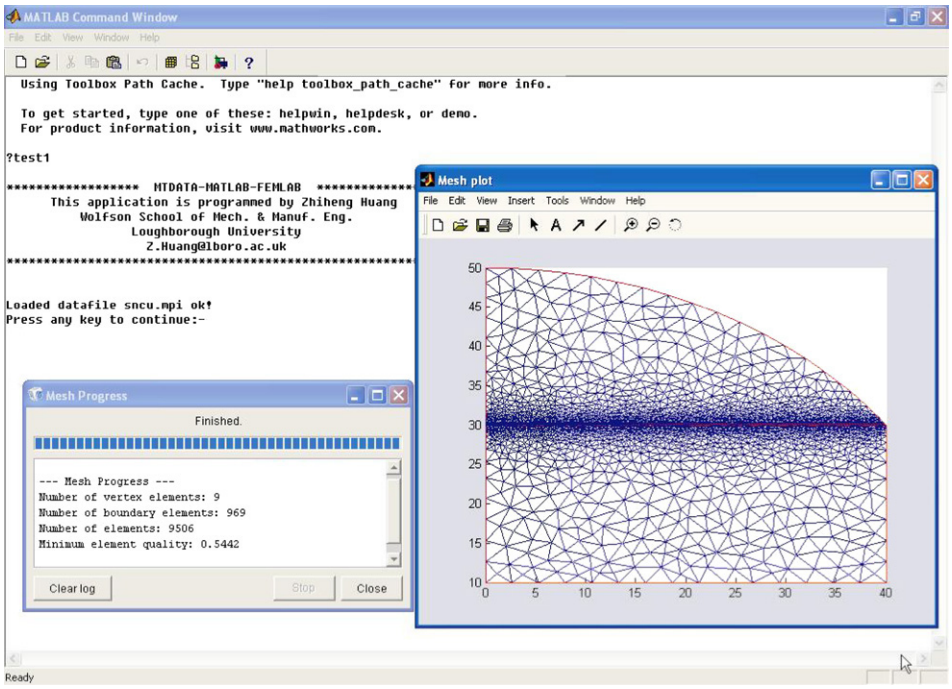

Fig. 3. A snapshot during the runtime of program 'test1.m': FEM mesh plot.

has been done. A series of snapshots of the windows during the runtime of the program can be found at the companion website of this paper. It should be noted the example presented here is for illustrative purpose only. A complete and fully functional model to study the kinetics of the intermetallic compounds (IMCs) formation in this system is currently ongoing. However, the source code of this example is well-suited to use as a template to solve similar problems.

8. Summary

This paper has described a novel computational interface between MTDATA and MATLAB. Further work that extends the interface to COMSOL has also been discussed. The capabilities of the thermodynamic calculation software MTDATA are greatly extended with such an interface to MATLAB and COMSOL. The routine calculations in MTDATA can now be conducted within MATLAB, COMSOL and combined MATLAB-COMSOL environments. Therefore, this interface has great potential to facilitate combined thermodynamic and kinetic modelling, e.g. direct calls to the thermodynamic software MTDATA are possible within both MATLAB and COMSOL during runtime if any thermodynamic parameters are needed in the kinetic models. The interface between MTDATA and MATLAB utilizing the DLL mechanism offers several advantages compared to static linking. In addition, the extension of the interface to COMSOL makes 3D combined thermodynamic-kinetic modelling possible.

Acknowledgements

One of the authors (Z. Huang) wish to acknowledge financial support from the UK's EPSRC-IMCRC at Loughborough University and the EPSRC 3D-Mintegration project.

Appendix

HEADER FILE FOR MTDATA DLL*

```
double ACT_COMPONENT_MOLES(int *);
double COMPONENT_FRACTION_IN_PHASE(int *,int *,int *);
double COMPONENT_MASS_IN_PHASE(int *,int *,int *);
double COMPONENT_W_OF_PHASE(int *,int *,int *);
double COMPONENT_X_OF_PHASE(int *,int *,int *);
double GAS_CONSTANT();
double GIBBS_ENERGY_OF_PHASE(int *);
double INIT_COMPONENT_MASS(int *);
double INIT_COMPONENT_MOLES(int *);
double MASS_IN_PHASE(int *,int *);
double MASS_OF_SYSTEM();
double MOLAR_MASS(int *);
double MOLE_FRACTION(char *,int);
double MOLES_OF_SPECIES_IN_PHASE(int *,int *);
double MOLES_OF_COMPONENTS_IN_PHASE(int *,int *);
double PARTIAL_SYSTEM_GIBBS_ENERGY(int *);
double PHASE_BOUNDARY_TEMPERATURE(int *);
double PRESSURE();
double SUM_OF_COMPONENT_MASS();
double SUM_OF_COMPONENT_AMOUNT();
double SYSTEM_ENTHALPY();
double SYSTEM_ENTROPY();
double SYSTEM_GIBBS_ENERGY();
double SYSTEM_VOLUME();
double TEMPERATURE();
double VOLUME_OF_PHASE(int *);
int ACT_NO_OF_COMPONENTS();
int ACT_NO_OF_PHASES();
int ACT_PHASE_NO(char *,int);
int ACT_PHASE_CLASS(int *);
int BOUNDARY_NO_OF_PHASES(int *);
int BOUNDARY_PHASE_NO(int *,int *);
int EQUIL_NO_OF_PHASES();
```

```
int EQUIL_PHASE_NO(char *,int);
int IMISC_ACT_PHASE(int *);
int INIT_COMPONENT_NO(char *,int);
int INIT_NO_OF_COMPONENTS();
int INIT_NO_OF_PHASES();
int INIT_PHASE_CLASS(int *);
int INIT_PHASE_NO(char *,int);
int NO_OF_PHASE_FIELDS();
int NORMAL_INIT_PHASE(int *);
int NUMBER_OF_ERRORS();
int PHASE_PRESENT_AT_EQUILIBRIUM(int *);
void ACT_COMPONENT_NAME(char *,int,int *);
void ACT_PHASE_NAME(char *,int,int *);
void COMPUTE_BOUNDARY_TEMPERATURE(double *,double *,…
double *,int *,int *,double *,double *,double *,int *,int *);
void COMPUTE_EQUILIBRIUM();
void EQUIL_PHASE_NAME(char *,int,int *);
void INIT_COMPONENT_NAME(char *,int,int *);
void INIT_PHASE_NAME(char *,int,int *);
void INITIALISE_MTDATA_DLL(int *);
void MTOPTN(int *,char *,int);
void OPEN_MPI_FILE(char *,int,char *,int,int *);
void SET_INIT_COMPONENT_AMOUNT(int *,double *);
void SET_INIT_COMPONENT_CLASS(int *,int *);
void SET_INIT_COMPONENT_MASS(int *,double *);
void SET_INIT_PHASE_CLASS(int *,int *);
void SET_INIT_PHASE_CLASS(int *,int *);
void SET_PRESSURE(double *);
void SET_TEMPERATURE(double *);
void WRITE_ERROR();
```

* This header file contains only part of the functions and subrountines in the MTDATA DLL.

References

[1] C.R. Kao, JOM 54 (2002) 44.
[2] R.H. Davies, A.T. Dinsdale, J.A. Gisby, J.A.J. Robinson, S.M. Martin, CALPHAD 26 (2002) 229–271.
[3] B. Sundman, B. Jansson, J. Andersson, CALPHAD 9 (1985) 153–190.
[4] T. Tanaka, K. Hack, S. Hara, CALPHAD 24 (2000) 465–474.
[5] H. Strandlund, Comput. Mater. Sci. 29 (2004) 187–194.
[6] MATLAB – The Language of Technical Computing – Getting Started with MATLAB Version 7, The Mathworks, Inc., MA, USA, 2005.
[7] H. Kobayashi, M. Ode, S.G. Kim, W.T. Kim, T. Suzuki, Scr. Mater. 48 (2003) 689–694.
[8] COMSOL Multiphysics User's Guide Version 3.2, COMSOL AB, Stockholm, Sweden, 2005.
[9] MATLAB – The Language of Technical Computing – Programming Version 7, The Mathworks, Inc, MA, USA, 2005.
[10] MATLAB – The Language of Technical Computing – Extenal Interfaces Version 7, The Mathworks, Inc., MA, USA, 2005.
[11] R.H. Davies, A.T. Dinsdale, J.A. Gisby, J.A.J. Robinson, MTDATA Application Interface Programming Guide, National Physical Laboratory, Teddington, Middlesex, UK, 2005.
[12] http://calphad07.googlepages.com.
[13] COMSOL Multiphysics — Scripting Guide Version 3.2, COMSOL AB, Stockholm, Sweden, 2005.
[14] M.C. Flemings, Solidification Processing, McGraw-Hill, Inc., New York, 1974.
[15] U.R. Kattner, C.A. Handwerker, Z. Metallk. 92 (2001) 740–746.
[16] T. Sourmail, C.H. Too, H.K.D.H. Bhadeshia, ISIJ Int. 43 (2003) 1814–1820.