Yao Shan, BSc

Monte Carlo study of interstitial diffusion in an fcc lattice in the presence of solute atom traps

MASTER THESIS

For obtaining the academic degree
Diplom-Ingenieur

Master Programme of
Technical Physics

Graz University of Technology

Supervisor:

Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Egbert Zojer

Institute of Solid State Physics

Graz, April 2012

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……. ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……. ……..
date (signature)

Acknowledgement

I would like to thank my supervisors Ernst Kozeschnik and Egbert Zojer for their patience and support during the process of my thesis.

I also thank my family for their support and motivating help. Especially my partner Yijia Zeng, who helped me finding errors in the thesis.

Further I want to thank my fellow students for inspirational moments and help (in alphabetic order): Simon Ausserlechner, Hannes Brandner, David Egger, Christian Hartler, Iris Hehn, Stefan Kirnstötter, Robert Krisper, Lukas Ladinig, Ferdinand Rissner, Georg Spanring and Martin Stolterfoht.

Financial support by the Austrian Federal Government (in particular from the Bundesministerium für Verkehr, Innovation und Technologie and the Bundesministerium für Wirtschaft, Familie und Jugend) and the Styrian Provincial Government, represented by Österreichische Forschungsförderungsgesellschaft mbH and by Steirische Wirtschaftförderungsgesellschaft mbH, within the research activities of the K2 Competence Centre on "Integrated Research in Materials, Processing and Product Engineering", operated by the Materials Center Leoben Forschung GmbH in the framework of the Austrian COMET Competence Centre Programme, is gratefully acknowledged.

Abstract

As an introduction, a short literature review of studies on interstitial diffusion and Monte Carlo simulation is given. The main work focuses on implementing a Monte Carlo framework for rigid lattice simulations. Diffusion is realized via vacancy jumps in the substitutional lattice and self-diffusion of atoms in the interstitial lattice. For the substitutional grid, a face centered cubic lattice is chosen. The interstitial diffusion takes place on the octahedral interstices. Various checks have been done to validate the implementation. They include comparison of the numerical results with the analytic expression provided by Fick's laws of diffusion. The trapping of interstitial atoms is investigated considering a higher binding enthalpy to substitutional solute atoms. The trapping fraction is evaluated for different binding enthalpies. Differences between a single trapping position per trap and multiple trapping positions are analysed. Further the charging and discharging of interstitial sites in the presence of attractive traps is simulated and compared to recent theoretical results from Svoboda and Fischer.

Abstrakt

Zu Beginn der Diplomarbeit wird eine kurze Literaturübersicht über Studien von interstitieller Diffusion und Monte Carlo Simulationen gegeben. Die Hauptarbeit konzentriert sich dann auf die Implementierung eines Monte Carlo Moduls für Simulationen in einem starren Gitter. Diffusion erfolgt über Leerstellensprünge im substitutionellen Gitter und über Selbstdiffusion der interstitiellen Atome. Für das substitutionelle Gitter wird eine kubisch raumzentrierte Struktur angenommen, die interstitielle Diffusion findet auf den oktaedrischen Zwischengitterplätzen statt. Verschiedene Kontrollen der korrekten Implementierung werden durchgeführt. Dies inkludiert den Vergleich der numerischen Ergebnisse mit dem analytischem Ausdruck, welcher durch die Fick'schen Gesetze der Diffusion bestimmt ist. Weiters wird das Trapping der interstitiellen Atome aufgrund höherer Bindungsenthalpie zu Fremdatomen im Hauptgitter untersucht. Unterschiede zwischen einer einzigen Haftstelle pro Fremdatom im Hauptgitter und mehreren Haftstellen werden analysiert. Abschließend wird das Füllen und Entleeren des interstitiellen Gitters mit starken attraktiven Haftstellen simuliert und mit aktuellen theoretischen Ergebnissen von Svoboda und Fischer verglichen.

Contents

1 Introduction

Effects of diffusion in metals have been studied a lot in recent years, both by theoretical and experimental means. With the increasing performance of computers, the possibility to use numerical methods grows steadily. Monte Carlo simulation offers an effective and fast method to study the atomistic diffusion and its effects in a rigid lattice.

With the introduction of a simple vacancy jump mechanism by Flinn and McManus [1], many new possibilities for lattice Monte Carlo simulations were created. Many works have so far been carried out for precipitation kinetics in metals, especially iron-copper systems [2, 3, 4, 5, 6, 7, 8, 9]. Figure 1 shows a simulation of copper precipitation in an iron matrix.

In these studies only the diffusion in the substitutional grid is concerned. Small atoms like hydrogen, oxygen, carbon or nitrogen, which diffuse through the interstitial lattice, are neglected. These have an effective diffusion coefficient of up to 6 orders higher than the elements in the substitutional lattice [10]. For processes like carburization or phase transformations, these elements play a big role and have to be considered.

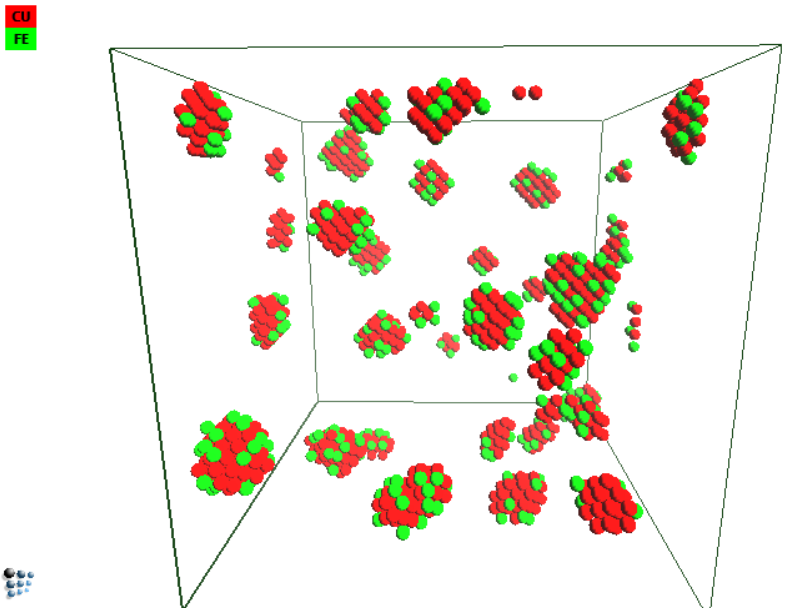


Figure 1: Monte Carlo simulation of precipitation kinetics

2 Objectives

In this work, a framework is created for running lattice Monte Carlo simulations in both the interstitial and substitutional lattice. The implementation is done in the thermokinetic software MatCalc [11] (see figure 2).

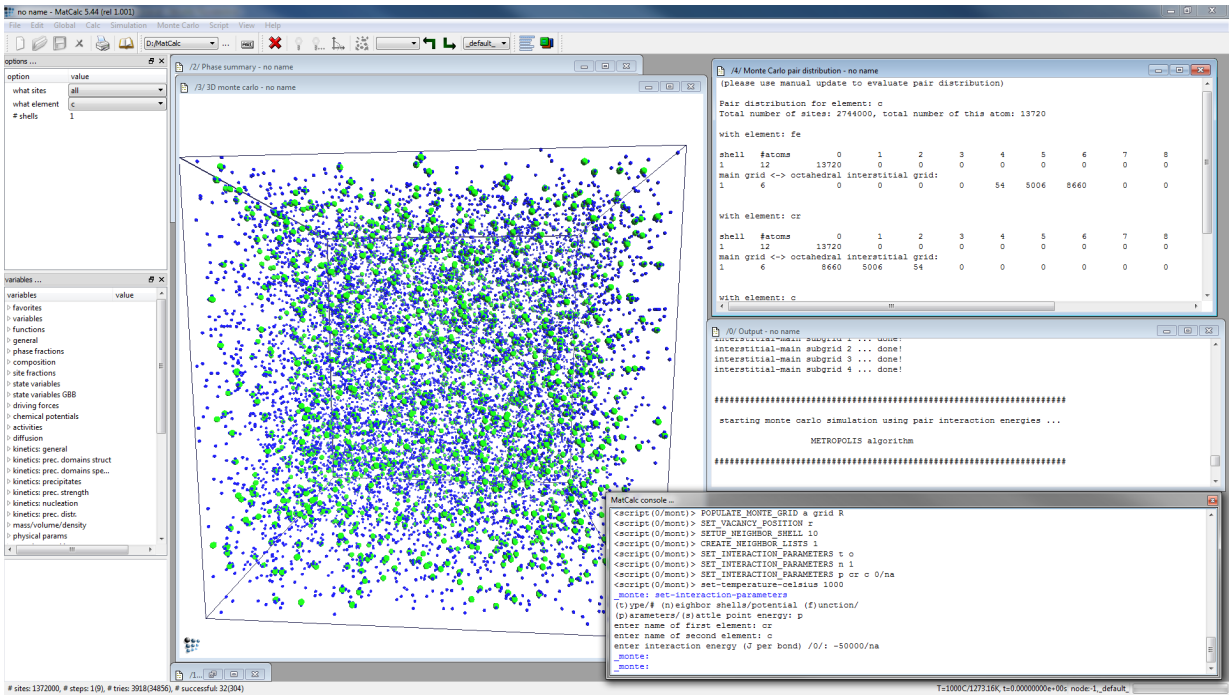


Figure 2: Monte Carlo Framework in MatCalc showing the trapping concentration evaluation

The validity of the implementation will be checked with various tests. These include

- **neighbors**: correct handling of nearest neighbors
- **interactions**: valid treatment of interaction parameters
- **physical agreement**: check with Fick's law of diffusion

Using the framework, trapping concentrations for different equilibrium constant K are evaluated. Charging and discharging simulations are run and compared with literature results.

3 Literature Review

3.1 Monte Carlo method

Monte Carlo method can be characterized by the following quote [12, 13]

> "The Monte Carlo method in statistical physics studies models of equilibrium and nonequilibrium thermodynamic systems by stochastic computer simulation"

These methods can be used for many different physical domains such as classical fluids, phase diagrams of mixtures and magnetic systems, quantum many-body problems, ... [12]. For the current work the focus is set on lattice Monte Carlo simulations, which are performed on an atomistic rigid lattice.

Based on a starting state of the system, small changes are applied to the system with Monte Carlo steps. The specific characteristic of these steps is the fact, that the result is linked to a random event. Depending on the system, various events and methods can be taken into account. The method used in this work is based on vacancy jump assisted diffusion in the main grid (the substitutional lattice) and self diffusion processes of the interstitial elements. The vacancy jump mechanism was first used by Flinn and McManus in 1961 [1], who studied the order-disorder transformations in a bcc lattice.

Depending on the type of Monte Carlo method, there exist rejection methods and rejection free methods [14]. The rejection applies for steps, which are not preferred for the system, but allows to, for example, overcome potential barriers. With the rejection free method, unlikely events will be weighted with their probability of occurrence. However, for this approach, a time scale has to be introduced. This method is also referred as the Kinetic Monte Carlo (KMC) algorithm. The first KMC simulations done with vacancy jumps was used by Young and Elcock in 1966 [15].

If the energy change of a step is negative (driving the system towards equilibrium), the new state will be accepted. In case of a positive change, the mean probability of a state a

going to a state b of higher energy, can be described with [13]

$$P\left(a \rightarrow b\right) = exp\left(\frac{-\Delta E}{k_B T}\right), \tag{1}$$

with ΔE the change of energy from state a to state b. In the rejection method, a uniform random number generator decides the acceptance of the step.

The implemented Monte Carlo method, which also uses the vacancy jump assisted mechanism, has already been used for precipitation simulations. The results have been compared with experimental results from atom probe analysis and small angle neutron scattering and show good agreement [9].

3.2 Diffusion theory

Diffusion generally describes the movement of atoms from a region with higher concentration to a region with lower concentration. In solid materials diffusion processes can be divided into two systems. One in the substitutional lattice, which is resembled by the original grid atoms and another system of the interstitial positions which are the free positions in between the lattice. In the substitutional lattice, atoms can move by exchanging their positions with vacancies. Thus, the previously mentioned method is called vacancy jump mechanism. In the interstitial lattice, the atoms can move freely on the interstitial sites [16]. There exist also other mechanisms, but these will be neglected in the present work.

The process of isotropic diffusion in one direction can be described by Fick's laws of diffusion [17]

$$J = -D\frac{\partial c}{\partial x}, \tag{2}$$

and

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}. \tag{3}$$

Where J is the flux of the atoms, c the concentration, x the diffusion length, t the diffusion time and D the diffusion coefficient, which is element-specific.

Considering one dimensional diffusion from a fixed reservoir, the second law (3) can be derived to

$$c(x,t) = c_0 \text{erfc}\left(\frac{x}{2\sqrt{Dt}}\right). \tag{4}$$

The random walk process in a lattice can be described with the Einstein formula [18]

$$6Dt = s\lambda^2, \tag{5}$$

where s is the number of random walks and λ describes the jumping distance. For a fixed lattice structure the time is

$$t = \frac{K\lambda^2}{6Dn}s, \tag{6}$$

with K being a structural factor. This result is due to the fact that the jump can not be performed in any possible direction, but depends on the lattice (lattice atoms block specific diffusion directions). n is the number of total lattice sites assigned to a single vacancy (this is a normalization factor denoting that only the vacancy can jump).

K values found in literature are given in table 1 [18]

Table 1: K values in equation (6)

lattice	K
simple cubic	0.65311
bcc	0.72722
fcc	0.78146
hcp (a-axis)	0.78146
hcp (c-axis)	0.78121

3.3 Interstitial Diffusion and Trapping effect

The effect of traps for interstitial hydrogen atoms has been studied by Oriani in 1970 [19], who showed that the usual diffusion equations can be used at low coverage of traps. However, the diffusion coefficient in the defect crystal with traps is different than in a perfect crystal. Koiwa explained this 1974 with the change of the saddle point energy near traps [20]. The ratio between defect crystal diffusion coefficient D^{dc} to perfect crystal diffusion coefficent D^{pc} is given by

$$\frac{D^{dc}}{D^{pc}} = \frac{1}{1 + K\Theta e^{-\Delta G*/k_B T}}, \tag{7}$$

where Θ is the density of trapping sites, $\Delta G*$ the free energy difference between an interstitial in a trapping site and a normal site and K a constant [21].

Lauf and Altstetter[22] did experiments with oxygen diffusing through pure Niobium and alloys. They showed that the effective diffusion in alloys was lower than in pure metals. Pressouyre and Bernstein [23] obtained similar results with hydrogen in iron-titanium alloys.

McLellan, later in 1979, derived a ratio of [24, 25]

$$\frac{D^{dc}}{D^{pc}} = \frac{1}{(1 + K\Theta e^{-\Delta G*/k_B T})^2}, \tag{8}$$

which is the same ratio as Oriani proposed, but squared.

However, de Avillez et al. in 1981 [26] and Farkas in 1983 [27] analyzed both ratios and came to the conclusion that both fit the experimental data equally well.

Farkas has studied the effective diffusion coefficient, depending on the concentration of diffusing interstitial species and on the effect of traps[28]. The work predicts, that the effective diffusion coefficient decreases with a higher concentration of traps as well as with a higher concentration of diffusing elements.

Kirchheim in 1987 [29] took the effect of changing saddle point energies near traps into account, which was mentioned earlier by Koiwa [20], and showed a sensitive relation with

diffusion. A Monte Carlo approach for interstitial diffusion was introduced, where the presence of substitutional atoms is treated as a heat bath, defining the equilibrium energies for interstitial positions. For the jumping probability in the interstitial lattice, he used the same equation (1), which was already used by Metropolis [13] for the substitutional grid. However ΔE in this case denotes the difference between the saddle point energy and the equilibrium energy of the starting position.

Norena and Bruzzoni in 2010 performed experiments for hydrogen in a modified 9%Cr-1%Mo ferritic-martensitic steel at a low temperature range and showed, that the diffusion coefficient is about 600 to 80.000 times lower than in pure iron [30].

In the work of Svoboda and Fischer [31], the diffusion of hydrogen in the presence of traps has been revisited. Based on the assumption of mass balance and trapping effects, the relation between free and trapped elements has been derived

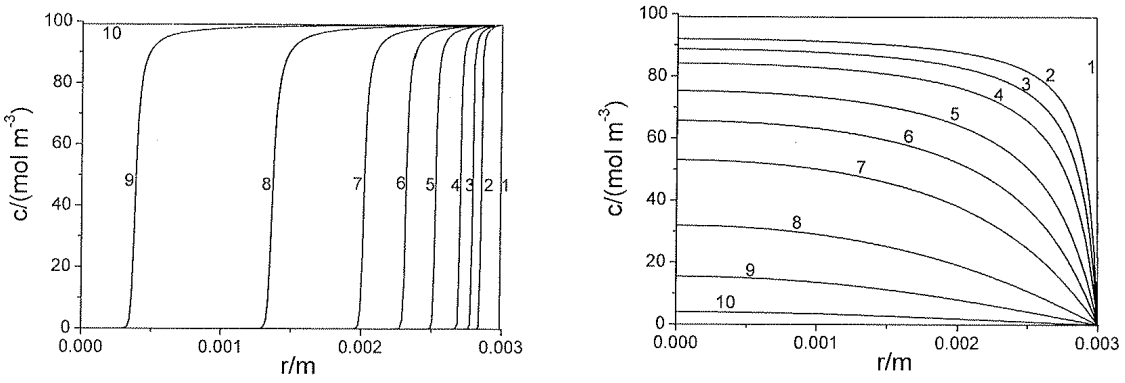
$$\frac{c_T}{c_L} = \frac{V_L}{V_T(K + V_L c_L(1 - K)}, \tag{9}$$

where c_L is the free concentration in the grid, c_T is the trapped concentration, V_L is the volume, which corresponds to one mole of interstitial positions in the lattice and V_T is the volume, which corresponds to one mole of possible trap positions. K is the equilibrium constant [19]

$$K = exp\left(\frac{-\Delta E}{k_B T}\right), \tag{10}$$

with ΔE as the trapping enthalpy of the trap. In case of Monte Carlo steps, this is the same as the probability for an element to escape the trap (see eq (1)).

Svoboda and Fischer also carried out charging and discharging simulations, based on their model. Figure 3 shows the results of charging and discharging with a low equilibrium constant K (strong traps). A notable effect of traps is, that the discharging process takes orders of 2 longer time than the charging process. Also the profiles are asynchronous in charging and discharging in contrast to normal diffusion, where the profiles are complementary.



(a) (b)

Figure 3: Charging (a) and discharging (b) in the presence of traps for $c_L^0 = 0.2$ mol/m^3, $K = 10^{-8}$ and $V_T = 10^{-2}$ m^3/mol. Atoms are entering (a) or leaving (b) the box at the right side. Shown is the concentration in dependence of the distance inside the box. The line numbers correspond for charging to 1 - 0 s, 2 - 5· 10^3 s, 3 - 10^4 s, 4 - 2· 10^4 s, 5 - 5· 10^4 s, 6 - 10^5 s, 7 - 2· 10^5 s, 8 - 5· 10^5 s, 9 - 10^6 s, 10 - 2· 10^6 s and for discharging to 1 - 0 s, 2 - 5· 10^5 s, 3 - 10^6 s, 4 - 2· 10^6 s, 5 - 5· 10^6 s, 6 - 10^7 s, 7 - 2· 10^7 s, 8 - 5· 10^7 s, 9 - 10^8 s, 10 - 2· 10^8 s [31]

4 Implementation

In this section, the compututational and numerical structure of the work is described. The source code is implemented in the scientific program MatCalc [11] as an independent module. The programming language used is C++. For the graphical user interface the package Qt, 4.7.1 [34] is used. The graphical output is drawn using OpenGL [35]. Furthermore, as a random number generator, the Mersenne Twister is used [36]. For sorting the arrays, a bubble sort algorithm is used [37].

4.1 Module architecture

In the following text, C++ objects will be written in *italic*.

The independent module composes of several classes, where the main class is the *CMCMonteGrid*, which defines the simulation grid, holds all the parameters used for simulation and runs the simulation itself. In this class, the geometry and arrangement of the grid sites is handled.

To access the class functions externally, an application programming interface (API) is provided with the *CMC_monteApp* object. This is generally used, to set the parameters for simulation, like setting up the grid, defining the interaction parameters or to get information and results about the grid.

Properties for each site of the grid, like the element, the position and the nearest neighbors, are stored in *CMCMGridSite*. The two classes for neighbors *CMCMNeighbor*, *CMCMNeighborShell* contain information about neighbor positions and are essential for the linking of the sites.

During the simulation, the state of the grid can be stored in the form of a *CMCMonteState*. Using *CMCMonteBuffer* a sequence of states can be stored, which contain additional information about the steps and time for each state. This system is adopted from the main program MatCalc and has been customized for the module.

CMCMMeanComposition is used for defining element compositions for populating the grid or setting a composition for an open boundary. The latter one, defined in *CMCMOpenBoundary*, will be explained in more detail in chapter 4.8.3.

As a random number generator, the Mersenne Twister is used in the class *MTRand*. Furthermore, there are two classes *CMCMAtomInteraction* and *CMCMIJSystem*, which are used for storing interaction parameters and handling the treatment of parameters.

To analyze the results, several classes have been implemented. *CMCMGridCondition* for setting various conditions, like elements, surrounding concentrations, grid positions and site selections. Using these conditions specific sites can be found and analyzed. *CMCMGridCriterion* puts several conditions into a logical combination. Finally, *CMCMClusterAnalysis* class is used for identifying clusters, their evolution and movement.

4.2 Grid

The grid defines the arrangement of the sites. This is achieved by defining:

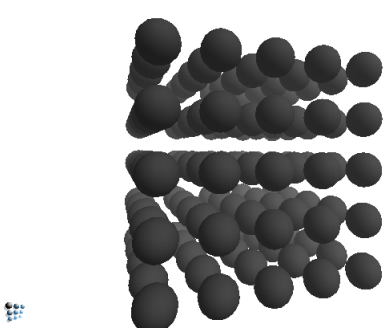
- the grid structure
- the lattice constant
- and the number of unit cells in each direction (x, y and z)

4.2.1 Grid structure

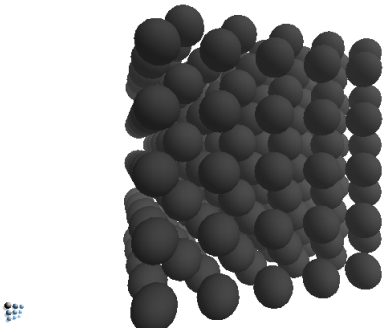
Figure 4 shows the implemented grid structures in the MC module. The simple cubic structure (figure 4a) can be thought as sites aligned on the corners of a cube. In the body centered structure (figure 4b), there exists an additional site in the center of the cube, whereas in the face centered cubic structure, there are additional sites in the center of the surfaces (figure 4c). Figure 4d shows an fcc structure with octahedral interstitial sites (small sites in the figure).

4.2.2 Lattice constant

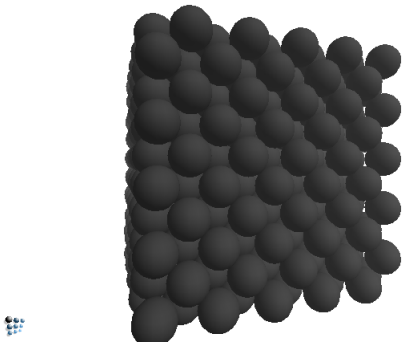
The lattice constant defines the real distance between a site and its nearest neighbors. This would be :



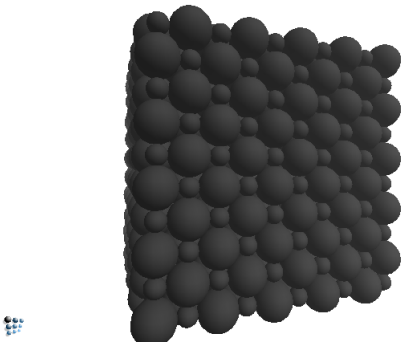
(a) simple cubic



(b) body centered cubic (bcc)



(c) face centered cubic (fcc)



(d) fcc with octahedral interstitials

Figure 4: Implemented grid structures

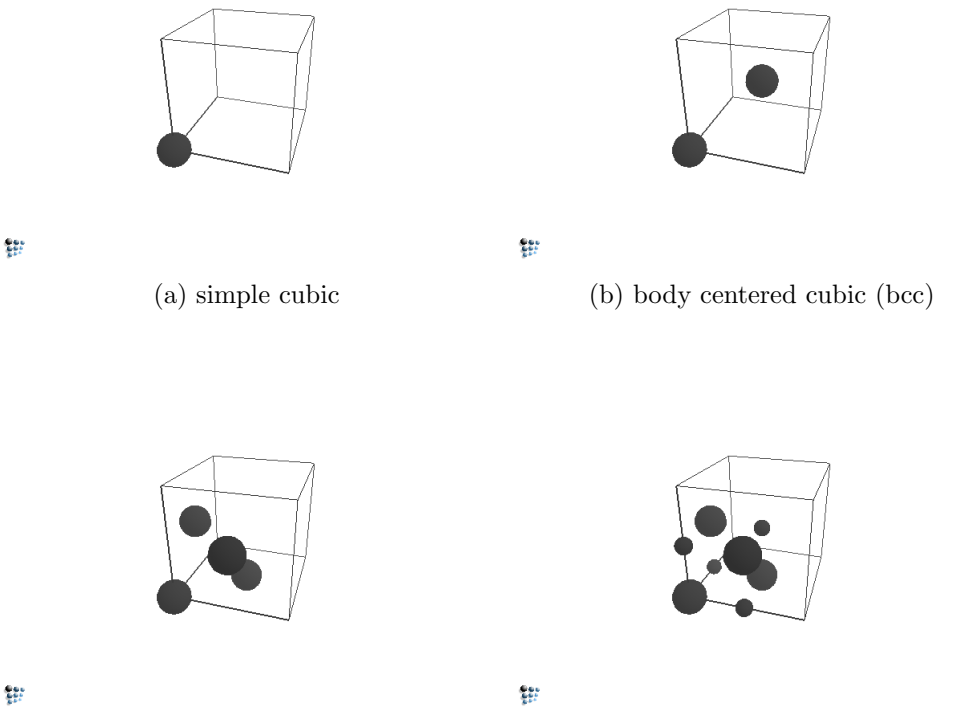
simple cubic lattice: the distance to the next site lying on an axis

body centered cubic lattice: the distance to the next site lying on the space diagonal

face centered cubic lattice: the distance to the next site lying on the face diagonal

face centered cubic lattice with interstitials: same as face centered cubic, the interstitials are ignored in the consideration of the lattice constant

4.2.3 Unit cell



(a) simple cubic

(b) body centered cubic (bcc)

(c) face centered cubic (fcc)

(d) fcc with octahedral interstitials

Figure 5: Unit cells for different grid structures

The unit cell describes the smallest possible arrangement of sites, which can be used to generate the whole lattice by reproducing this cell in different directions. Figure 5 shows the unit cells for the previously mentioned structures.

Defining the number of unit cells in each direction sets the total size of the simulation

grid. Depending on the simulation, the number of the x, y and z unit cells are matched. Recommended sizes without loss of generality are:

- $x = y = z$: for simulating direction independent systems, like precipitation or general trapping concentrations
- $x < y, z$; $y = z$: for simulating surfaces, like the process of charging or discharging
- $x > y, z$; $y = z$: for studying diffusion speed in one direction
- $z = 1$: two dimensional simulations

The size of the grid is an important part of the simulation process. Choosing a too large grid results in longer simulation time, but more precise results. On the other hand, the simulation on a smaller grid can give fast raw information about the results, which can be expected.

4.2.4 Indexing

To be able to differentiate between sites, they must be assigned unique numbers. An easy way to do this is to number them one by one, beginning from 0. This is called one dimensional indexing in the following. Using this indexing, general manipulations on the grid can be done very fast, with a loop over the sites.

However, using this method, accessing an individual site is computationally inefficient and not straightforward. Therefore, a multi dimensional indexing is introduced in parallel. Considering the simple cubic structure, the indexing would look like in figure 6. In this figure, some sites are indexed with their relative position in the (x,y,z) space.

To use this model further for bcc or fcc, it has to be modified slightly. Considering that the unit cell can be reproduced in every direction, one can access every site with an index, denoting the relative position in the unit cell, and 3 indices, denoting the unit cell itself. Figure 7 shows the indices for the sites in the first unit cell. The different positions in a unit cell are colored individually. For fcc, the index in the unit cell is defined as followed:

- 0: the site positioned on the corner

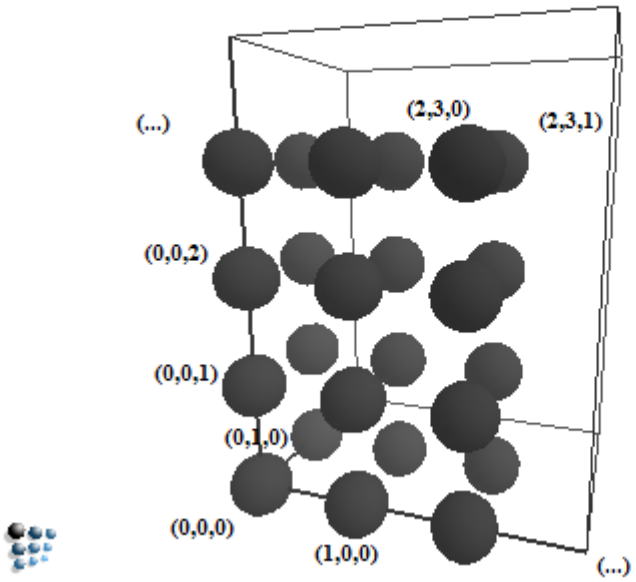


Figure 6: Indexing the simple cubic

- 1: the site positioned on the y/z-plane
- 2: the site positioned on the x/z-plane
- 3: the site positioned on the x/y-plane

The index, denoting the position in the unit cell, will be called subgrid index in the following.

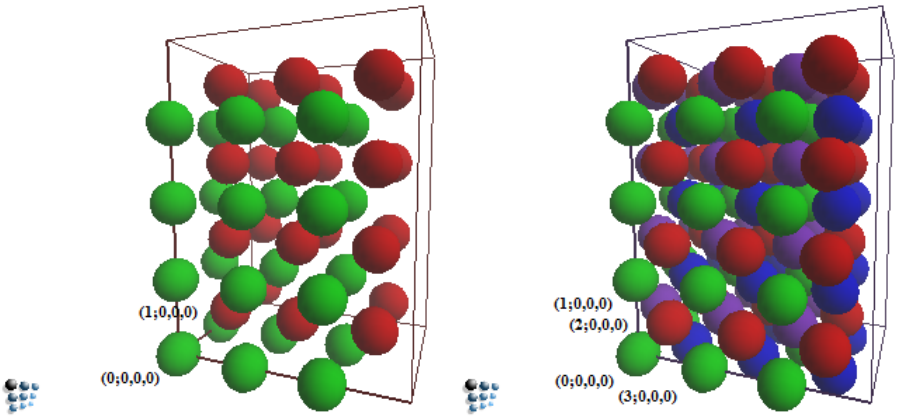


Figure 7: Indexing bcc and fcc

Furthermore, for octahedral interstitial sites in the fcc structure, the indexing is implemented in a similar way as in fcc. The subgrid indices are defined as:

- 0: the site positioned on the corner
- 1: the site positioned on the x-axis
- 2: the site positioned on the y-axis
- 3: the site positioned on the z-axis

To distinguish the index of normal fcc positions and interstitial positions, an additional letter *o* (for octahedral interstitial grid) is added. Figure 8 shows the indicees of the first unit cell.

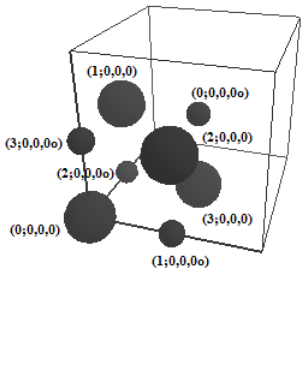


Figure 8: Indexing of fcc with interstitials

4.2.5 Transformation between indices

In the following, the transformation algorithm between the two indexing forms is presented. For the multi dimensional indexing a priority of indices is defined

1. z index
2. y index
3. x index
4. subgrid index

The priority defines the order of iteration. To get the one dimensional index of a site (s;x,y,z), the following rule applies

$$I = z + N_z \cdot y + N_z \cdot N_y \cdot x + N_z \cdot N_y \cdot N_x \cdot s, \tag{11}$$

where N_i stands for the number of unit cells in direction i.

To get the multi dimensional index, the one dimensional index and, accordingly, the modulus of the index have to be divided by the respective factors:

$$s = \lfloor \frac{I}{N_z \cdot N_y \cdot N_x} \rfloor, \tag{12}$$

$$x = \lfloor \frac{I \mathrm{mod} N_z \cdot N_y \cdot N_x}{N_z \cdot N_y} \rfloor, \tag{13}$$

$$y = \lfloor \frac{(I \mathrm{mod} N_z \cdot N_y \cdot N_x) \, \mathrm{mod} N_z \cdot N_y}{N_z} \rfloor, \tag{14}$$

$$z = \lfloor \frac{[(I \mathrm{mod} N_z \cdot N_y \cdot N_x) \, \mathrm{mod} N_z \cdot N_y] \, \mathrm{mod} N_z}{N_z} \rfloor. \tag{15}$$

4.2.6 Real space coordinates

The real space coordinates of the sites can be easily calculated, using the multi dimensional indices. Depending on the position in the unit cell, a shift in the x, y and z direction has to be defined. Table 2 shows the shifts in each direction.

The real space positions r_r are then calculated using the index positions r_i with

$$r_r = r_i \cdot a + r_s(s). \tag{16}$$

Table 2: Relative shifts in unit cell

x_s ... shift in x direction

y_s ... shift in y direction

z_s ... shift in z direction

a ... side length of unit cell

d ... lattice constant

structure and subgrid	x_s	y_s	z_s	a
bcc 0	$0.5a$	$0.5a$	$0.5a$	$\frac{2}{\sqrt{3}}d$
fcc 0	0	0	0	$\sqrt{2}d$
fcc 1	0	$0.5a$	$0.5a$	$\sqrt{2}d$
fcc 2	$0.5a$	0	$0.5a$	$\sqrt{2}d$
fcc 3	$0.5a$	$0.5a$	0	$\sqrt{2}d$
oct. interst. 0	$0.5a$	$0.5a$	$0.5a$	$\sqrt{2}d$
oct. interst. 1	$0.5a$	0	0	$\sqrt{2}d$
oct. interst. 2	0	$0.5a$	0	$\sqrt{2}d$
oct. interst. 3	0	0	$0.5a$	$\sqrt{2}d$

4.3 Neighbors

4.3.1 Substitutional grid

To speed up the simulation, it is essential to have a well ordered structure to access the nearest neighbors of a site. Neighbors will be distinguished by their distance to the observed site. There exists more neighbors with the same distance, these are summarized in a shell.

The relative difference of the indices from a neighbor site and the observed site is fixed. For example, the nearest neighbors in the unit cell always have the same relative indices. Concerning the site (0:x.y.z) in the fcc structure, the site with index (1:x.y.z) is always a nearest neighbor, which can be accessed with an index shift of (+1;+0.+0.+0) (see figure 7). However, this only applies to sites in a specified subgrid. Therefore, neighbor lists in the form of index shifts are generated for each subgrid. Figure 9 shows an example for a neighbor list in an fcc structure. The green site represents the center site, whereas the other sites are colored with respect to their subgrids.

To identify neighbors, a range d has to be defined, which is the distance up to which the

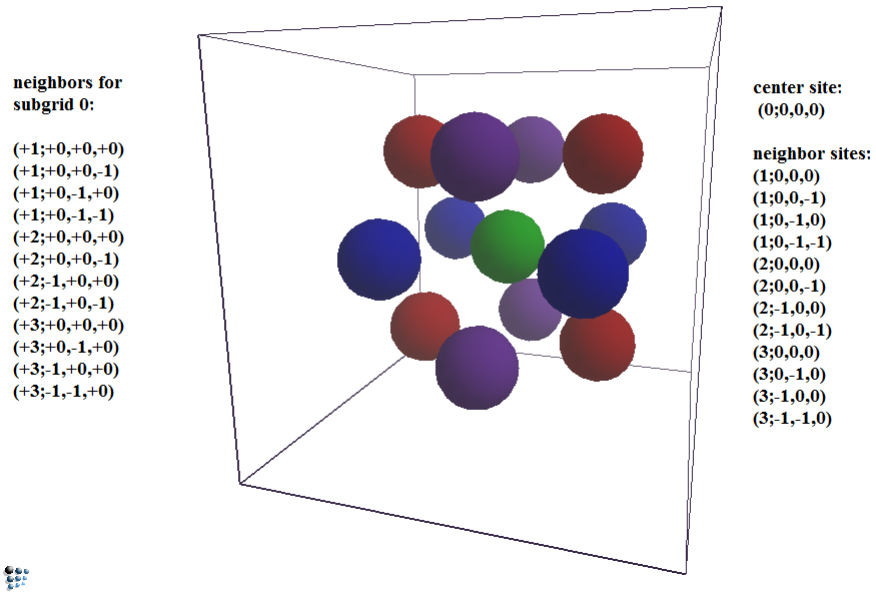


Figure 9: Neighbors of the (0:0.0.0) site in fcc

neighbors are being looked for. A grid with size d x d x d unit cells is generated. These sites are then stored in a list, with their indices and their distance to the observed site, which is positioned in the first unit cell. Using a bubble sort algorithm, this list is sorted after the distance. Sites with a greater distance than the range are discarded.

After that, the symmetric sites are scanned. For this, the relative position between the neighbor site and the observed site determines how many symmetrical sites have to be added:

- 7 symmetric sites: the x, y and z position are greater than the observed site: the symmetric sites lie in (x,y,-z),(x,-y,z),(-x,y,z),(x,-y,-z),(-x,y,-z),(-x,-y,z) and (-x,-y,-z)
- 3 symmetric sites: two positions are greater than the observed site: for example x and y greater, then the symmetric sites are in (x,-y,z),(-x,y,z) and (-x,-y,z)
- 1 symmetric site: one position is greater: for example x, then the symmetric site is (-x,y,z)

These are the general cases, which can occur. Besides, it must also be taken into consideration, that there exists neighbors whose symmetric sites are already covered. For example the (1:0.0.0) site in bcc as observed site has eight neighbors: (0:0.0.0), (0:0.0.1), (0:0.1.0),

(0:1.0.0), (0:0.1.1), (0:1.0.1), (0:1.1.0), (0:1.1.1) of which all are already included in the list.

With all neighbors determined, the neighbor shells are generated, which is a list containing the information about the number of nearest neighbors for each shell and their relative shifts. Tables 3 and 7 (in the appendix) show the generated neighbor list for the first two shells in fcc.

Table 3: Neighbor list for first shell in fcc with 12 neighbors

subgrid 0				subgrid 1				subgrid 2				subgrid 3			
Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz
1	0	0	0	-1	0	0	0	-2	0	0	0	-3	0	0	0
1	0	0	-1	-1	0	0	1	-2	0	0	1	-3	0	1	0
1	0	-1	0	-1	0	1	0	-2	1	0	0	-3	1	0	0
1	0	-1	-1	-1	0	1	1	-2	1	0	1	-3	1	1	0
2	0	0	0	1	0	0	0	-1	0	0	0	-2	0	0	0
2	0	0	-1	1	-1	0	0	-1	0	-1	0	-2	0	0	-1
2	-1	0	0	1	0	1	0	-1	1	0	0	-2	1	0	0
2	-1	0	-1	1	-1	1	0	-1	1	-1	0	-2	1	0	-1
3	0	0	0	2	0	0	0	1	0	0	0	-1	0	0	0
3	0	-1	0	2	-1	0	0	1	0	-1	0	-1	0	0	-1
3	-1	0	0	2	0	0	1	1	0	0	1	-1	0	1	0
3	-1	-1	0	2	-1	0	1	1	0	-1	1	-1	0	1	-1

4.3.2 Interstitial grid

For the interstitial grid, extra neighbor lists have to be generated. The generation is analogue to the fcc grid. However, the subgrids are aligned in a different geometry, resulting in different shifts for the neighbor lists. Tables 8 and 9 (see appendix) show the generated lists for the first two shells.

Furthermore, neighbor shells for the connection between the interstitial and the main grid have to be generated. These behave the same way as the previous lists with the additional condition that the grid is changed from the main to the interstitial or vice versa. Tables 10 and 11 show the lists for the neighbors of main grid sites. Tables 12 and 13 are the other way round (see appendix).

4.4 Boundary conditions

Boundary conditions are necessary for the border of the grid. They define the treatment of indices outside of the grid. These can be

- closed
- symmetric
- periodic

Depending on the simulation setup, different boundary conditions are recommended. While doing simulations in bulk regions, periodic conditions are preferred. Closed and symmetric conditions are recommended for surface or direction-dependent simulations. The six boundaries of the simulation box can be assigned independently. For example, two symmetric boundaries on opposite sides and 4 periodic surfaces.

The boundary conditions are represented by the neighbors of the boundary sites:

closed: there are no neighbors available. The sites will be returned as zero.

symmetric: the neighbors are the same as the neighbors in the grid, symmetric to the boundary. This can be imagined like a mirror boundary.

periodic: the neighbors are the sites, beginning from the opposite side of the boundary.

Numerically, the symmetric boundary condition are treated as follows:

$$r_s = 2 \cdot N_r - r, \tag{17}$$

with r_s as the new index of r, N_r the number of unit cells in r for $r > N_r$. For $r < 0$, the symmetric condition is:

$$r_s = -r. \tag{18}$$

The periodic condition is

$$r_p = N_r - r, \tag{19}$$

for $r > N_r$ and

$$r_p = N_r + r, \tag{20}$$

for $r < N_r$.

4.5 Grid population

To populate the grid, chemical elements have to be defined. These are basically identified only by their name. Names could be simply A, B, C, etc., if phenomenal effects are simulated or if the elements are known, the respective element names. Apart from open boundaries, which will be explained later in section 4.8.3, the total number of atoms of each type in the grid cannot change during simulations. Therefore, a fixed composition set, defining the amounts of each element, can be defined. This fixed composition, which will also be called mean composition, is given by the fractional amounts of the elements in the main grid and the fractional amounts in the interstitial grid. Since elements cannot move between the main grid and the interstitial grid, the fraction of each element has to be set in both grids separately. A big difference between the main and the interstitial grid is, that the main grid is populated completely, whereas the interstitial grid is only partially filled or even left empty. Therefore, the fractions of the elements in the main grid are also the percentaged amounts in the main grid, whereas the fraction of the interstitial elements are compared to the total fraction in the main grid. The missing part will be filled with vacancies.

To illustrate this better: The mean composition is given by

- main grid: A=92, B=5, C=3
- interstitial grid: D=3, E=1

This implies, that 92 percent of the main grid will be filled with A, 5 percent with B and 3 percent with C. The interstitial grid is then filled with 3 percent D and 1 percent E, the remaining 96 percent of the interstitial grid is left empty.

Now a gedankenexperiment: To populate a grid with 49 main grid sites, the absolute values for each element are:

- A: $\frac{92}{92+5+3} = 45.08$
- B: $\frac{5}{92+5+3} = 2.45$
- C: $\frac{3}{92+5+3} = 1.47$

The rounded values for the grid are then: 45 A, 2 B and 1 C. Using this result would only populate 48 of the total 49 grid sites.

To avoid this rounding problem a major element is defined before the actual population step. This is usually the element with the highest fraction. If two or more elements have the same fraction, the first is taken as major element (elements are sorted in the order of creation). The number of occupied sites for each element, except the major element, is then calculated and the remaining sites are reserved for the major element.

The process of populating the grid is done randomly. This is done with the following procedure:

1. the absolute number of sites for each element is calculated
2. a list of single elements is generated, with each element appearing the number of times equal to their fraction (the number calculated in the previous part)
3. this ordered list is then indexed with 0, 1, 2, 3, ...
4. a loop with n iterations is started with n the number of total sites
    - a random number r between 0 and $n-1-i$ is drawn using the Mersenne Twister algorithm, where i is the number of the current iteration (i goes from 0 to $n-1$)
    - the element with index r changes index with the element with index $n-1-i$

5. the randomized indices are used as the position index in the one dimensional index array of the grid (see section 4.2.4)

4.6 Interaction parameters

4.6.1 Pair interaction model

Before a simulation can be started, interaction parameters have to be defined. These in general comprise:

- the energy model
- the range of interaction in numbers of neighbor shells
- the potential function
- model parameters

Various energy models have been implemented in the module. The simplest one is the pair interaction model. In this approach, only the binding enthalpy of two atoms is considered. The binding enthalpy H_s for a site s is calculated by the sum of the single interactions H_{is} between the site s and its neighbors i divided by the normalized neighbor amount $N(n)$.

$$H_s = \frac{1}{N(n)} \sum_{i=1}^{n} f(i) H_{is}(i, s). \tag{21}$$

The function $f(i)$ is the potential function, which defines the decreasing influence of the enthalpy over the distance. This function is by default set as the Leonard Jones potential

$$f(i) = \left(\frac{d_i}{d_n} \right)^{-6}. \tag{22}$$

d_i is the distance to the neighbor site i, d_n is the distance to the nearest neighbors.

This potential function can however also be assigned with a user specific function, which is implemented in MatCalc with the MatCalc internal functions.

The normalized neighbor amount is given as

$$N(n) = \sum_{i=1}^{n} f(i). \tag{23}$$

The range of the interaction is a crucial factor, concerning the accuracy and the simulation time. Taking more neighbor shells for the calculation increases the accuracy at cost of efficiency.

The model parameters are defined depending on the model. For pair interactions these are the binding enthalpies for two elements. For n elements there are $n!$ possible pair interactions (see also table 4).

Table 4: Pair interactions

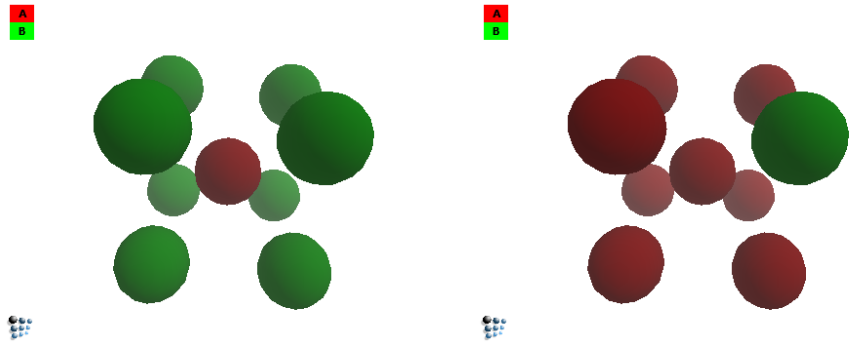
	A	B	C	...
A	H_{AA}	H_{AB}	H_{AC}	...
B		H_{BB}	H_{BC}	...
C			H_{CC}	...
...				...

4.6.2 Global energy description model

A main disadvantage of the pair interaction model is its lack to distinguish the difference between a binding between two elements A and B in an A rich environment and in a B rich environment (see figure 10).

To extend the model of pair interactions, the concentration of the environment has to be taken into account too. For this, a scan is performed for each site lying in the interaction range of the current atom to get the concentration, which is then used to get an extrapolated equilibrium enthalpy. In this model, the interaction enthalpy is extracted from the core module of MatCalc using its equilibrium calculation routine for the given concentration and extracting the binding energy of the specified two elements afterwards.

One can see, that scanning the environment for each atom now and extracting an extrapolated energy implies a huge amount of additional computing power. Since for every neighbor shell the amount of sites is fixed, depending on the grid structure, it is possible



(a) A-B binding in B environment (b) A-B binding in A environment

Figure 10: different cases of A-B bindings

to define a list containing all energies for each possible configuration, which are calculated and stored before the calculation. This way, the procedure of calculating the energy can be spared during the calculation, saving an immense amount of calculation time.

The size of the list M is defined by the number of possibilities to arrange K different elements into N spots

$$M = \begin{pmatrix} N+K-1 \\ K-1 \end{pmatrix} = \frac{(N+K-1)!}{N!(K-1)!}. \tag{24}$$

The energies are then ordered in such a way that the first composition in the list starts with the composition, in which all available sites are occupied by the first type (N elements of type 1, 0 elements of type 2, ..., 0 elements of type K). After that the further positions of the list are populated with the following scheme:

1. iterate over the elements until one element is found, which is not present indicated with the index P

2. if no such element is found the last element will be increased

3. increase element P by one, while the element before P will be decreased by one

4. if there are elements present behind element P, those element counts are added to P

Table 5 shows an example of 3 different elements and 4 possible sites ($M = 15$)

Table 5: Sorting example for global energies, 3 elements, 4 positions

I	A	B	C
1	4	0	0
2	3	1	0
3	3	0	1
4	2	2	0
5	2	1	1
6	2	0	2
7	1	3	0
8	1	2	1
9	1	1	2
10	1	0	3
11	0	4	0
12	0	3	1
13	0	2	2
14	0	1	3
15	0	0	4

To access the energy of a given composition, the index I can be calculated using equation (25). X_i denotes the count of element i in the composition.

$$I = \sum_{j=1}^{K-1} \sum_{i=0}^{N-\sum_{l=1}^{j} X_i} \begin{pmatrix} i+K-j-1 \\ K-j-1 \end{pmatrix} \tag{25}$$

A big drawback of this model is its huge amount of preallocated disk space for each composition energy. Table 6 shows some possible number of combinations in a bcc lattice. One can see that at a system of 6 different elements and 3 neighbor shells the memory of a normal computer is insufficient already. This model is only suitable for systems with low number of different elements or number of shells.

Table 6: number of possible compositions in bcc

	2 elements	3 elements	4 elements	5 elements	6 elements
1 neighbor shell	9	45	165	495	1287
2 neighbor shells	63	1260	13860	103950	594594
3 neighbor shells	819	114660	6306300	189189000	3679347672

4.6.3 Local chemical environment model

An improvement to the previous method is introduced with the LCE approach [9]. Instead of storing each possible configuration, the interaction enthalpies between an element A with an element B will now be extrapolated with a model function

$$L_{AB}^{bond} = \tilde{X}_A L_{AB}^{AA} + \tilde{X}_B L_{AB}^{BB} + \tilde{X}_A \tilde{X}_B L_{AB}^{AB}, \tag{26}$$

with X_I being the normalized concentration of element I

$$\tilde{X}_I = \frac{X_I}{\sum_{i=0}^{K} X_i}, \tag{27}$$

and L_{ST}^{XY} being the model parameter for an binding between S and T in an X and Y environment. If X and Y are the same element, this implies an environment only consisting of this element.

Analogue the interaction between two elements of the same type is defined by

$$L_{AA}^{bond} = \tilde{X}_A L_{AA}^{AA} + \tilde{X}_B L_{AA}^{BB} + \tilde{X}_A \tilde{X}_B L_{AA}^{AB}, \tag{28}$$

$$L_{BB}^{bond} = \tilde{X}_A L_{BB}^{AA} + \tilde{X}_B L_{BB}^{BB} + \tilde{X}_A \tilde{X}_B L_{BB}^{AB}, \tag{29}$$

The total energy of a site of element i is then the sum of all pairwise bond energies

$$E_{LCE,i} = \frac{1}{N(n)} \sum_{j=1}^{n} f(j) L_{ij}^{bond}, \tag{30}$$

analogue to equation (21).

Using this model, the number of parameters which has to be stored is decreased highly, allowing this model also to handle more complex systems. However, of the introduced models this is also the most time consuming one, caused by the fact, that the L values have to be calculated for each binding in each Monte Carlo step.

4.7 Determination of the parameters

For the determination of the physical parameters, especially the binding enthalpies, there are many methods. In the following the CALPHAD approach [32], which is based on Gibbs energy functions that are fitted to experimental thermo-physical data and Density Functional Theory, in short DFT [33], which is an ab initio approach, will be described in short. The software MatCalc is based on the CALPHAD method.

CALPHAD, which originally means "Calculation of Phase Diagrams" is based on thermodynamic databases. These are in general collections of model functions which are fitting the Gibbs energy

$$G = H - TS, \tag{31}$$

where H is the enthalpy, T the temperature and S the entropy, in such a way that the phase diagram can be reproduced. With the Gibbs energy known, other physical parameters can be extracted, like the enthalpy or the chemical potential which can then further be used for calculating the diffusion coefficient.

The model functions themselves are found empirically by using a large variety of free parameters that can be varied to fit known experimental or theoretical results. However these model functions only cover certain ranges of temperatures, elemental compositions or other physical parameters, depending on how wide the range of the experimental or theoretical reference were. Therefore, many different databases exist for specific type of alloys, which are also improved by time, with the availability of better experimental results.

The binding enthalpy $\tilde{H}_{ij}$ of atoms i and j is the second derivative of the system enthalpy to the compositions of the atoms. The normal enthalpy can be extracted easily from the Gibbs energy (31).

$$\tilde{H}_{ij} = \frac{\partial^2 H}{\partial X_i \partial X_j} \tag{32}$$

A strong point of this method is the ability to predict very complex multi-component systems with a decent effort. However using this method, its applicability is only as large as the reference material, used for the models.

On the other hand, there is the possibilty to extract the physical parameters from scratch using DFT. In a many-body system the Schrödinger equation is solved, by using functionals of the electron density to overcome the otherwise highly complex problem of determining the many-particle wavefunction. With the Schrödinger equation solved, the total energy of atomic configurations can be extracted and further the binding enthalpy.

A drawback of DFT is its still high complexity and therefore long calculation time for systems, which consist of 64 or more atoms in the unit cell. Therefore, binding enthalpies for elements with a low concentration cannot be calculated yet in a decent amount of time.

In the work of Farkas [28] the trapping enthalpy of chromium trapping carbon is given with -12.5 kJ/mol. This value will be used as a raw reference point in the following.

4.8 Monte Carlo method

4.8.1 Substitutional grid

The Monte Carlo steps in the main grid (substitutional grid) are done via vacancy jumps. Hereby, a vacancy must be introduced into the substitutional grid. Preferably, a major element will be substituted by the vacancy. The vacancy will then jump through the grid by switching positions with its nearest neighbors. A Monte Carlo step consists of the following steps:

1. Take a random neighbor site of the vacancy. This is done by gathering all valid nearest neighbors n and then taking a random number from 1 to n defining the interaction site.

2. Calculate the change of energy before and after the jump ΔE.

3. Depending on ΔE:

   - $\Delta E \leq 0$ the exchange of position will be carried out.

   - $\Delta E > 0$ draw a uniformly distributed random number a between 0 and 1, if $a < e^{-\frac{\Delta E}{k_B T}}$ the position will be changed.

For the first step, the amount of valid neighbor sites is usually the number of nearest neighbors of the grid structure. However, this changes on closed or symmetric boundaries. Since on closed boundaries, sites lying outside the grid will be returned as invalid sites. Those will not be considered as a valid neighbor. On symmetric boundaries, sites lying outside the grid will be returned as a mirror site from inside the grid. The problem at this point is that the sites, which are returned by the symmetric boundary condition, are valid jumping sites as well. In this case, those sites would be proposed twice, leading to an artificial flux into those sites. As a result, sites lying outside the grid will be neglected as jumping sites for symmetric boundaries, as in the case for closed boundaries.

4.8.2 Interstitial grid

The simulation steps in the interstitial grid are treated separately from the substitutional grid. The link between the two grids would be the real time, which can be calculated by the Einstein formula (5). This means that, for the first step, a substitutional step and an interstitial step is done. For both steps, the time is calculated. Preferably, the time step for the substitutional grid is longer than for the interstitial grid (the diffusion coefficient of intersitials is up to 10^{-6} smaller than the substitutional diffusion coefficient [10], thus an interstitial step takes much less time). Therefore, interstitial steps will be done, until the total time of interstitial steps is greater than the time of one substitutional step, before the next substitutional step is done. To do a simulation only in the interstitial grid, the diffusion coefficient for the main grid elements is set very high compared to the diffusion coefficient of the interstitial elements. For example if the coefficient is of 4 orders higher, a substitutional step is done after every 10000 interstitial steps.

The Monte Carlo step, in the interstitial grid itself, is handled differently than in the substitutional grid. Instead of only letting the vacancy jump, every element is allowed to jump in a single step. For this, all interstitial elements are stored in a list. However, in the case of high concentration, an atom can be fully surrounded by other elements and, thus, have no possible jump direction. During each simulation step, every element is given the possibility to jump into a vacancy spot in its nearest neighbor sites. The time step will then be calculated as the averaged sum of the single time steps.

$$\Delta t_{step} = \sum_{i=1}^{N} \Delta t_{step,i}. \tag{33}$$

As for the single element steps in the interstitial grid, the process is the same as in the substitutional grid, mentioned above.

4.8.3 Open boundaries

The main usage for the open boundary is to generate a constant site fraction at the surface of the grid. With this boundary condition it is then possible, to allow new atoms to enter the grid, or if the site fraction at the boundary is set lower than the site fraction of the grid, to deplete the grid over time. This feature is especially useful to simulate charging or discharging of a grid like in the process of carburization.

These open boundaries act similar to the closed boundaries. However, instead of returning invalid sites for out of boundary positions, holes are returned on an open boundary. As for interaction parameters, these holes are treated like vacancies. For the jumping mechanism, these holes are valid sites and can be jumped into. The result of such a jump is, that the element vanishes and is replaced with a vacancy.

To prevent the elements from vanishing all through the open boundary, a condition has to be made, which also allows new elements to get into the grid. This is done by regulating the element site fraction at the open boundary. A mean composition is assigned to an influence region of the open boundary. Given a tolerance factor, the open boundary, controls the element site fraction in the influence region, to be inside the tolerance region.

Possible events for the open boundary are

1. an element leaves the influence region forming a hole
2. an element leaves the influence region to the bulk
3. the element stays in the influence region
4. an element moves into the influence region

In the case of an element leaving the influence region, the new composition is checked against the mean composition, set for the open boundary. If the value drops below the desired composition plus tolerance, a new element will be set randomly into a free spot of the influence region. In the third case nothing will happen. For the fourth case, the

new composition will be checked against the upper limit. If too many elements are in the influence region, a random element of that kind will be removed. The reason for removing a random element and not the element, which moved into the influence region, is to prevent a flux inside of the influence region. This consideration only applies, if the influence region spreads over more than one layer of the surface. To preserve a constant flux in the influence region, incoming and outgoing elements should be distributed uniformly over the influence region. An additional flux inside the influence region would result in either depleting all elements out of the grid or charging too many elements into the grid.

5 Simulation and Discussion

All simulations in this thesis have been carried out on an fcc lattice with octahedral interstitial sites. The temperature is set to 1000°C. The high temperature is chosen due to the fact, that the fcc phase of iron is stable at this temperature.

5.1 Verification of implementation

Before simulations are run, some checks are performed.

5.1.1 Neighbors

To check, whether the neighbors were initialized correctly, the neighbor shell lists are written out and checked manually (see table 3 and table 7 to 13 in the appendix). This concerns especially the number of neighbors in each shell and their distance to the center site.

5.1.2 Interaction parameters

To check the correct handling of interaction parameters, a system with three elements is investigated. The elements are bulk elements B, trap elements T and interstitial elements I, which can interact with the traps. The only interaction parameter is set between T and I with a negative value E (a negative value stands for an attractive potential).

An fcc grid of 5x5x5 unit cells is filled with B. Then, a trapping site T is set into the middle of the simulation box. The change of the total energy is observed, which should stay constant.

In the next step, an element I is introduced at a corner of the grid in the interstitial lattice 'far away' from the trap T. In this context, 'far away' is a distance, in which the site is not located in the neighbor shells, which are evaluated for the trap. The change of the total energy of the grid is again observed, which should still be constant.

The element I will then be moved towards the trap stepwise, while the change of total energy is observed simultaneously. As soon as I enters the interaction domain of T, a change is occurring. Depending on how the potential function $V(r)$ is defined, the total energy of the grid drops at a value of $E \cdot V(r)$.

The check has been done on all first and second nearest neighbor sites around the trap to exclude the possibility of an error in the implementation.

5.2 Trapped fraction versus trapping enthalpy

With the verification of correct neighbor and interaction parameter handling, the trapping effects can now be studied. In the following, the influence of the binding enthalpy on the trapping fraction is analysed. This is particularly useful to get a feeling about how strongly the traps act.

Equilibrium simulations are done in a 70x70x70 fcc grid (results in 1.372.000 substitutional sites and 1.372.000 interstitial sites, total 2.744.000 sites) with octahedral interstitial sites and periodic boundary conditions. The grid is populated with a bulk element B and filled with trap sites T, with a site fraction of $\frac{1}{600}$ in the substitutional grid. For an interaction range of only 1 neighbor shell, the system setup leads to about 1% of the interstitial sites, to be trapping sites, due to the fact that one trapping element T can trap 6 of his nearest neighbors in the interstitial lattice. The value is a little bit below 1%, because traps can share nearest neighbor sites, if they are located close together, thus reducing the effective amount of trapping sites.

2nd and 3rd neighbor shells will also be evaluated with the same configuration. The change of the interaction range from 1 neighbor shell to more neighbor shells, can be seen as a progression from a square potential well to a step wise potential, which fits to the Lennard Jones potential (see figure 11).

For the interstitial elements I, the interstitial grid is initialized with a site fraction of 1% and distributed randomly.

With the described parameters, the simulation is run for trapping enthalpies from 0 to -90 kJ/mol in intervals of 10 kJ/mol. To check for equilibrium, the simulation is run from two initial states: One with lower trapping enthaply (leading to lower saturation of traps) and

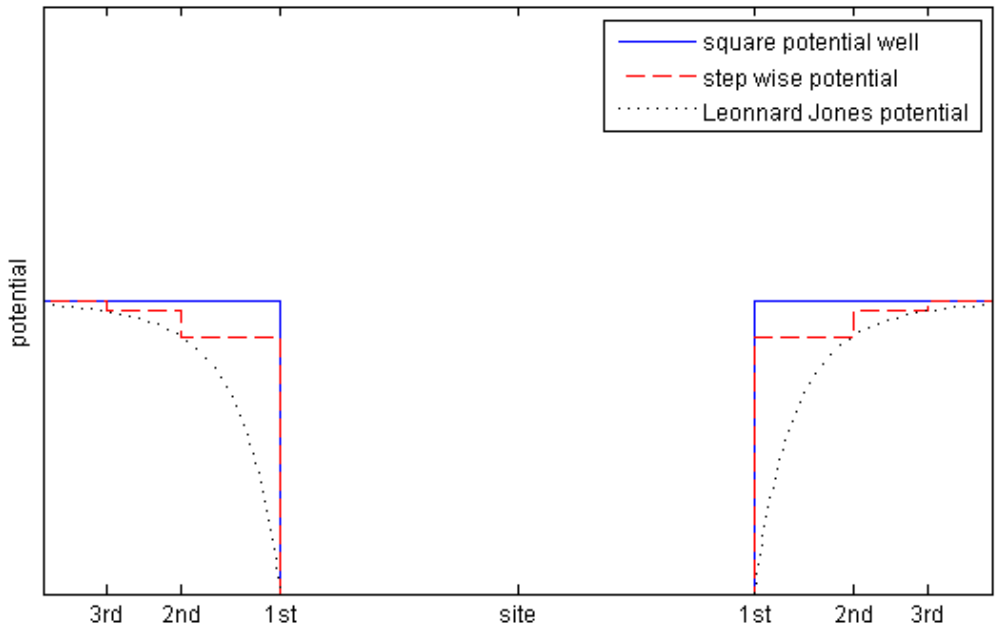


Figure 11: Potential curves: the positions of neighbors are marked on the x-axis

one with a higher trapping enthaply (leading to higher saturation of traps). Both are then set to the desired trapping enthalpy and the end values are checked respectively.

For example, to evaluate the trapping enthalpy of -20 kJ/mol, a simulation was first done for -10 kJ/mol with 500 steps[1]. Then the trapping enthalpy was decreased to -20 kJ/mol and run to equilibrium with about 1000 steps. The site fraction of trapped sites for the final state is then evaluated and stored. The check is then done by simulating with -30 kJ/mol for 500 steps and then doing again an equilibrium simulation with -20 kJ/mol. The number of steps done for each trapping enthalpy value ranges from 500 steps (0 kJ/mol) to 10000 steps (-90 kJ/mol).

The definition for a trapped site is as following: an element I counts as trapped, if at least one of its nearest neighbors in the substitutional grid is a trap element T.

The total amount of elements used in the simulations are: 1369713 B, 2286 T and 13720 I. This setup showed good statistical results with a decent simulation time.

In a further study, the effect of one trapping site per trap and more trapping sites per trap were studied. The previous results were obtained by using the nearest neighbor shells as

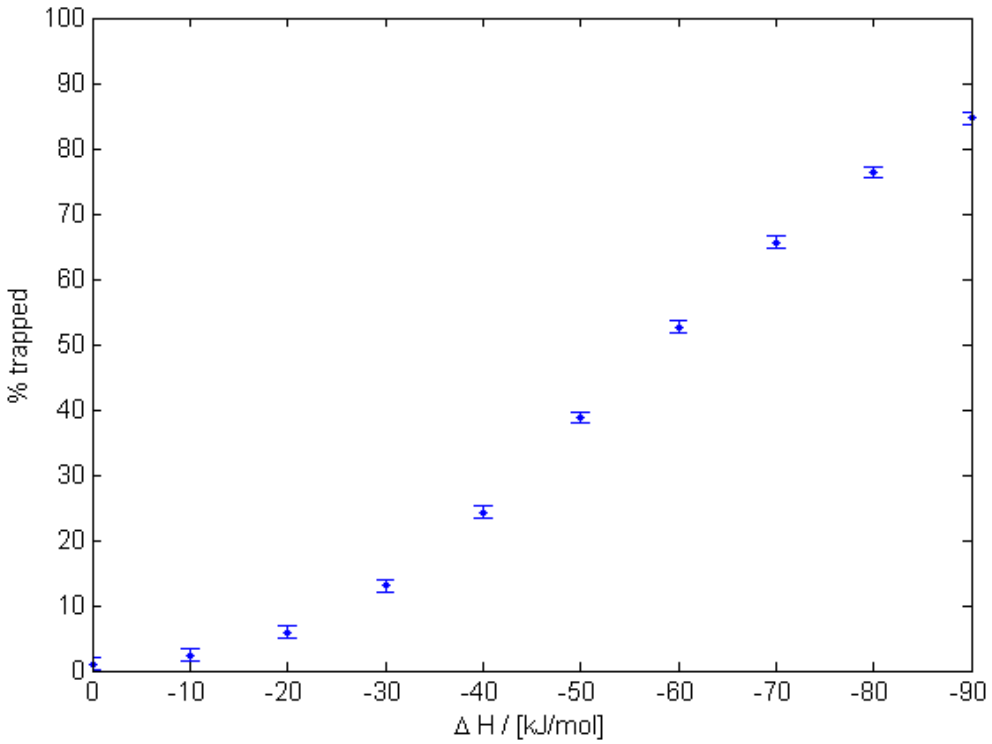
[1] 1 interstitial step means a jump for every interstitial element (see section 4.8.2)

trapping sites, which included at least 6 sites (if only nearest neighbors are considered). However many trapping models [19, 24, 31] only consider one trapping site per trap. To study that and to reproduce the theoretical models the code was modified considering only one of the neighboring sites as a valid trapping site. The nearest interstitial neighbor of a substitutional site in the positive x direction is taken for this. In the sorted neighbor lists 10 and 12, these are the first sites in case of subgrid 0 and 1, the fifth in case of subgrid 2 and the sixth in case of subgrid 3.

To keep the number of trapping sites the same as in the previous simulations, the number of substitutional traps was increased by a factor of 6.

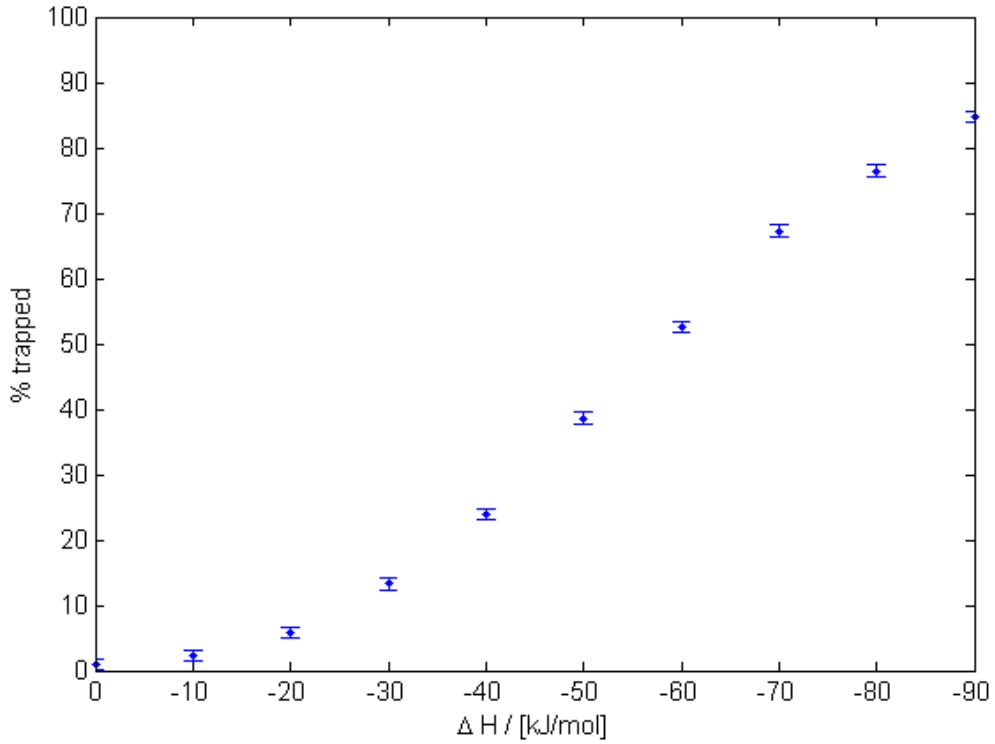
Figure 12 shows the trapped site fraction of interstitial elements in relation to the trapping enthalpy. Interaction ranges were set to 1 neighbor shell (figure 12a), 2 neighbor shells (figure 12b) and 3 neighbor shells (figure 12c).

The error is calculated with the standard deviation $\frac{1}{\sqrt{N}}$, where N is the number of interstitial atoms. For $N = 13720$ the error is about 0.8 %.

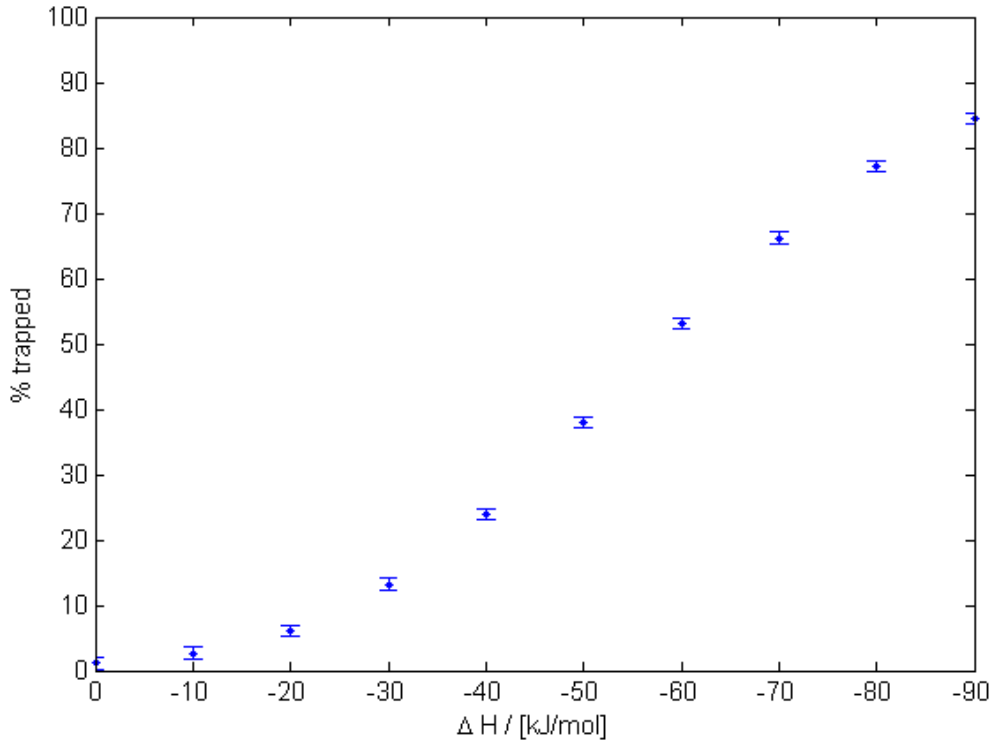


(a) 1 neighbor shell

Figure 12: Fraction of trapped atoms in dependence of the trapping enthalpy ΔH for various interaction ranges



(b) 2 neighbor shells



(c) 3 neighbor shells

Figure 12: Fraction of trapped atoms in dependence of the trapping enthalpy ΔH for various interaction ranges (continued)

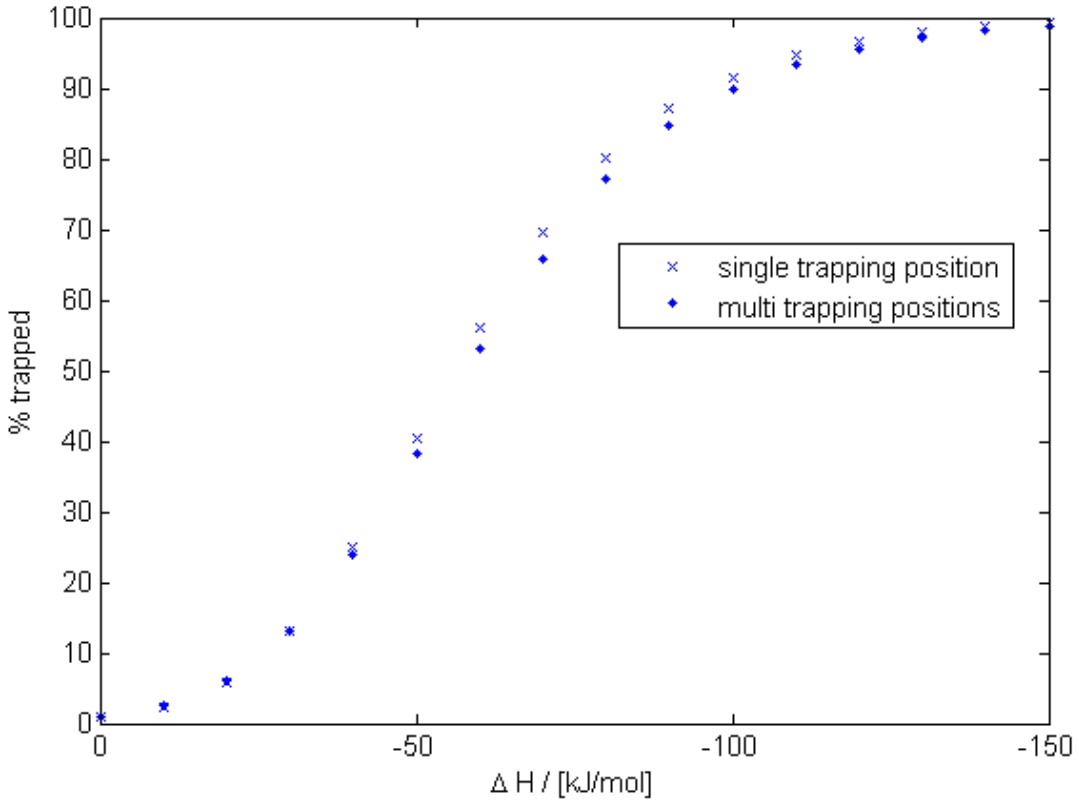


Figure 13: Fraction of trapped atoms in dependence of the trapping enthalpy ΔH for single and multiple trapping positions per trap

Figure 13 shows the results obtained by using only one trapping site per trap, compared to the previous simulation allowing trapping of all nearest neighbor sites. The error was omitted, but is still the same as in the previous simulations.

The simulation showed that there is not much difference between choosing an interaction range of 1 or 3 neighbor shells. This implies, that the potential pot approach fits very well with a stepwise potential function. The curve itself shows the form of an 'S'. Up to a trapped site fraction of 50 %, the gradient is rising and then dropping again. Since the temperature was held constant, the x-axis can also be seen as the equilibrium constant K (see equation (10)) going from 1 to 0.

Note that even at a trapping enthalpy of 0 kJ/mol, about 1% of interstitial elements are accounted as trapped. This is because the definition of a trapped element is realized by checking whether it has a trap element within its nearest neighbor shell. The interstitial elements I are uniformly distributed over the grid, since no interactions occur. However, 1 % of the sites are concerned as trap sites, due to the presence of the traps, and therefore

counted as trapped.

A comparison between single and multiple trapping positions per trap shows a significant difference of the trapped fraction for trapping enthalpies between -50 to -100 kJ/mol. Since only the arrangement of trapping sites changed (the total number of trapping sites is the same in both simulations), this shows a geometrical dependence of the trapping effect. A lower equilibrium trapping concentration is caused by the clustering of trapping sites around a single trap. Since the number of directions to access a trap is reduced by the filled traps, the possibility of an interstitial atom to get trapped is reduced and therefore decreasing the effective number of trapped atoms.

5.3 Charging and discharging

For the last series of simulations, the charging and discharging of the interstitial grid will be simulated. This is achieved by setting an open boundary condition at a border of the grid (in the simulations, this is the left side of the simulation box with x=0), which has a different element concentration than the rest of the grid. For charging, the initial site fraction is zero, whereas the open boundary will have a site fraction of 1 atom%. For discharging, the site fraction of the grid is set to 1 atom% and the open boundary site fraction to 0 atom%.

As for setting the influence region of the open boundary, only the outermost layer of the interstitial grid will be used. Note that the unit cells with x = 0 contain two layers. Therefore, only the subgrids 2 and 3 of the unit cells with x = 0 are used (these are the sites with no x shift in the unit cell, see section 4.2.4 for index declaration).

The size of the simulation grid is always taken with 50x100x100 unit cells(results in 2 million substitutional sites and 2 million interstitial sites).

At first, the agreement with Fick's law will be checked. For this, a pure grid without trapping elements is taken. The only interaction in this system is between interstitial atoms and bulk, which is constant. The interstitial atoms, thus, move according to a random walk.

Figures 14 and 15 show the diffusion profiles for charging and discharging without traps. The y-axis shows the site fraction of interstitial elements. The x-axis shows the diffusion depth in numbers of layers.

The crosses represent the simulation results, whereas the line is the expected diffusion profile according to Fick's law.

The evolution of the profile is shown by doubling the number of simulation steps between every plot.

Using Einstein's equation (5) in Fick's second law (4) for one dimensional diffusion gives

$$c(x,s) = c(0)\mathrm{erfc}\left(\frac{x}{2\sqrt{\frac{\lambda^2 K s}{6n}}}\right). \tag{34}$$

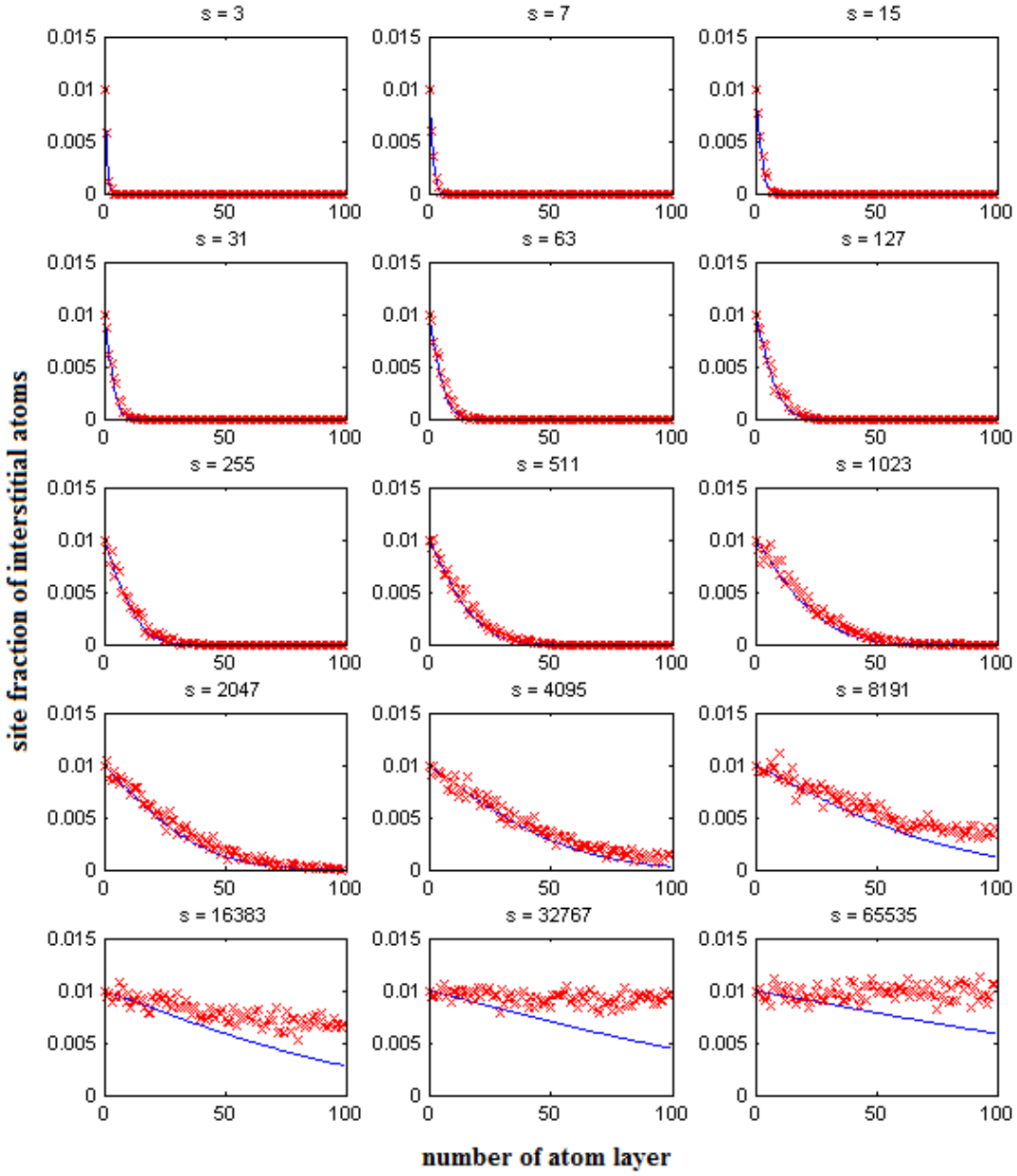


Figure 14: Charging of a grid without traps. The atoms enter the grid from the left side. The line shows the theoretical values calculated with Fick's law. s is the number simulation steps

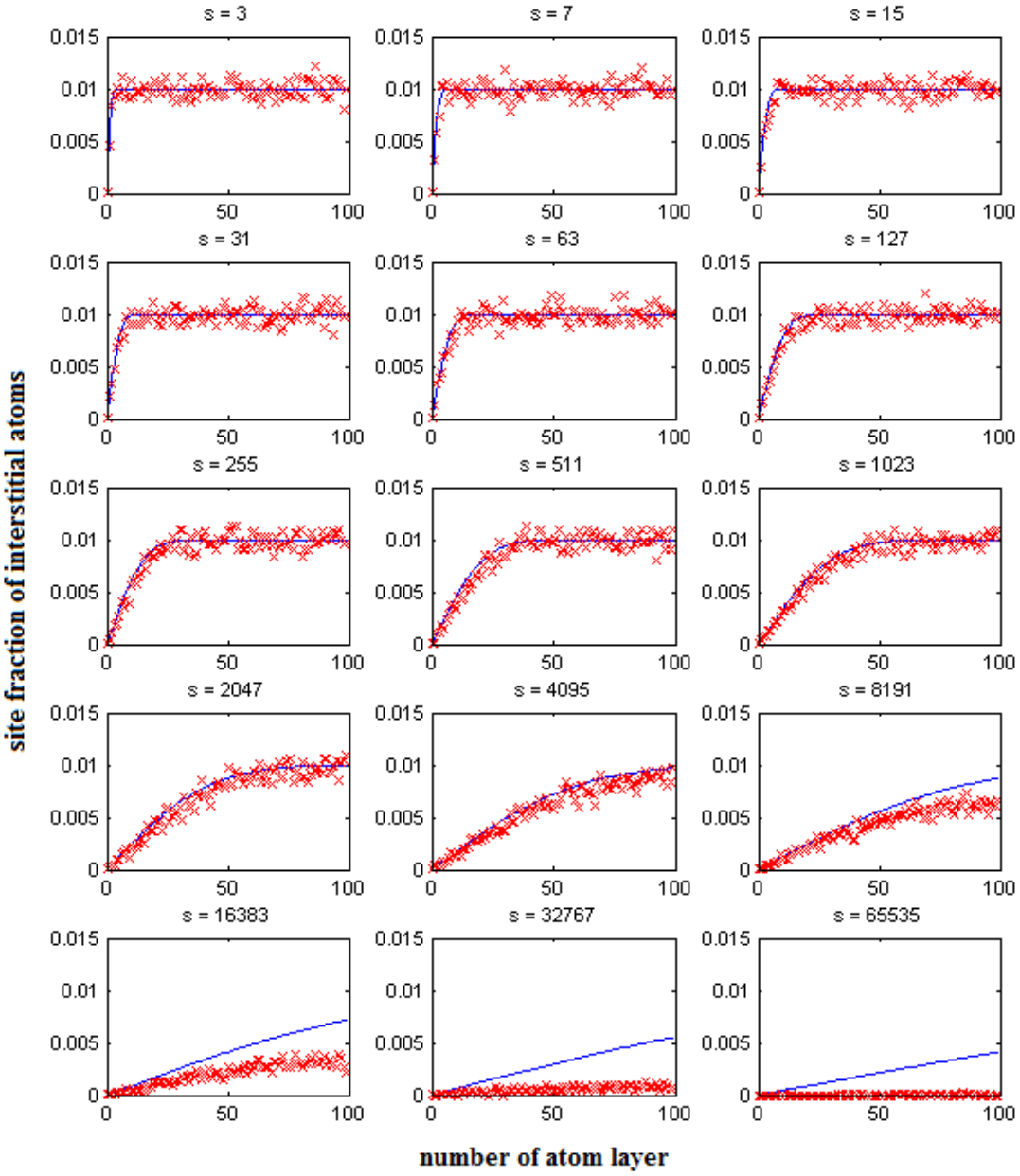


Figure 15: Discharging of a grid without traps. The atoms enter the grid from the left side. The line shows the theoretical values calculated with Fick's law. s is the number simulation steps

The time dependency t has been replaced with the number of Monte Carlo steps s. Since n is a ratio of the the jumping sites to the total sites, this factor is used as 1 (all interstitials can jump). Furthermore, the diffusion length x will be given in number of horizontal layers l in the fcc structure. The relation is

$$x = 2\sqrt{2}\lambda l. \tag{35}$$

Two layers are equal to the length of a side in the unit cell (also see table 2). Using this, equation (34) simplifies to

$$c(l,s) = c(0)\mathrm{erfc}\left(\underbrace{\sqrt{\frac{12}{K}}}_{const}\frac{l}{\sqrt{s}}\right). \tag{36}$$

The result is neither correlated to the diffusion coefficient D nor to the lattice constant λ. Using this equation, the lines were plotted in the diffusion profiles, acting as the theoretically expected values.

Figure 14 shows that the simulated charging fits very well to the theoretical values for early stages. However, after the diffusion field reaches the end of the grid, the symmetric boundary conditions result in a faster completion of the charging of the grid. Theoretically, the concentration should be zero, when the first element reaches the border. However, due to the definition of the condition, another identical element is mirrored outside the grid, resulting in a lower gradient. Nonetheless, this effect does not affect the studies in a qualitative way. After the grid is saturated, the concentration remains constant. This is an additional check for unexpected fluxes at the borders as a consequence of coding bugs.

Discharging can be seen as the complementary process of charging. The diffusion equation is analogue

$$c(l,s) = c(0)\left(1 - \mathrm{erfc}\left(\sqrt{\frac{12}{K}}\frac{l}{\sqrt{s}}\right)\right). \tag{37}$$

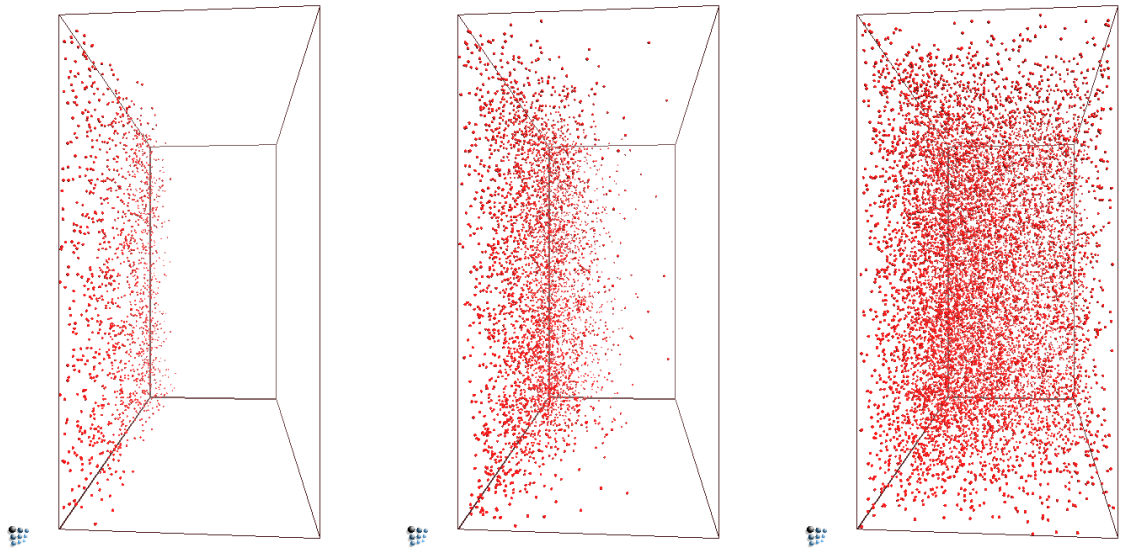
Figure 15 shows the simulation done. Again the symmetrical boundary condition changes the result, after the concentration profile reaches the right edge.

5.4 Effect of traps

In the last simulations, strong attractive traps are introduced into the main grid, with a site fraction of 1 atom %. The interaction parameter between diffusing elements in the interstitial grid and traps is set to -1e5 kJ/mol. This leads to an equilibrium constant K of about 8e-5.

Figure 16 shows a graphical representation of the simulation box after 127 steps (figure 16a), 1023 steps (figure 16b) and 8191 steps (figure 16c) without traps in the grid. For better visualisation, the bulk elements are hidden. In contrast, figure 17 shows the simulation box in the presence of traps. Traps are represented as bigger green spheres. Figure 18 shows the same simulation box with traps hidden.

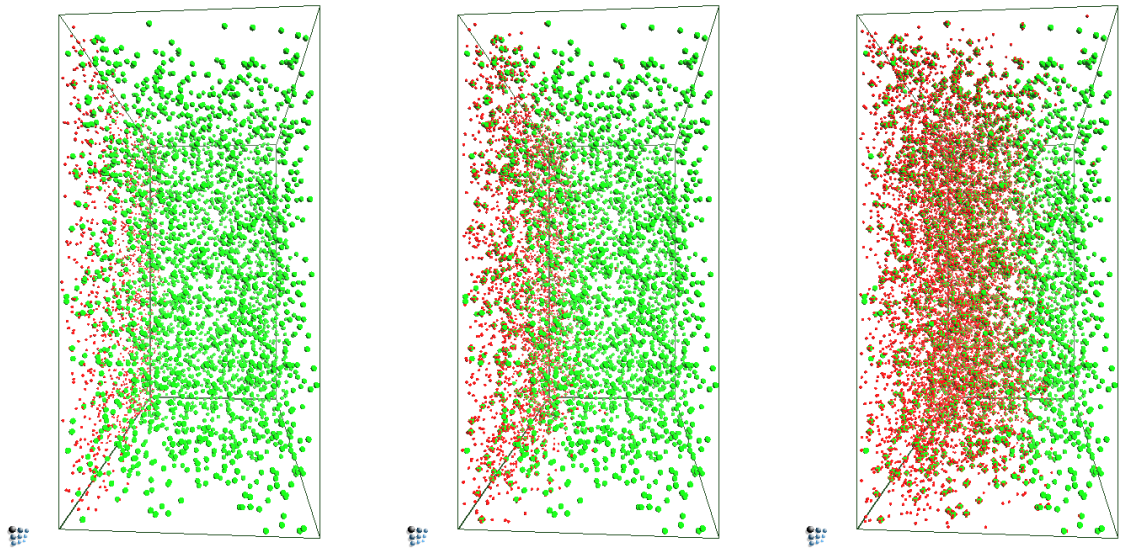
Figures 19 and 20 show the diffusion profiles in the presence of traps in a similar manner as in the previous section (compare figures 14 and 15). For the charging process, Fick's law has been used with a higher starting concentration (explanation follows in the next section) and for discharging more steps have been recorded.



(a) 127 steps (b) 1023 steps (c) 8191 steps

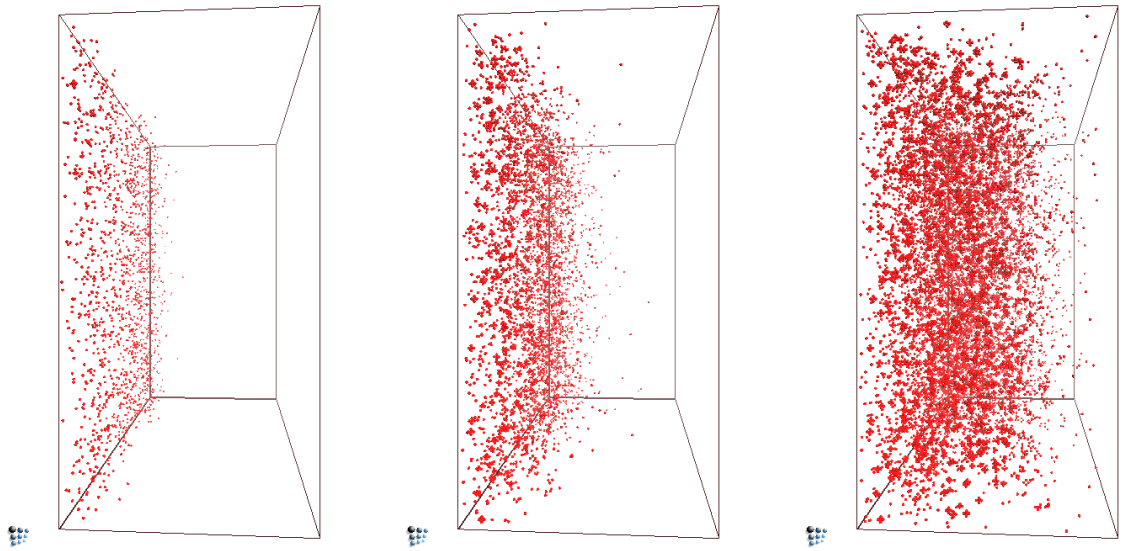
Figure 16: Graphical illustration of diffusion without traps in a 50x50x70 fcc box. The dots represent the interstitial atoms

Figure 16 and 18 show a qualitative difference between the two processes of charging and discharging. A higher end concentration in the second case can be clearly seen. Also a slower process of diffusion is noticeable in figure 18c.



(a) 127 steps (b) 1023 steps (c) 8191 steps

Figure 17: Graphical illustration of diffusion with traps in a 50x50x70 fcc box. The smaller red dots represent the interstitial atoms. The bigger green spheres represent the traps



(a) 127 steps (b) 1023 steps (c) 8191 steps

Figure 18: Graphical illustration of diffusion with traps in a 50x50x70 fcc box. The dots represent the interstitial atoms

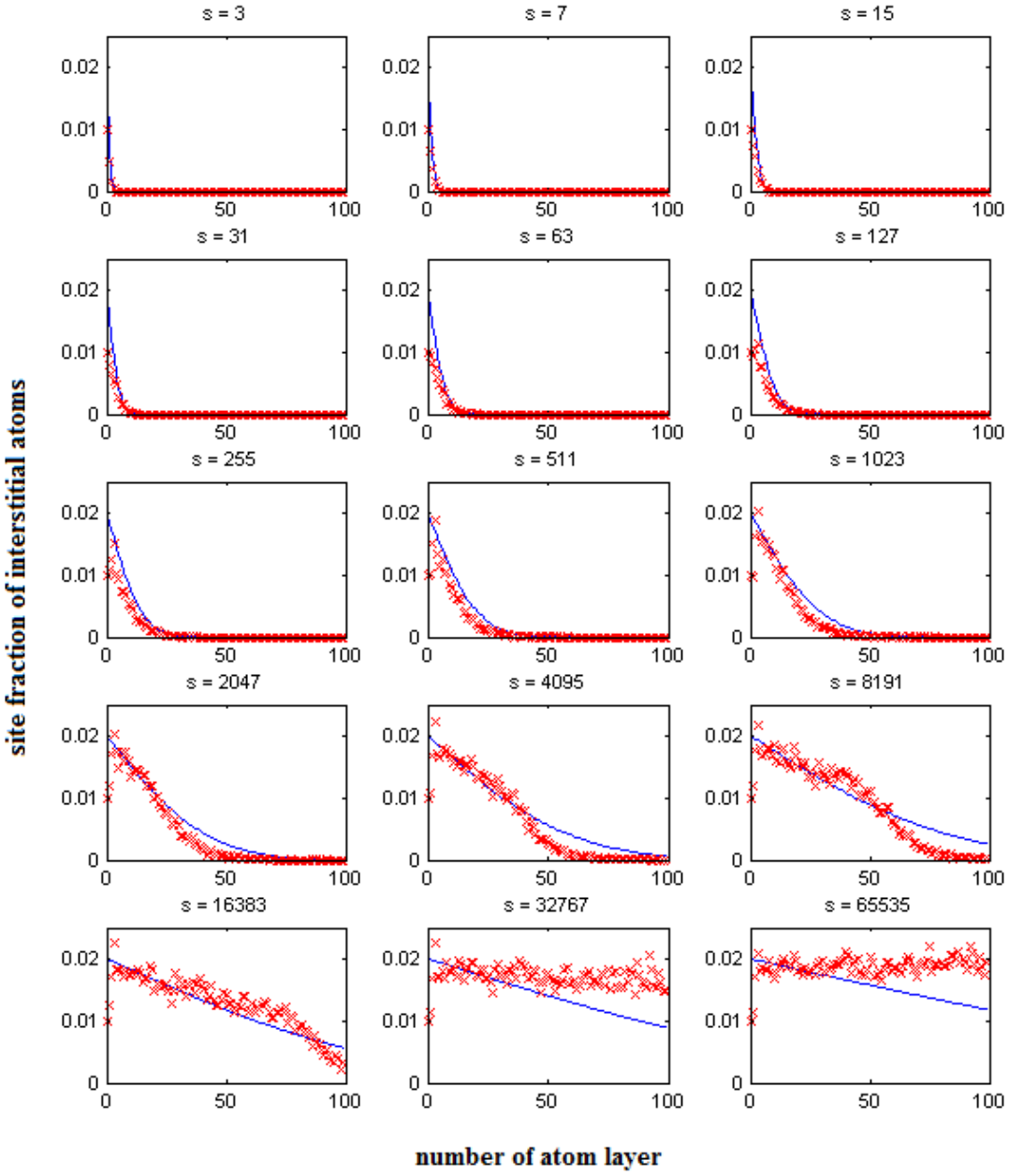


Figure 19: Charging of a grid with traps. The atoms enter the grid from the left side. The line shows the theoretical values calculated with Fick's law. s is the number simulation steps

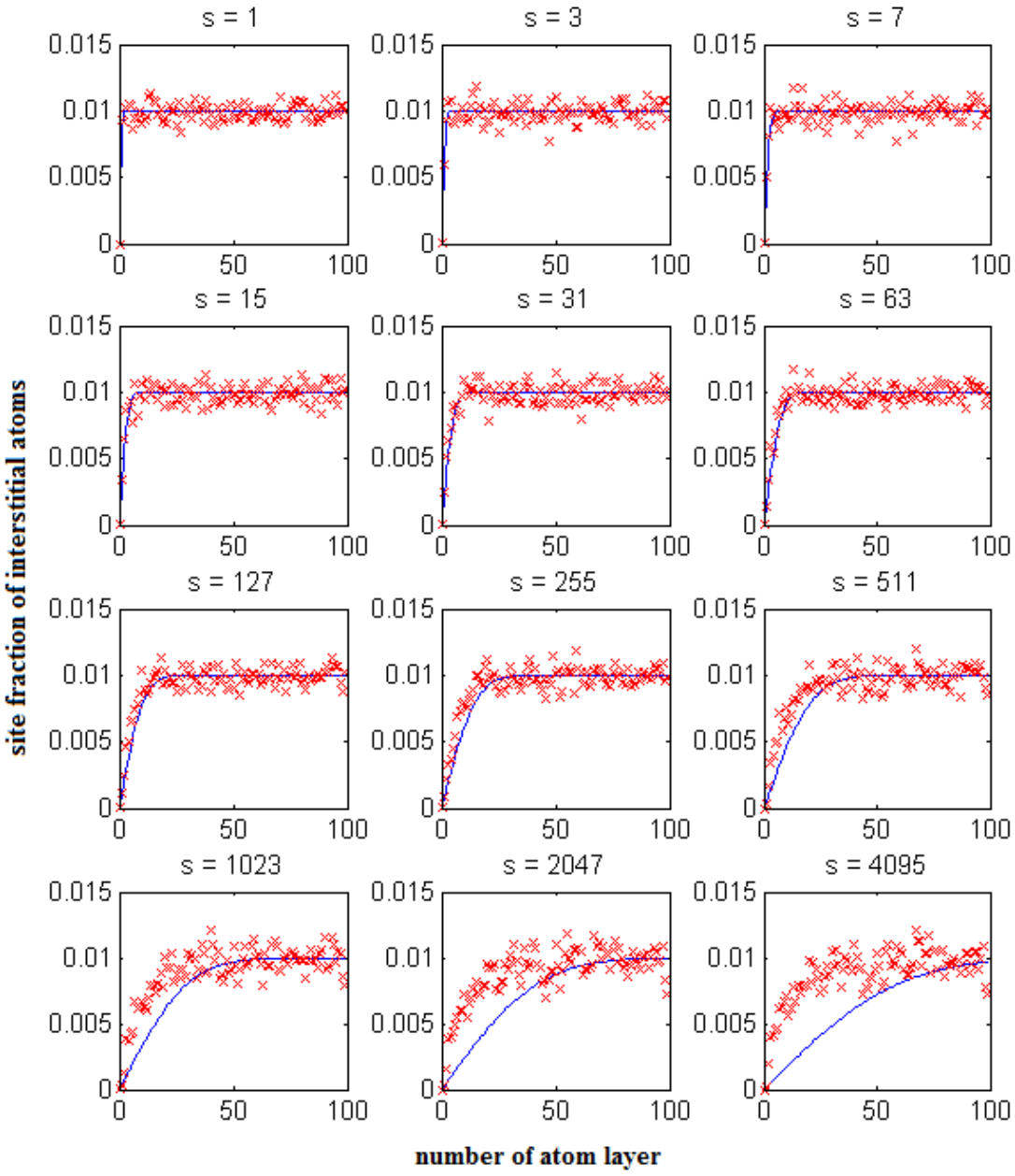


Figure 20: Discharging of a grid with traps. The atoms enter the grid from the left side. The line shows the theoretical values calculated with Fick's law. s is the number simulation steps

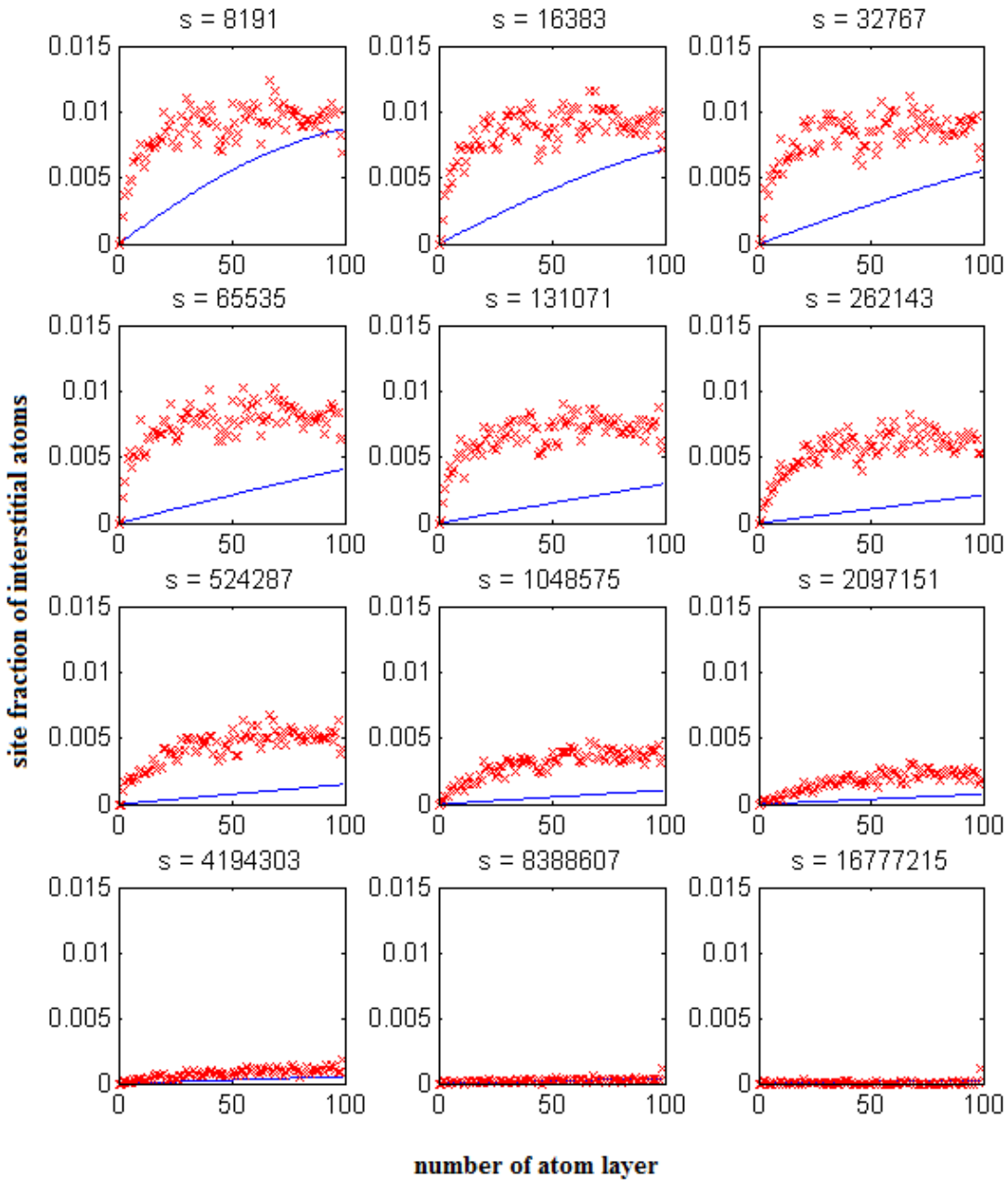


Figure 21: Discharging of a grid with traps (continued). The atoms enter the grid from the left side. The line shows the theoretical values calculated with Fick's law. s is the number simulation steps

Figure 19 shows the charging process with strongly attractive traps. The trap causes the grid to take more interstitial elements than assigned to the boundary. Because the trapped elements can hardly escape, the trap positions must be filled before other elements can pass the trap. As a result, the total concentration in the grid increases by an amount of the trapped sites, which is 1 atom %. The theoretical profile, according to Fick's law, is plotted as a reference with a higher starting concentration.

At closer inspection, one can see that there are two diffusion speeds. The first is above the trapped site fraction of 1 % (the site concentration of traps is at 1 %) and fits pretty well with Fick's law. Below 1 % the diffusion happens to be slower. This is caused by the process of filling traps, before further diffusion becomes possible.

In figure 20 and 21, the discharging process is shown. The first thing to notice is the fact that the number of steps, needed for full discharge, is about 2 orders of magnitude higher than for a full charge. The form of the diffusion profile however fits well to the form of Fick's law, but is at a much slower rate.

Compared to the results of Svoboda and Fischer, the asynchronous diffusion profiles have been reproduced, as well as the time difference of 2 orders of magnitude. However, the form of the profile during charging differs from the theoretical model. The curves in the model (see figure 3a) correspond to the lower part of the simulated profiles (below 1%).

6 Summary

A framework for simulating diffusion processes in an octahedral interstitial grid of an fcc structure has been developed. The implementation has been documented with all mechanisms used for the simulation process.

Within this framework, Fick's phenomenological law of diffusion was successfully reproduced. Various checks have been performed for the validation of the implementation, concerning neighbor handling and interaction parameters.

The evaluated trapping concentrations for different equilibrium constants K show an S-curve behaviour. The change of the interaction range, which is connected to the change from a square potential well to a discretized classical Lennard Jones potential, does not affect the characteristics of the curve in a noticeable way. However, the number of trapping sites per trap changes the results significantly. Multiple trapping sites per trap lead to a lower concentration of trapped interstitials for binding enthalpies between -50 to -100 kJ/mol at a temperature of 1000°C. This can be explained by a geometrical effect. Clustering of trapped atoms around a trap reduces the accessibility of free trap positions, thus reducing the effective trapping probability.

Charging and discharging processes have been evaluated as well and show good qualitative agreement with the results of Svoboda and Fischer [31], who have investigated hydrogen diffusion with traps theoretically. The difference in discharging and charging time, proposed by these authors, has been reproduced well with two orders of magnitude. However the characteristics of the profiles in charging differ in the theoretical and numerical approach. This will be analysed in future work.

References

[1] P. A. Flinn and G. M. McManus. Monte carlo calculation of order-disorder transformation in body-centered cubic lattice. *Physical Review*, 124(1):54–&, 1961.

[2] F. Soisson, A. Barbu, and G. Martin. Monte carlo simulations of copper precipitation in dilute iron-copper alloys during thermal ageing and under electron irradiation rid b-2917-2009. *Acta Materialia*, 44(9):3789–3800, September 1996.

[3] C. Domain, C. S. Becquart, and J. C. van Duysen. Kinetic monte carlo simulations of fecu alloys. *Microstructural Processes In Irradiated Materials*, 540:643–648, 1999.

[4] Y. Le Bouar and F. Soisson. Kinetic pathways from embedded-atom-method potentials: Influence of the activation barriers rid b-2917-2009. *Physical Review B*, 65(9):094103, March 2002.

[5] S. Schmauder and P. Binkele. Atomistic computer simulation of the formation of cu-precipitates in steels. *Computational Materials Science*, 24(1-2):42–53, May 2002.

[6] E. Vincent, C. S. Becquart, and C. Domain. Ab initio calculations of self-interstitial interaction and migration with solute atoms in bcc fe. *Journal of Nuclear Materials*, 359(3):227–237, December 2006.

[7] F. Soisson and C. C. Fu. Cu-precipitation kinetics in alpha-fe from atomistic simulations: Vacancy-trapping effects and cu-cluster mobility rid b-2917-2009. *Physical Review B*, 76(21):214102, December 2007.

[8] E. Vincent, C. S. Becquart, C. Pareige, P. Pareige, and C. Domain. Precipitation of the fecu system: A critical review of atomic kinetic monte carlo simulations. *Journal of Nuclear Materials*, 373(1-3):387–401, February 2008.

[9] P. Warczok, Y. Shan, M. Schober, H. Leitner, and E. Kozeschnik. Analysis of clustering characteristics during early stages of cu precipitation in bcc-fe. In *Solid State Phenomena (Volumes 172 - 174)*, pages 309–314, 2011.

[10] C. Hin, Y. Brechet, P. Maugis, and F. Soisson. Kinetics of heterogeneous grain boundary precipitation of nbc in alpha-iron: A monte carlo study rid b-2917-2009. *Acta Materialia*, 56(19):5653–5667, November 2008.

[11] Ernst Kozeschnik. Matcalc, the materials calculator. `http://www.matcalc.com`, October 2011.

[12] K. Binder. *Monte Carlo Methods in Statistical Physics.* Number 0-387-16514-2. Springer, 2nd edition, 1986.

[13] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[14] A. Chatterjee and D. G. Vlachos. An overview of spatial microscopic and accelerated kinetic monte carlo methods. *Journal of Computer-aided Materials Design*, 14(2):253–308, July 2007.

[15] W. M. Young and E. W. Elcock. Monte carlo studies of vacancy migration in binary ordered alloys - i. *Proceedings of the Physical Society of London*, 89(565P):735–&, 1966.

[16] G. Gottstein. *Physikalische Grundlagen der Materialkunde.* Springer-Lehrbuch. Springer, 1998.

[17] H.J. Bargel and G. Schulze. *Werkstoffkunde.* Springer-Lehrbuch. Springer, 7th edition, 2000.

[18] M.E. Glicksman. *Diffusion in solids: field theory, solid-state principles, and applications.* Number 0-471-23972-0. Wiley Interscience,, 2000.

[19] R. A. Oriani. Diffusion and trapping of hydrogen in steel. *Acta Metallurgica*, 18(1):147–&, 1970.

[20] M. Koiwa. Trapping effect in diffusion of interstitial impurity atoms in bcc lattices. *Acta Metallurgica*, 22(10):1259–1268, 1974.

[21] R. B. McLellan. The trapping of interstitial atoms in solid-solutions. *Scripta Metallurgica*, 15(11):1251–1253, 1981.

[22] R. J. Lauf and C. J. Altstetter. Diffusion and trapping of oxygen in refractory-metal alloys. *Acta Metallurgica*, 27(7):1157–1163, 1979.

[23] G. M. Pressouyre and I. M. Bernstein. Kinetic trapping model for hydrogen-induced cracking. *Acta Metallurgica*, 27(1):89–100, 1979.

[24] R. B. McLellan. Thermodynamics and diffusion behavior of interstital solute atoms in non-perfect solvent crystals. *Acta Metallurgica*, 27(10):1655–1663, 1979.

[25] R. B. McLellan and R. Kirchheim. The effect of substitutional impurity atoms on the diffusivity of an interstitial species. *Journal of Physics and Chemistry of Solids*, 41(11):1241–1246, 1980.

[26] R. R. de Avillez, R. J. Lauf, and C. J. Altstetter. Models for diffusion with trapping effects rid a-2592-2010. *Scripta Metallurgica*, 15(8):909–912, 1981.

[27] D. Farkas. Interstitial diffusion in the presence of substitutional traps. *Scripta Metallurgica*, 17(7):837–839, 1983.

[28] D. Farkas. Concentration-dependence of interstitial diffusion in the presence of traps. *Scripta Metallurgica*, 19(3):245–248, 1985.

[29] R. Kirchheim. Monte-carlo simulations of interstitial diffusion and trapping .1. one type of traps and dislocations. *Acta Metallurgica*, 35(2):271–280, February 1987.

[30] C. H. Norena and P. Bruzzoni. Effect of microstructure on hydrogen diffusion and trapping in a modified 9 *Materials Science and Engineering A-structural Materials Properties Microstructure and Processing*, 527(3):410–416, January 2010.

[31] J. Svoboda and F. D. Fischer. An improved model for hydrogen diffusion in metals with traps. *Acta Materialia*, in print, 2011.

[32] Hans Lukas, Suzana G. Fries, and Bo Sundman. *Computational Thermodynamics: The Calphad Method*. Cambridge University Press, New York, NY, USA, 1st edition, 2007.

[33] E.K.U. Gross, R.M. Dreizler, and North Atlantic Treaty Organization. Scientific Affairs Division. *Density functional theory*. NATO ASI series: Physics. Plenum Press, 1995.

[34] Nokia. Qt - cross-platform application and ui framework. `qt.nokia.com`, October 2011.

[35] Opengl, open graphics library. `http://www.opengl.org/`, October 2011.

[36] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, Jan. 1998.

[37] D.E. Knuth. *The art of computer programming*. The art of computer programming: Seminumerical algorithms. Addison-Wesley, 2001.

Appendix

Neighbor lists

Table 7: Neighbor list for second shell in fcc with 6 neighbors

subgrid 0				subgrid 1				subgrid 2				subgrid 3			
Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	-1	0	0	0	-1	0	0	0	-1	0	0	0	-1
0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	-1	0	0	0	-1	0	0	0	-1	0	0	0	-1	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	-1	0	0	0	-1	0	0	0	-1	0	0	0	-1	0	0

Table 8: Neighbor list for first shell in octahedral interstitial grid with 12 neighbors

subgrid 0				subgrid 1				subgrid 2				subgrid 3			
Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz
1	0	0	0	-1	0	0	0	-2	0	0	0	-3	0	0	0
1	0	0	1	-1	0	0	-1	-2	0	0	-1	-3	0	-1	0
1	0	1	0	-1	0	-1	0	-2	-1	0	0	-3	-1	0	0
1	0	1	1	-1	0	-1	-1	-2	-1	0	-1	-3	-1	-1	0
2	0	0	0	1	0	0	0	-1	0	0	0	-2	0	0	0
2	0	0	1	1	0	-1	0	-1	-1	0	0	-2	-1	0	0
2	1	0	0	1	1	0	0	-1	0	1	0	-2	0	0	1
2	1	0	1	1	1	-1	0	-1	-1	1	0	-2	-1	0	1
3	0	0	0	2	0	0	0	1	0	0	0	-1	0	0	0
3	0	1	0	2	0	0	-1	1	0	0	-1	-1	0	-1	0
3	1	0	0	2	1	0	0	1	0	1	0	-1	0	0	1
3	1	1	0	2	1	0	-1	1	0	1	-1	-1	0	-1	1

Table 9: Neighbor list for second shell in octahedral interstitial grid with 6 neighbors

subgrid 0				subgrid 1				subgrid 2				subgrid 3			
Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	-1	0	0	0	-1	0	0	0	-1	0	0	0	-1
0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	-1	0	0	0	-1	0	0	0	-1	0	0	0	-1	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	-1	0	0	0	-1	0	0	0	-1	0	0	0	-1	0	0

Table 10: Neighbor list for nearest neighbors from interstitial grid to main grid

subgrid 0				subgrid 1				subgrid 2				subgrid 3			
Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz
1	0	0	0	-1	0	0	0	-2	0	0	0	-3	0	0	0
1	-1	0	0	-1	-1	0	0	-2	0	-1	0	-3	0	0	-1
2	0	0	0	1	0	0	0	-1	0	0	0	-2	0	0	0
2	0	-1	0	1	0	0	1	-1	0	0	1	-2	0	1	0
3	0	0	0	2	0	0	0	1	0	0	0	-1	0	0	0
3	0	0	-1	2	0	1	0	1	1	0	0	-1	1	0	0

Table 11: Neighbor list for second nearest neighbors from interstitial grid to main grid

subgrid 0				subgrid 1				subgrid 2				subgrid 3			
Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	-1	0	-1	0	0	0	0	-1	0	0	0	0	-1
0	0	-1	0	0	0	0	1	0	0	0	1	0	0	1	0
0	0	-1	-1	0	-1	0	1	0	0	-1	1	0	0	1	-1
0	-1	0	0	0	0	1	0	0	1	0	0	0	1	0	0
0	-1	0	-1	0	-1	1	0	0	1	-1	0	0	1	0	-1
0	-1	-1	0	0	0	1	1	0	1	0	1	0	1	1	0
0	-1	-1	-1	0	-1	1	1	0	1	-1	1	0	1	1	-1

Table 12: Neighbor list for nearest neighbors from main grid to interstitial grid

subgrid 0				subgrid 1				subgrid 2				subgrid 3			
Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz
1	0	0	0	-1	0	0	0	-2	0	0	0	-3	0	0	0
1	1	0	0	-1	1	0	0	-2	0	1	0	-3	0	0	1
2	0	0	0	1	0	0	0	-1	0	0	0	-2	0	0	0
2	0	1	0	1	0	0	-1	-1	0	0	-1	-2	0	-1	0
3	0	0	0	2	0	0	0	1	0	0	0	-1	0	0	0
3	0	0	1	2	0	-1	0	1	-1	0	0	-1	-1	0	0

Table 13: Neighbor list for second nearest neighbors from main grid to interstitial grid

subgrid 0				subgrid 1				subgrid 2				subgrid 3			
Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz	Δs	Δx	Δy	Δz
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	-1	0	0	0	-1	0	0	-1	0
0	0	1	0	0	0	-1	0	0	-1	0	0	0	-1	0	0
0	0	1	1	0	0	-1	-1	0	-1	0	-1	0	-1	-1	0
0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1
0	1	0	1	0	1	0	-1	0	0	1	-1	0	0	-1	1
0	1	1	0	0	1	-1	0	0	-1	1	0	0	-1	0	1
0	1	1	1	0	1	-1	-1	0	-1	1	-1	0	-1	-1	1

Simulation script

```
$ ******************************************************************************************************
$ ************************************ GENERAL INFORMATION ***********************************************
$ ******************************************************************************************************

$ Template script for carburization using monte carlo simulations.
$
$ Creation date: 2011-6-9, for MatCalc-version (5.43 (rel. 1.004))
$ Author: Y. Shan
$ last update: 2011-11-28, for MatCalc-version (5.44 (rel. 0.024))


$ ******************************************************************************************************
$ ************************************ SETUP INFORMATION *************************************************
$ ******************************************************************************************************

$ verify correct MatCalc version (is accessible as internal variable)
if (matcalc_version<5431004)
   send-dialog-string "MatCalc version must be 5.43.1004 or higher to run this script. Stopping."
   stop_run_script                                    $ stop script
endif

use-module core                          $ use core module for kinetic simulation

close-workspace f                        $ close any open workspace without asking to save
                                         $ this feature is particularly useful for debugging purposes
new-workspace

set-workspace-info MatCalc Monte Carlo Carburization
set-workspace-info +
set-workspace-info +

$ ******************************************************************************************************
$ ************************************ GLOBAL VARIABLES **************************************************
$ ******************************************************************************************************
```

```
set-variable-value num_sites_yz 70       $ number of unit cells in y,z direction
set-variable-value num_sites_x 50        $ number of unit cells in x direction
set-variable-value num_shells 1          $ number of nearest neighbor shells used for calculations
set-variable-value c_amount 0.01         $ carbon content
set-variable-value cr_amount 0.01/6      $ chromium content
set-variable-value use_pure_boundary 0   $ defines whether the open boundary should have traps
set-variable-value c_cr_interaction -2e5/na $ interaction energy between c and cr
set-variable-value c_c_interaction 0     $ interaction energy between c and c
set-variable-value sim_steps 1e4         $ number of simulation steps
set-variable-value decarb 0              $ simulate decarburization

$ ********************************************************************************************
$ ************************************* SYSTEM SETUP *****************************************
$ ********************************************************************************************

set-variable-value simulation_option 1           $ default value for interaction type

@send-console-string
@send-console-string
@send-console-string Possible simulation options
@send-console-string   1 ... carburization with 1at.% C without traps
@send-console-string   2 ... carburization with 1at.% C, 5at.% Cr traps
@send-console-string   3 ... manual selection
@input-variable-value simulation_option "Enter selection: "

if (simulation_option==1)
   set-variable-value c_amount 0.01        $ c amount for octahedral interstitial grid
   set-variable-value cr_amount 0.00       $ cr amount in substitutional grid acting as traps
elseif (simulation_option==2)
   set-variable-value c_amount 0.01        $ c amount for octahedral interstitial grid
   set-variable-value cr_amount 0.05       $ cr amount in substitutional grid acting as traps
elseif (simulation_option==3)
   set-variable-value c_amount
   set-variable-value cr_amount
   set-variable-value num_sites_yz
   set-variable-value num_sites_x
   set-variable-value use_pure_boundary
   set-variable-value c_cr_interaction
$   set-variable-value c_c_interaction
   set-variable-value sim_steps
   set-variable-value decarb
else
   send-dialog-string Invalid selection for simulation option. Select 1, 2 or 3.
   send-dialog-string Script is interrupted.
   stop-run-script  $ stop script
endif

$ ********************************************************************************************
$ ********************** DATABASES; CHEMICAL COMPOSITION; SELECTED PHASES *********************
$ ********************************************************************************************

open-thermo-database mc_sample_fe.tdb                  $ thermodynamic database

select-elements C Fe Cr
select-phases fcc_a1
```

```
set-reference-element fe

read-thermodyn-database

read-mobility-database mc_sample_fe.ddb                 $ diffusion database

$ define nominal composition
enter-composition x fe=1.0-c_amount-cr_amount c=c_amount cr=cr_amount

$ initialize with some equilibrium
set-temperature-celsius 1000                             $ define something
set-automatic-startvalues                               $ initiate equilibrium calculation (estimate variables)
calculate-equilibrium                                   $ calculate equilibrium state


$ ********************************************************************************************
$ ********************************* MONTE CARLO SIMULATION GRID ******************************
$ ********************************************************************************************

use-module monte                                        $ switch to monte carlo module

create-monte-grid f num_sites_x num_sites_yz num_sites_yz $ create fcc simulation grid
create-interstitial-grid o                              $ create an octahedral interstitial grid
define-system l fcc_a1                                  $ link system with fcc_a1 phase from core

create-composition-set pure_fe                          $ create a composition set for pure fe
set-composition-property pure_fe a fe=1.00 cr=0.00 c=0.00

create-composition-set fe_cr                            $ create composition for substitutional grid
set-composition-property fe_cr a fe=1.00-cr_amount cr=cr_amount c=0

if (decarb==1)
   set-composition-property fe_cr o c=c_amount
endif

populate-monte-grid a fe_cr r                           $ populate substitutional grid

if (use_pure_boundary)
   add-site-selection u -1:0.0.0 -1:0.num_sites_yz-1.num_sites_yz-1 $ populate boundary with pure fe without traps
   populate-monte-grid * pure_fe r
   remove-site-selection a
endif

set-boundary-condition a p                              $ first set all boundary conditions to periodic
set-boundary-conditions l c                             $ left boundary is closed (needed for open boundary)
set-boundary-conditions r s                             $ right boundary is symmetric

setup-neighbor-shell 10                                 $ prepare neighbor shells in range of
                                                        $ 10 times the nearest neighbor distance
if (_MTNS<=1e9)
   create-neighbor-lists num_shells                     $ create neighbor lists for each site if
                                                     $ total number of sites is smaller than 1e9
endif

set-vacancy-position p 1:1.1.1                          $ set vacancy into grid
```

```
$ ********************************************************************************************
$ ************************************  OPEN BOUNDARY ****************************************
$ ********************************************************************************************

create-composition-set open_boundary       $ composition set for open boundary
set-composition-property open_boundary o c=c_amount
if (decarb==1)
    set-composition-property open_boundary o c=0.0
endif
create-open-boundary ob

set-open-boundary ob g o                   $ grid: octahedral interstitial
add-site-selection U 2:0.0.0.o 2:0.num_sites_yz-1.num_sites_yz-1.o   $ select all sites at left boundary in subgrid 0
add-site-selection U 3:0.0.0.o 3:0.num_sites_yz-1.num_sites_yz-1.o   $ select all sites at left boundary in subgrid 1
set-open-boundary ob s                     $ add selected sites to open boundary
remove-site-selection a

set-open-boundary ob v c                   $ valid elements: c
set-open-boundary ob c open_boundary       $ composition set: open_boundary
set-open-boundary ob t 2                   $ tolerance: 2

apply-open-boundary ob                     $ apply boundary composition to open boundary


$ ********************************************************************************************
$ ********************************** INTERACTION PARAMETERS **********************************
$ ********************************************************************************************

set-interaction-parameters t o                                $ choose pair interactions
set-interaction-parameters n num_shells
set-interaction-parameters p cr c c_cr_interaction
set-interaction-parameters p c c c_c_interaction

set-temperature-celsius 1000
set-vacancy-concentration c                    $ get the vacancy concentration from core
set-mobility-parameter i c                     $ get the diffusion coefficients from core

$ ********************************************************************************************
$ ********************************** OUTPUT WINDOWS ******************************************
$ ********************************************************************************************

new-gui-window g8                    $ generate window showing copper and vacancy in grid
set-gui-window-property . o a c      $ display c
set-gui-window-property . o o 0.24   $ zoom factor
set-gui-window-property . o s 5.0    $ site size
move-gui-window . 10 10 450 600      $ position window


$ ********************************************************************************************
$ ************************** START MONTE CARLO SIMULATION ************************************
$ ********************************************************************************************

create-monte-buffer _default_                          $ create buffer for simulation states
```

```
set-simulation-parameters u 1e2                          $ update every 100 steps
set-simulation-parameters s o 1 2                      $ store to buffer every 1e6 steps
set-simulation-parameters m sim_steps                        $ set simulation end after 1024 steps


$ ---------------------------- START MONTE CARLO SIMULATION ---------------------------------

$ let's go
start-monte-simulation

$ ********************************************************************************************
$ ***************************** MONTE CARLO SIMULATION FINISHED ******************************
$ ********************************************************************************************


set-variable-value evaluate 1           $ default value for interaction type

@send-console-string
@send-console-string
@send-console-string evaluate simulation?
@send-console-string   1 ... yes
@send-console-string   2 ... no
@input-variable-value evaluate "Enter selection: "

if (evaluate==1)
                    $ continue script, nothing to do here
elseif (evaluate==2)
   stop-run-script  $ nothing more to do
else
   send-dialog-string Invalid selection for simulation option. Select 1 or 2.
   send-dialog-string Script is interrupted.
   stop-run-script  $ stop script
endif

$ ********************************************************************************************
$ ********************************** SIMULATION EVALUATION ***********************************
$ ********************************************************************************************

create-global-table time
new-gui-window p1
set-variable-value diff_profile_id active_frame_id


new-gui-window g8                     $ generate window showing evaluation progress
set-gui-window-property . o w o       $ display what: only selected sites
set-gui-window-property . o o 0.24    $ zoom factor
set-gui-window-property . o s 5       $ site size
move-gui-window . 500 10 450 600      $ position window

set-variable-value corr_factor 0.78146
set-function-expression diff c_amount*erfc((x+1e-10)/sqrt(corr_factor/3*4*_MSIMNIS))

for (i;0..10:1)
   load-monte-buffer-state _default_ i
```

```
   add-table-entry time i _msimt
   format-variable-string index %d i
   create-global-table carbon#index
   for (j;0..(_MNUMX-1))
@     remove-site-selection a
@     add-site-selection u 2:j.0.0.o 2:j.num_sites_yz-1.num_sites_yz-1.o
@     add-site-selection u 3:j.0.0.o 3:j.num_sites_yz-1.num_sites_yz-1.o
@     add-table-entry carbon#index 2*j (_mtsn$c)/(_mtsn$c+_mnss)
@     remove-site-selection a
@     add-site-selection u 0:j.0.0.o 0:j.num_sites_yz-1.num_sites_yz-1.o
@     add-site-selection u 1:j.0.0.o 1:j.num_sites_yz-1.num_sites_yz-1.o
@     add-table-entry carbon#index 2*j+1 (_mtsn$c)/(_mtsn$c+_mnss)
   endfor
@  remove-site-selection A
   set-plot-option last_plot_id a y 1 s 0..2*c_amount
   set-plot-option last_plot_id a x 1 s 0..num_sites_x
   set-plot-option last_plot_id s n t carbon#index
   set-plot-option last_plot_id s n f diff auto
   if (i!=10)
      create-new-plot x diff_profile_id
   endif
endfor
```