CALPHAD: Computer Coupling of Phase Diagrams and Thermochemistry 35 (2011) 391–395

Contents lists available at SciVerse ScienceDirect

CALPHAD: Computer Coupling of Phase Diagrams and Thermochemistry

journal homepage: www.elsevier.com/locate/calphad

A generalized computational interface for combined thermodynamic and kinetic modeling

Hua Xiong [a], Zhiheng Huang [a,*], Zhiyong Wu [a], Paul P. Conway [b]

[a] School of Physics and Engineering, Sun Yat-sen University, 135 West Xingang Road, Guangzhou, 510275, China

[b] Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, LE11 3TU, UK

ARTICLE INFO

Article history:
Received 19 March 2011
Received in revised form
25 April 2011
Accepted 18 May 2011
Available online 12 June 2011

Keywords:
Computational thermodynamics
Kinetic modeling
Microstructure
Phase field
Phase field crystal

ABSTRACT

The computational interface developed by Huang et al. (2008) [Z. Huang, P.P. Conway, R.C. Thomson, A.T. Dinsdale, J.A.J. Robinson, CALPHAD 32 (2008) 129–134] has been extended and generalized in different programming and modeling environments, which includes C, Fortran, Python and Java besides MATLAB and COMSOL Multiphysics. The generalized computational interface can be used to integrate various software packages for materials and process modeling into one programming platform, within which complicated modeling processes beyond the capability of these software packages can be achieved, such as combined thermodynamic and kinetic modeling, microstructural morphology evolution modeling for systems with arbitrary geometries and microstructure-based property prediction. The interface is applicable to all software packages that provide a dynamic-link library or DLL and the incorporation of Thermo-Calc and MTDATA are introduced in this paper. Several application examples utilizing the thermodynamic data of a Cu–Sn binary alloy system and an Fe–Cr–C ternary system are presented. In addition, modeling of solidification, using both a phase field and a phase field crystal models with the finite element method, are conducted within the integrated platform.

1. Introduction

With the maturity of computation, modeling and simulation techniques, the discipline of integrated computational materials engineering (ICME) has recently emerged, which promises to accelerate materials development and better unify design and manufacturing [1]. Developing the ICME approach could provide significant economic benefits. In particular, it will make a difference if time and cost consuming experiments can be reduced and fast and reliable materials processing simulations can be achieved.

The ICME approach does face significant challenges. It requires many versatile tools equipped with various modeling capabilities, from thermodynamic calculation, kinetic modeling to microstructural morphology evolution modeling and properties prediction when materials or devices are under multiphysics (e.g. an electromagnetic field) effects, because the material systems in practical applications are rather complicated. For example, when the sizes of solder bumps used in electronics joining processes shrink down to microscales, relatively larger intermetallic compounds (IMCs) can form, which can result in fast degradation of the mechanical properties [2]. Apart from the size effects, the geometries utilized can also cause local stress to build up during interfacial reactions and therefore can form unique phase structures [3]. Furthermore, strong couplings of various physical phenomena can occur in such phase transitions, for example electro-migration may occur as overly large current densities deform nanowires [4].

Many existing software packages for materials and process modeling are available to facilitate the ICME approach, but none of them can provide all the modeling capabilities that ICME requires. For example, the software Thermo-Calc [5] and MTDATA [6] are good at thermodynamic calculations but cannot model microstructural evolutions involving kinetics; DICTRA [7] can conduct diffusion modeling in multi-component alloy systems but the information that it provides is essentially one-dimensional in nature. This paper introduces a methodology of integrating the existing software packages to build a programming platform for microstructure-based multiphysics modeling. The main principle of such integration with MTDATA has been introduced in our earlier study [8], which accesses the thermodynamic functions from MATLAB programs by linking with the dynamic-link library (DLL) of MTDATA. This paper extends the applicability of the method to Thermo-Calc and more general scenarios. The incorporation of MTDATA and Thermo-Calc enables the integrated platform to couple thermodynamics into microstructural modeling that involves kinetics. Compared to MTDATA, Thermo-Calc has a seamless interface to DICTRA, which can also be incorporated into the platform that proposed in this study for kinetic modeling

* Corresponding author. Tel.: +86 0 20 8411 0530; fax: +86 0 20 8411 0530.
E-mail address: hzh29@mail.sysu.edu.cn (Z. Huang).

0364-5916/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.calphad.2011.05.004

such as the carburizing of steel and the coarsening of precipitates. However, COMSOL Multiphysics [9] (hereinafter referred to as "COMSOL") is chosen for solving kinetic equations, by which the finite element method (FEM) is used to solve the governing equations for phase field modeling [10] and the phase field crystal modeling [11] in systems with arbitrary geometries. The phase field crystal model self-consistently incorporates many physical features such as elastic effects, multiple crystal orientations and the nucleation and motion of dislocations [11], and therefore it is possible for the integrated platform to conduct microstructure-based property predictions. The programming interfaces with the following four languages, i.e. C, Fortran, Python and Java will be introduced respectively. The structure of the paper is organized as follows: Section 2 introduces the elements of the integrated framework; Section 3 presents application examples. In addition, a companion website [12] to this paper has been created to provide details of the implementation and computer codes.

2. The integrated framework

A schematic diagram for the computational interface is illustrated in Fig. 1. Programs written in C, Fortran, Python, Java or MATLAB are able to load the DLL of Thermo-Calc or MTDATA and then use functions exported by the DLL to calculate the thermodynamic equilibrium of a material system. Thermodynamic properties, for example, stable phase constitutions, internal energy and chemical potential, can then be retrieved. The advantages of this approach are increased simulation efficiency and flexibility in combining thermodynamic calculations with user-designed algorithms, for example, calculating the density of a system as a function of temperature. Thermo-Calc has application programming interfaces including TC-API, TQ-I and the TC MATLAB Toolbox, which are used with C, Fortran and MATLAB programs respectively. However, the DLL-linking mechanism introduced in this paper works in a unified approach and covers all the functionalities of TC-API, TQ-I, and the TC MATLAB Toolbox. In the integrated framework, COMSOL can be invoked when the kinetics of a system should be modeled and simulated, for example, to simulate dynamic processes in a material system with a specific geometry. In addition, other physical effects such as electric or magnetic fields may be coupled into the microstructure evolution process by solving the coupling equations within the COMSOL simulation framework. The modeling and simulation results can be imported into MATLAB for post-processing and visualization.

2.1. Enabling thermodynamics

The interface programs written in C, Fortran and Python are compiled and run by the free software Cygwin [13]. Java is an exception and it will be explained in the following section that introduces the Java interface. One of the purposes of using Cygwin is to unify the environment and commands for compilation, rather than use specific compilation tools, such as Visual C++ and Compaq Visual Fortran, for different programming languages. The other purpose is to reduce the complexity of loading the DLL of the thermodynamic calculation software.

For simplification, only the main principles of the linking mechanisms are presented in the following text. The examples are provided for Thermo-Calc only, but the mechanisms are the same for other thermodynamic calculation software. Detailed examples for Thermo-Calc and MTDATA can be found at the companion website to this paper.

2.1.1. The C interface

Importing thermodynamic calculations into C programs does not require the definition of the functions from the DLL of the thermodynamic calculation software, but requires a header file which is used to declare these functions. The key point is to load the DLL with a command such as the following to compile the C program:

```
gcc example.c -L. tcdlls.dll -o example
```

where "-L. tcdlls.dll" instructs the program to call the functions from the Thermo-Calc DLL directly.

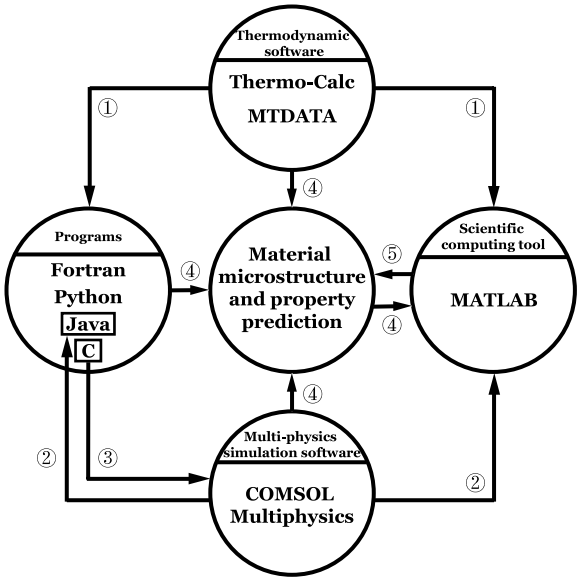

Fig. 1. The schematic diagram of the generalized computational interface. Thermodynamic calculations are imported into C, Fortran, Python, Java and MATLAB programs utilizing the DLL-linking mechanism. Combined thermodynamic and kinetic modeling can be conducted in Java, MATLAB programs and the COMSOL GUI as extensions.

2.1.2. The Fortran interface

The compilation for Fortran source files is similar to C except for an additional option. An example is provided as below:

```
gfortran example.f -L. tcdlls.dll
  -fno-underscoring -o example
```

where the option "-fno-underscoring" is used to omit the underscores attached to the names of the functions in the DLL, which is necessary to fix the mismatch of function names caused by the Fortran compilers.

2.1.3. The Python interface

Python [14] is a programming language with more than a decade development and getting increasingly used in the modeling and simulation community. By importing a function package named "ctypes", Python programs can call the functions from DLLs of thermodynamic calculation software. The command below compiles the Python program:

```
python example.py
```

The program will run automatically after the compilation. It is noted that the data types of returned values of the functions should be set properly in Python before the functions are called.

2.1.4. The Java interface

Java [15] is an object-oriented programming language with wide ranging portability and may be suitable for conducting

computing intensive modeling and simulations, taking advantage of cloud computing techniques. In this paper the Java native interface (JNI) is used to access DLLs. JNI defines the rules of data transfer between Java and DLLs, and requires an interface DLL written in C to be created. It can be achieved by using the following command from the free software MinGW [16]:

```
gcc -Wall -D_JNI_IMPLEMENTATION_ -Wl,--kill-at -I<JDK
  path>/include-I<JDK path>/include/win32
  -shared example.c -o example.dll
```

where the <JDK path> is the installation directory for the Java development kit; the options are used for activating the JNI and avoiding function name mismatches. Cygwin is not used here because it involves a much more complicated process for Cygwin to build a DLL applicable to the JNI.

2.1.5. MATLAB interface programming

The interface between MATLAB and MTDATA has been introduced in our earlier study [8]. The mechanism can be equally applied to linking MATLAB to Thermo-Calc, but it is noted that a modifier "__stdcall" should be added to the beginning of every function declaration when constructing the header file.

2.2. Enabling kinetics

COMSOL is a modeling and simulation tool capable of solving scientific and engineering problems based on partial differential equations (PDEs) and FEM. This can be used to extend the computational interface with kinetic modeling functionality. Since COMSOL version 4.0, the PDE-based multiphysics (coupled physics) modeling can be conducted both in MATLAB and Java programs. The MATLAB–COMSOL interface has been presented in our earlier study [8], and this paper will introduce the Java-COMSOL linking method as well as a new way of importing functions from DLLs into the COMSOL graphical user interface (GUI).

2.2.1. The Java-COMSOL linking

The thermodynamic calculation functions are imported into Java programs as described in Section 2.1.4 and two COMSOL function packages, i.e. "com.comsol.model" and "com.comsol.model.util" should be imported to activate the PDE and FEM based modeling. It is noted that a Java source file needs to be compiled by a specific COMSOL command:

```
<COMSOL path>\bin\win32\comsolcompile
  -jdkroot <JDK path> example.java
```

where <COMSOL path> is the installation directory for COMSOL Multiphysics, and the command for running the class file is

```
<COMSOL path>\bin\win32\comsolbatch
  -inputfile example.class
```

where the ".class" suffix should not be omitted or COMSOL will not be able to find the file.

2.2.2. Thermodynamic calculations in the COMSOL GUI

Similar to the JNI mechanism, an interface DLL written in C needs to be generated to transfer the thermodynamic calculations from DLLs to the GUI of COMSOL [17]. However, the Windows API functions, instead of Cygwin, are used to access the DLL of the thermodynamic calculation software, so that the generated DLL can be linked to COMSOL under a Windows operating system. The source codes of an example utilizing this linking mechanism can be found at the companion website, which activate a function to calculate the chemical potential of Sn in a Sn–Cu binary alloy system in the COMSOL GUI.

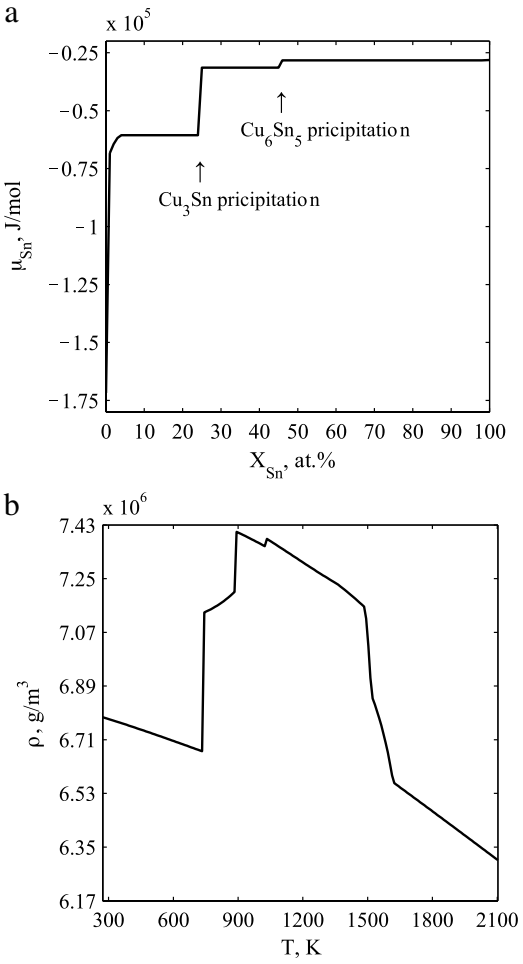

Fig. 2. (a) The chemical potential of Sn vs. the composition of Sn–Cu alloy system at $T = 513.15$ K and $P = 101\,325$ Pa, (b) The density of an Fe-0.16 Cr-0.19 C (at.%) alloy system vs. temperature at $P = 101\,325$ Pa.

3. Applications

The following application examples demonstrate the basic use of the generalized computational interface,[1] which include calculating the chemical potential of Sn in a Sn–Cu binary alloy system, calculating the density of an Fe–Cr–C ternary system, simulating the diffusion process of Cu into a Sn solder bump, and simulating the microstructure evolution during solidification using a phase field model and a phase field crystal model respectively.

3.1. Chemical potential and density calculations

Two examples of using the computational interface to calculate thermodynamic equilibrium are presented, i.e. calculating the chemical potential of Sn in a Sn–Cu alloy system and the density of an Fe–0.16 Cr–0.19 C (at.%) system. The thermodynamic data of Sn–Cu alloy are retrieved from the solder database developed by NIST [18] and the data of Fe–Cr–C alloy are from the TCFE6 database developed by Thermo-Calc. The chemical potential μ_{Sn} as a function of the mole fraction x_{Sn} is shown in Fig. 2(a), and the density ρ of the Fe–Cr–C system as a function of temperature is shown in Fig. 2(b). The calculations are conducted by C programs,

[1] Cygwin version 2.697, TC-API version 5.0, MATLAB R2010a and COMSOL v4.1 were used in the examples.

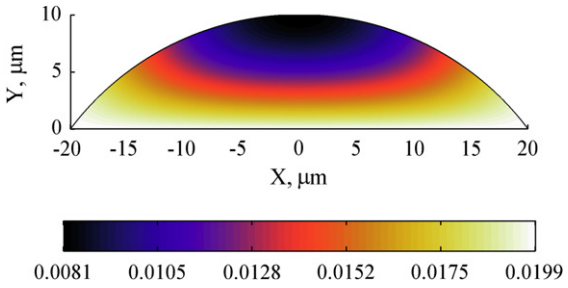

Fig. 3. A plot of the Cu concentration distribution at $t = 0.01$ s of the diffusion simulation.

and the results are imported into MATLAB for visualization. The results from Fortran, Python and Java are the same and therefore not presented, but all the source codes can be found at the companion website. Those two examples reveal the possibility for combining the CALPHAD method with user-designed algorithms, which can be extended into kinetic modeling, e.g. calculating mobility as a function of concentration during diffusion process.

3.2. Diffusion of solid Cu toward liquid Sn

This example demonstrates a combined thermodynamic and kinetic modeling and simulation conducted by the computational interface, i.e. the simulation of Cu atoms diffusing from solid Cu into liquid Sn. The distribution of the concentration of Cu within a cross-section of a microscale solder bump is calculated. Considering the effect of surface tension, the geometry of the bump is constructed as a truncated sphere, as shown in Fig. 3. The studied domain of the bump is initially pure liquid Sn. Fick's second law is set as the governing equation, which is defined using the general PDE form of COMSOL. The diffusion coefficient of Cu in liquid Sn in this model is obtained from the study of Yoshifusa et al. [19], i.e.

$$D_L = 4.2 \times 10^{-9} \times \exp\left(-\frac{11\ \text{kJ/mol}}{RT}\right) \text{m}^2/\text{s}$$

where R is the gas constant and T (=513.15 K) is the absolute temperature of the system. The Cu atoms only enter the bump from the boundary at the bottom. At the Cu–Sn interface, the concentration of Cu is assumed to remain at the saturation composition. This can be calculated by varying the composition of Cu in the liquid phase to satisfy the equilibrium state. The mole fraction of Cu at this saturation composition in liquid Sn is found to be $x_{\text{Cu}} = 0.019894$ at.%. The concentration distribution of Cu at $t = 0.01$ s of the diffusion simulation process is presented in Fig. 3, which shows the geometry effect on the diffusion resulting in concave contours of concentration. The complete Java and MATLAB source codes and the results at other times of this simulation are referred to on the companion website to this paper.

3.3. Solidification modeling

The following two examples demonstrate the flexibility of the computational interface to conduct multi-scale kinetic modeling and simulation. Two different models, i.e. a phase field model and a phase field crystal model are used to simulate different microstructural characteristics during the solidification process. The complete source codes can be found at the companion website to this paper.

3.3.1. Phase field model

A phase field model [10] is used in this example to simulate the growth of the meso-scale dendritic structure. A crystal nucleus is placed at the center of a square supercooled liquid domain, and its growth is governed by the coupling of the dynamic free energy minimization equation and the heat diffusion equation [10]:

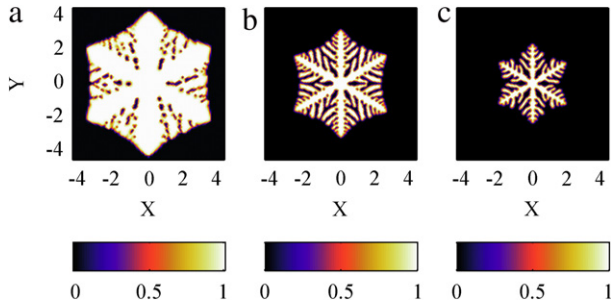

Fig. 4. Plots of the dendritic morphology during a phase field modeling and simulation of solidification at (a) $K = 1.2$, (b) $K = 1.6$ and (c) $K = 2.0$.

$$\tau\frac{\partial \Phi}{\partial t} = -\frac{\partial}{\partial x}\left(\varepsilon\frac{\mathrm{d}\varepsilon}{\mathrm{d}\theta}\frac{\partial \Phi}{\partial y}\right) + \frac{\partial}{\partial y}\left(\varepsilon\frac{\mathrm{d}\varepsilon}{\mathrm{d}\theta}\frac{\partial \Phi}{\partial x}\right) + \nabla\cdot\left(\varepsilon^2\nabla\Phi\right) + \Phi\,(1-\Phi)\,(\Phi - 0.5 + m)$$

$$\frac{\partial T}{\partial t} = \nabla^2 T + K\frac{\partial \Phi}{\partial t}$$

where Φ is the order parameter; ε (the anisotropic factor) and m (the driving force factor) are essential parameters determining the crystal morphology; K is the latent heat of phase transition; θ is a factor representing the boundary orientation; τ (=0.0003) is a constant.

The two equations are defined using the general PDE form of COMSOL. The simulated crystal morphology at $t = 0.24$ are presented in Fig. 4, where K has the values of 1.2, 1.6 and 2.0 respectively. As shown in the figures, the effect of the latent heat on the growth rate and the dendritic structure can be observed clearly. Using the integrated framework, it is possible to replace the values of thermodynamic parameters such as ε and m by those retrieved directly from thermodynamic databases. Related investigations are on going to make the phase field model of more practical use for predicting material properties and behavior.

3.3.2. Phase field crystal model

This example demonstrates solidification in a solder bump, focusing on the internal structure of the crystal grains. The crystal structure is represented by the atom density distribution and the modeling and simulation of grain growth can be conducted on a diffusive time scale with a phase field crystal model [11]. By minimizing the free energy to form a lattice of triangular symmetry in two dimensions, the microstructure evolution equation can be derived [11]:

$$\frac{\partial \Phi}{\partial t} = \nabla^2\left(-\alpha + \left(1+\nabla^2\right)^2\right)\Phi + \nabla^2\Phi^3$$

where ϕ is the dimensionless density of the material and α is the degree of supercooling. This equation is set by using the coefficient PDE form of COMSOL. A crystal nucleus is placed at the bottom initially, which is represented by the following equation [11]:

$$\Phi = A_t\left(\cos\left(\frac{\sqrt{3}x}{2}\right)\cos\left(\frac{y}{2}\right) - \frac{1}{2}\cos(y)\right) + \Phi_0$$

where A_t is the amplitude; Φ_0 is the density of the liquid phase. The simulated stable crystal structure is presented in Fig. 5. It is noted that many approximations are made to simplify the phase field crystal model for this simulation. For systems of more components or other crystal symmetries, more complicated free energy functionals need to be developed.

4. Summary

A generalized computational interface is introduced in this paper. Utilizing the DLL-linking mechanism, programs written in five different programming languages, i.e. C, Fortran, Python,

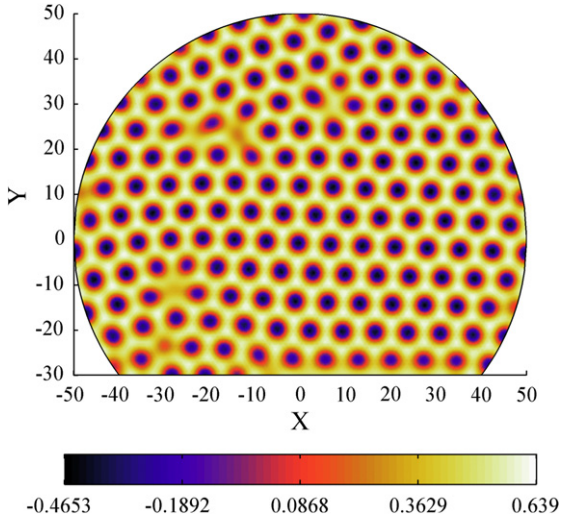

Fig. 5. A plot of the crystal grains formed during a phase field crystal modeling and simulation of solidification.

Java and MATLAB, can link with all of the thermodynamic calculation software packages that provide DLLs. Thermo-Calc and MTDATA are chosen in this paper to construct such interfaces. In addition, COMSOL can be incorporated to conduct combined thermodynamic and kinetic modeling based on PDEs and FEM. Several application examples are presented to demonstrate the use of the computational interface and demonstrate its potential on multi-scale simulation with systems of complicated geometry. It is expected that the generalized computational interface can be used to assist material research and design by integrating various modeling techniques in one programming environment and providing flexibility in utilizing thermodynamics.

Acknowledgments

One of the authors (Z. Huang) wish to acknowledge the financial supports from the Sun Yat-sen University (SYSU, grant no. 30000-3171314), Guangdong Natural Science Fund Committee (grant no. 9451027501002594), the Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP) from the Ministry of Education under grant no. 20090171120007, the Fundamental Research Funds for the Central Universities from the Ministry of Education (SYSU internal grant no. 30000-3161451), the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry (SYSU internal grant no. 2010-30000-4105346), and the National Natural Science Foundation of China (NSFC) under grant no. 51004118. Technical supports from the MTDATA team at the National Physical Laboratory (UK), Dr. G. Wang of CnTech, and Dr. Z. Wang of NERCAST on various issues of MTDATA, COMSOL Multiphysics, and Thermo-Calc are also acknowledged.

References

[1] U.R. Kattner, H.J. Seifert, CALPHAD 34 (2010) 385–386.
[2] K. Zeng, K.N. Tu, Mater. Sci. Eng. R 38 (2002) 55–105.
[3] S. Chen, C. Chiu, Scr. Mater. 56 (2007) 97–99.
[4] K. Sasagawa, M. Hasegawa, M. Saka, H. Abe, J. Appl. Phys. 91 (2002) 9005–9014.
[5] Thermo-Calc classic user's guide version S, Thermo-Calc Software AB, Stockholm, Sweden, 2008.
[6] R.H. Davies, A.T. Dinsdale, J.A. Gisby, J.A.J. Robinson, S.M. Martin, CALPHAD 26 (2002) 229–271.
[7] DICTRA user's guide version 25, Thermo-Calc Software AB, Stockholm, Sweden, 2008.
[8] Z. Huang, P.P. Conway, R.C. Thomson, A.T. Dinsdale, J.A.J. Robinson, CALPHAD 32 (2008) 129–134.
[9] COMSOL multiphysics user's guide version 4.1, COMSOL AB, Stockholm, Sweden, 2010.
[10] R. Kobayashi, J.A. Warren, W.C. Carter, Physica D 119 (1988) 415–423.
[11] K.R. Elder, N. Provatas, J. Berry, P. Stefanovic, M. Grant, Phys. Rev. B 75 (2007) 064107.
[12] http://spe.sysu.edu.cn/icme/calphad11.
[13] http://www.redhat.com/services/custom/cygwin.
[14] M. Lutz, Programming Python, second ed., O'Reilly Media, California, USA, 2001.
[15] J. Gosling, B. Joy, G. Steele, G. Bracha, The Java Language Specification, third ed., Addison-Wesley, MA, USA, 2005.
[16] http://www.mingw.org/wiki/MinGW.
[17] COMSOL multiphysics API guide version 4.1, COMSOL AB, Stockholm, Sweden, 2010.
[18] http://www.metallurgy.nist.gov/phase/solder/cusn.
[19] S. Yoshifusa, U. Sosuke, A. Tadashi, Mater. Trans. JIM 21 (1980) 383–389.