THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Miha GUNDE

Le 26 novembre 2021

Développement de l'IRA : un algorithme de shape matching, sa mise en oeuvre et son utilité dans un kMC général hors réseau

---

Ecole doctorale : **GEETS - Génie Electrique Electronique,Télécommunications et Santé : du système au nanosystème**

Spécialité : **MicroNano Systèmes**

Unité de recherche :
LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes

Thèse dirigée par
Anne HEMERYCK et Layla MARTIN SAMOS

Jury

M. Alessandro LAIO, Rapporteur
M. Hannes JÓNSSON, Rapporteur
M. Normand MOUSSEAU, Examinateur
M. Victor MAGRON, Examinateur
M. Stefano DE GIRONCOLI, Examinateur
Mme Alexandra GORYAEVA, Examinatrice
Mme Anne HEMERYCK, Directrice de thèse
Mme Layla MARTIN-SAMOS, Co-directrice de thèse

Development of IRA: a shape matching algorithm, its implementation, and utility in a general off-lattice kMC kernel

Miha Gunde

December 2, 2021

Contents

Acknowledgements

During the PhD project, I have received an enormous amount of trust and patience, for which I am extremely grateful. It would have not been possible without the multi-level support from the supervisors Anne Hémeryck, and Layla Martin-Samos. They have taken very good care of me, let me explore (un-)related topics, and made certain things possible. I sincerely thank you for all of that. A large part of my gratitude goes also to Nicolas Salles, who introduced me to (however not only) modern programming concepts, and who built the general architecture of our kMC software.

I would also like to express gratitude to prof. Normand Mousseau, who received us in his group at Université de Montréal, and provided insightful information and discussions. Through this connection I have also been able to get to see the kART code up-close, which served as inspiration and continues to be the major influence, and the reference point for ideas developed and used in our project.

Additionally, I would like to thank all the members of the Jury, prof. Alessandro Laio, prof. Hannes Jónsson, prof. Normand Mousseau, prof. Stefano de Gironcoli, dr. Victor Magron, dr. Alexandra Goryaeva, dr. Nicolas Richard, and dr. Luca Grisanti.

I also thank the M3 team at LAAS, its past and current members, in particular Ruth Tichauer for the many lunchbreak discussions, Antoine Jay and Pierre-Louis Julliard for being the early-version users/testers of our kMC code, and Georges Landa for the responsive technical support.

As a member of the Multiscale And Multi-Model Approach for MaterialS In Applied Science (MAMMASMIAS) consortium, I acknowledge the efforts of the consortium in fostering scientific collaboration, and thank (in no specific order): Ruggero Lot, Gabriela Herrero Saboya, Franco Pellegrini, Matic Poberžnik, Claudio Zeni, and others.

Finally, I would like to thank my family for all the support and understanding. The same goes to my friends, Kelsey Calhoun, Rafael Torres, Adu Offei-Danso, Matija Pertot, Gašper Marinič, Katja Petelin, Adriana Rangelova, Rajat Sharma, and others.

Introduction

The reliable and accurate modelling of large-scale materials and their long-time evolution is still a frontier in research today. Since the early stages of computational modelling, several strategies have been developed to bridge the highly accurate but computationally expensive first-principles approaches, and more coarse-grained models. Some of the most commonly adopted approaches are, on one side, the semi-empirical potentials, that coarse-grain the quantum nature of the electrons into classical effective-charge potentials, and on the other side, the QM/MM techniques, that divide the system of simulation into two regions. A smaller region, treated by ab initio quantum mechanics (QM), and a larger region, treated by classical charge potentials within molecular mechanics (MM). In the fully atomistic description of a system, the first-principles approaches might address a few hundred atoms for a few pico seconds, while the classical potentials might be able to simulate a million atoms for few nano seconds. Beyond the atomistic description, coarse-grained models can be used to coarse-grain the system size, but not the simulated time. Crucial processes such as material degradation, oxidation, and growth occur on time scales that might be significantly larger than the nano second.

In order to enable the simulation of these processes, the question that needs to be answered is how to coarse-grain the time. In other words, how to filter the degrees of freedom of a system by their representative time scales, while maintaining a high degree of accuracy.

In the dynamics, the time scale of different degrees of freedom depends on the potential energy surface (PES) and the kinetic energy (temperature), thus the total energy of the system. The PES is a surface in a high-dimensional space, that links the potential energy to the geometry of an atomistic structure. A single point on the PES represents a unique structure, and its associated potential energy. As any surface, the PES contains critical points, i.e. points of a zero derivative. Such points are minima, maxima, and saddle points. Two minima can be connected by a line, such that the line passes through a saddle point. The dynamic evolution of a system is a trajectory on the PES.

When the kinetic energy is comparable to the relative heights of saddle points

of the PES, the system will be able to explore a large portion of the surface, and quickly reach a stationary equilibrium. When the kinetic energy of a system is significantly lower, the system will spend most of the time exploring a small region around a minimum point. In order to accelerate the exploration of large portions of the PES, several approaches have been developed, ranging from biased dynamics, such as metadynamics[1, 2], to kinetic Monte Carlo (kMC)-based algorithms[3, 4]. In industrial applications of materials science, the latter is the method of choice [5].

While dynamics-based approaches explore the PES by following some dynamical trajectory (biased or not), kMC-based approaches are, in principle, disentangled from the details of any trajectory. The kMC evolution occurs through single points of the PES, weighted by their occurence probability. As a consequence, kMC allows to coarse-grain both the length- and time-scales. However, a meaningful kMC simulation requires a meaningful coarse-graining of the PES. The main question is: given a certain material and process, which is the relevant level of abstraction?

One of the major and consequential assumptions in kMC is that the structure of a material and its long-term evolution can be coarse-grained by points on a rigid lattice. This assumption has been successfully implemented and applied on numerous occasions[6, 7, 8, 9, 10, 11, 12, 13, 14]. However, the lattice picture fails for some important materials and processes, such as amorphous materials, grain boundaries, interfaces, dislocations, nucleation, and surface growth problems. The main reason it fails is that the description of a structure imposed by the lattice is too simplistic for these kinds of problems, since subtle structural details might be driving the real evolution of the system. Few attempts of going beyond the lattice assumption have been made[3, 15, 16, 4, 17]. Contrary to lattice-based kMC approaches, the mapping between structures visited during the evolution and the real representation of the PES is the crucial feature of an off-lattice kMC algorithm. The major differences among off-lattice kMC implementations are therefore in the way this mapping is managed, i.e. how the structural information is treated, automatized, and reused.

The main effort of the present work has been to develop a workflow which allows a precise description of a structure and at the same time enables simple treatment, automatization, and reuse of structural information. The workflow allows us to link the specific points of the PES that are visited during a kMC simulation to atomic structures from a catalogue of structures. This link defines the possible move(s) to the next point(s) on the PES. The link between the two is the solution of the shape matching problem. Shape matching can be described as finding a rigid transformation between two structures, such that the two structures match as best as possible when overlaid one over the other. We have developed an algorithm called Iterative Rotations and Assignments (IRA), which accurately

and efficiently resolves the shape matching problem between two generic atomic structures, including structures with different number of atoms. The solution is given in the form of a rotation matrix, a translation vector, and a permutation matrix. After two structures are matched in this way, it is simple to evaluate the distortion between them.

The developed workflow containing the IRA algorithm has been implemented into an in-house general off-lattice kMC kernel. This workflow allows for efficient and fully automatic reuse of structural information. Due to the nature of the shape matching problem and our IRA algorithm that solves it, structural symmetries can be recognized and explicitly included with each structure. The performances and potential applications are showcased by examples ranging from non-trivial diffusion processes, to off-lattice surface problems, such as oxidation. The algorithm developed can also be used independently of the kMC setting, in any application that works with atomic structures, or in combination with a catalogue or database of structures.

Chapter 1 gives a more detailed introduction to the kMC algorithm, and the description of different assumptions and approximations that can be made. Chapter 2 introduces the concept of shape, and the considerations that need to be made when representing and expressing similarities between them, in particular when working with atomic structures. Chapter 3 describes the shape matching problem, and its application to atomic structures. It also defines our IRA algorithm, and describes all its components in detail. The results of performance tests are included. Chapter 4 describes the working principle of our in-house kMC, and defines the workflow used to link the kMC evolution of a system with a catalogue of atomic structures, with the help of the IRA algorithm. It also details the procedure of finding the explicit symmetries of structures involved. Chapter 5 gives some examples of kMC simulations, using the developed workflow. We showcase the different capabilities, as well as address the difficulties and potential problems we encountered. Some future directions for the work started in this thesis are given in Chapter 6, including potential application ideas outside of the kMC setting.

Chapter 1

Kinetic Monte Carlo

The dynamical evolution of a system is represented by its equations of motion, which can be integrated by Molecular Dynamics (MD) techniques. The basic idea of MD is to discretize the equations of motion via discretizing the time into small time-steps, for each time-step compute the instantaneous force on all atoms, and move the atoms according to that force. Repeating this for many time-steps generates a trajectory which represents a realization of the equations of motion. The problem is however that the quality of the resulting trajectory depends on the size of the time-step, a too large time-step can give unphysical trajectories, while a too small time-step can result in an overwhelming amount of computation for an underwhelming result. However, in any case a trajectory involving long-time scale dynamics of a system, computed with MD would require an enormous amount of computation.

If we think about the trajectory of an MD simulation as exploration of a Potential Energy Surface (PES), such that each discrete step in that trajectory represents a point on the PES, an MD trajectory would typically look like a bunch of connected points (a line). In Fig. 1.1, a hypothetical MD trajectory is shown as a black line, and two basins on the PES are marked as A and B. The colors represent the height, or the value of the PES, where low energy values are represented by blue, and high values by red.

Typically, most of the MD simulation is spent exploring the area surrounding the minimum of the current basin, which in practical terms means the atoms are vibrating around their equilibrium positions. The extent to which an MD simulation moves around the PES mainly depends on two factors. One factor is the shape of the PES, namely the depth/height of its features in terms of potential energy, and the other factor is the simulated temperature T. The simulated temperature brings an energy proportional to kT into the system, where k is the Boltzmann constant, which means any feature of the PES with an energy lower or comparable to kT can be accessed relatively easily by the simulation. On the other hand,

any feature of energy difference much larger than kT is hardly accessible, and the probability to reach it becomes very low. If two basins are separated by an energy barrier much higher than kT, then the transition between these two basins happens only rarely, and it is called a rare event. The systems which exhibit this kind of features on their PES are called rare-event systems.

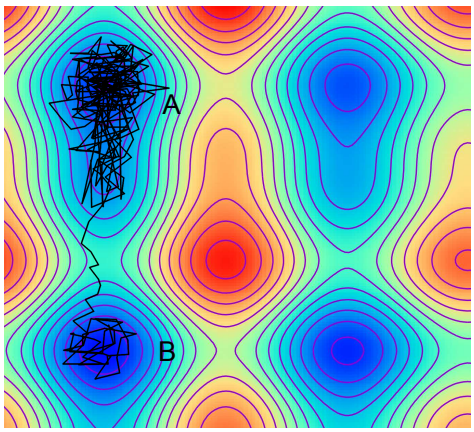


Figure 1.1: Abstract representation of a hypothetical MD trajectory, as set of connected points (black line), on a PES. The blue regions of the PES represent low energy values (basins), and red regions represent high values. The jump from basin A to basin B is called a rare event if the basins are separated by an energy barrier much higher than kT.

The long-time evolution of a rare-event system consists of jumps from state to state. We define the basin A as a partition of the whole $\mathbb{R}^{3N}$ configurational space, such that any configuration q is said to be in basin A if upon structural relaxation, it converges to the minimum of basin A. The state A then represents the union of all such configurations q. A rare-event system will stay within some basin for a long time relative to its vibrational motion within the same basin. By ignoring all movements within the same basin, the state-to-state evolution of a system can be written as equation:

$$\frac{\mathrm{d}P_A}{\mathrm{d}t} = \sum_B w_{BA} P_B(t) - \sum_B w_{AB} P_A(t), \tag{1.1}$$

where $P_A(t)$ and $P_B(t)$ represent the probability to find the system in the state A or B respectively, at time t, and w_{AB} represents the rate of transition from state A to state B, and similarly for w_{BA}. The Eq. (1.1) is called the master equation. In fact, a rare-event system will stay within the same basin long enough for the history of how it got there to become irrelevant. Exiting a state A has nothing to do with the history prior to A, so all transitions probabilities w_{AB} are independent of states

before state A. The system has no memory, the jumps are therefore independent and uncorrelated. Within this assumption, the system forms a Markov chain.

The rate w_{AB} of transitions from state A to state B is given by the Transition State Theory (TST) as:

$$w_{AB} = \nu_0 e^{-\frac{E_{ac}}{kT}}, \tag{1.2}$$

where E_{ac} is the energy barrier that is the difference between the energy of the initial minimum A, and the energy at the saddle point S_{AB} of the PES that is connecting the states A and B (see Fig. 1.2), and ν_0 is the vibrational frequency given as[18]:

$$\nu_0 = \frac{\prod_{i=1}^{3N} \nu_A}{\prod_{j=1}^{3N-1} \nu_{S_{AB}}}, \tag{1.3}$$

where ν_A are the vibrational frequancies at the minimum state A, and $\nu_{S_{AB}}$ are the vibrational frequencies at the saddle point S_{AB}. The frequency ν_0 is often approximated as constant, and called the vibrational prefactor, attempt frequency, attack frequency, or typical frequency. A jump from state A to state B on a hypothetical PES is represented on Fig. 1.2

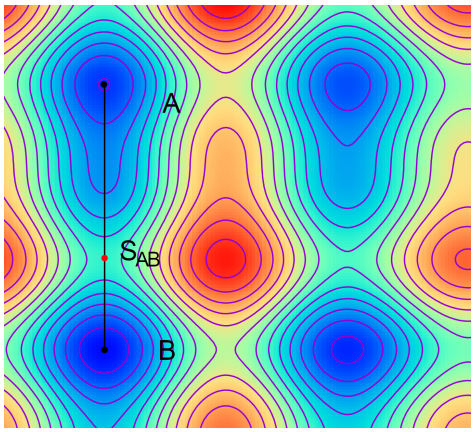


Figure 1.2: A jump from state A to state B, over the saddle point S_{AB} of the hypothetical PES.

Kinetic Monte Carlo (kMC) simulates the evolution of a system by performing state-to-state jumps, also called events, with associated rate in Eq. (1.2). The kMC algorithm provides a way to choose from the possible events, and generate a stochastic trajectory of events, which can be seen as a specific realization of the master equation in Eq. (1.1). The kMC algorithm also provides a way to

advance the clock, since the rate of an event can be related to a specific time, the overall simulation has an associated clock, see in the following Eq. (1.9). The transition rate w_{AB} has the units of frequency, and is often referred to as the transition probability.

1.1 The kMC algorithm

A generic kMC procedure can be separated into two aspects, see Alg. 1. One is related to the evolution of the system in terms of the statistics, namely how to choose the sequence of events such that the evolution is correct from the statistical point of view, and correspondingly, how to update the simulation clock in a correct way. The second aspect is related to the evolution of the physical structure, how precisely do the structural changes given by the kMC events follow the real physics, and how are the different events distinguished from each other based on their structure. In the following, the two aspects are labelled as "statistics", and "structure".

Algorithm 1 A generic kMC procedure. The IDENTIFY_POSSIBLE_EVENTS() part is responsible for the identification of all events that are possible to execute at the current simulation step, the CHOOSE_EVENT() part is responsible for the choice of an event based on its probability, the APPLY_EVENT() is responsible for the correct application of the structural change, given by the chosen event, and the UPDATE_SYSTEM() part is responsible for updating the simulation clock correspondingly.

```
1: procedure KMC( )
2:     while continue_simulation do
3:         IDENTIFY_POSSIBLE_EVENTS()          ▷ "structure"
4:         CHOOSE_EVENT()                      ▷ "statistics"
5:         APPLY_EVENT()                       ▷ "structure"
6:         UPDATE_SYSTEM()                     ▷ "statistics"
7:     end while
8: end procedure
```

1.1.1 Statistical aspect

As the events considered in a kMC simulation are assumed to be independent, uncorrelated rare events, they follow the Poisson distribution. In the Poisson dis-

tribution, the probability for n equivalent events to occur in time t is given by:

$$P(n) = \frac{(\gamma t)^n e^{-\gamma t}}{n!}, \tag{1.4}$$

where γ is the event rate. The probability $P(1)$ for one event to occur is then:

$$P(n = 1) = \gamma t e^{-\gamma t}. \tag{1.5}$$

Meaning that the expectation value $\langle t_1 \rangle$ of the time for one event to occur can be calculated by integrating:

$$\langle t_1 \rangle = \int_0^\infty \gamma t e^{-\gamma t} \mathrm{d}t = \frac{1}{\gamma}. \tag{1.6}$$

Since the events in a kMC simulation can have nonequivalent rates, the rate at which *anything* happens in kMC is given by the sum of all the rates w_i of events possible at the current simulation step i, we write total rate γ_{tot} as:

$$\gamma_{tot} = \sum_i w_i, \tag{1.7}$$

which can then be inserted into the Poisson disribution. The same is also true for each event separately, the probabilty distribution for a single event with rate w_i, to occur in time t is given by:

$$p(t) = w_i e^{-w_i t}. \tag{1.8}$$

In order to correctly choose events according to their rates, and update the simulation clock, there are two main kMC procedures. One is the Gillespie algorithm [19], and the other is the BKL algorithm [20]. Presently, both of these algorithms are standard kMC algorithms, the use of one or the other seems to be essentially community-related. While other algorithms exist, such as the constant-time algorithm [21], they will not be further discussed here.

Gillespie algorithm

The Gillespie algorithm [19] was first developed in 1976 to model the biochemical networks, such that the probability of possible reactions is respected. It is sometimes also called dynamic Monte Carlo, or stochastic simulation algorithm (SSA). The algorithm is as follows.

For each possible event rate w_i, draw a random number according to the distribution in Eq. (1.8), and call it t_i. Since only one event can happen per simulation step, choose the fastest one, meaning the event with smallest t_i. In order to draw

random numbers following the distribution in Eq. (1.8), we can draw a random number Z from a flat distribution [0,1], and compute:

$$t_i = -\frac{1}{w_i} \ln Z. \tag{1.9}$$

This time t_i is then also used to advance the simulation clock.

BKL algorithm

The BKL algorithm was proposed in 1974, by Bortz, Kalos, and Lebowitz, [20] for the study of Ising spin systems, in particular for cases where a system is close to an equilibrium, or some metastable state. The BKL algorithm is also called the n-fold way, or residence-time algorithm. The algorithm is as follows.

Draw a single random number Z_1 from a flat distribution [0,γ_{tot}], and then choose the event j such that Z_1 is between the partial sum of rates up to j, and partial sum of rates up to $j+1$. To compute the time t_j of this event j we draw another random number Z_2 from a flat distribution [0,1], and compute

$$t_j = -\frac{1}{w_j} \ln Z_2. \tag{1.10}$$

The time t_j is then used to advance the simulation clock.

The procedure CHOOSE_EVENT() from Alg. 1 can then be written based on which of the two algorithms is used for the choice of an event, as Alg. 2. The main purpose of the procedure UPDATE_SYSTEM() is to update the simulation clock with the time of the chosen event t_j. The reason for keeping the two procedures separated, is to leave some margin for any possible event denials during the APPLY_EVENT() procedure.

Algorithm 2 Choice of an event, based on the Gillespie, and BKL algorithms.

```
1:  procedure CHOOSE_EVENT()
2:      if Gillespie then
3:          compute random time t_i for all events i          ▷ Eq. (1.9)
4:          choose fastest event j and time t_j
5:      else if BKL then
6:          draw Z_1 from [0, γ_tot]
7:          choose event j: Σ_i^j w_i ≤ Z_1 < Σ_i^{j+1} w_i
8:          compute time t_j                                  ▷ Eq. (1.10)
9:      end if
10: end procedure
```

For detailed proof of the kMC algorithm, see for instance Ref. [22]. It has also been shown to be a general algorithm to simulate adsorption/desorption events in Ref. [23]. The event rate of adsorption from gas is related to the partial pressure of the specific gas. The rate of adsorption r_i of specie i follows the Hertz-Knudsen formula:

$$r_i = \frac{\alpha_i A p_i}{\sqrt{2\pi m_i kT}}, \tag{1.11}$$

where α_i is the sticking coefficient that takes a value in the range [0,1], which expresses the fact that only a fraction of incoming molecules will be adsorbed, A is the surface area of the atomic site of adsorption, p_i is the partial gas pressure, and m_i is the mass of the gas specie. The rate of adsorption r_i can be thought of as the vibrational frequency ν_0 in the TST expression for event rate in Eq. (1.2), thus the event rate for an adsorption event in a kMC simulation is expressed as

$$w_{ads} = r_i e^{-\frac{E_{ads}}{kT}}. \tag{1.12}$$

An adsorption event can be thought of as an event where an initially isolated gas molecule becomes attached to the surface with an adsorption energy E_{ads}, which can be represented by a gas molecule which arrives from a position far away from the surface. More simply, the initial state of an adsorption event is a clean surface, and the final state is the adsorbed molecule attached to the surface. An adsorption event thus changes the total number of atoms in the simulation.

kMC accelerations

It can be observed that the largest time resolution of kMC is on the time scale where no two events occur simultaneously. In a scenario where the simulation includes events with different orders of magnitude for the rates, the algorithm essentially makes many events of the fast rate, and almost none of the slower rates. Thus the time resolution is effectively small, given by the smallest event rate. In practice, such time resolution might lead to scenarios in which one single event (e.g. a simple atomic jump with a low energy barrier) is applied many times, before anything more interesting should occur, which hinders the overall performance of the software in terms of the desired timescales to be reached. This often undesirable feature is called flickering, or low-barrier problem. Methods of adjusting the kMC procedure in order to avoid this scenario can be made by absorbing Markov Chains [24]. This method came to be known as basin, or mean-rate method [25, 26]. It has been successfully applied to diffusion problem in Ref. [27], and in other studies, for instance Ref. [28]. The mean-rate method can be seen as filtering between slow and fast degrees of freedom, or small and big transition probabilities.

Modifications which effectively accelerate the execution of kMC are possible also at the software level, such as efficient parallelization, binary-tree search algorithms, etc. They will not be further discussed in this work.

1.1.2 Structural aspect

As can be seen from the statistical aspect of kMC described in Sec. 1.1.1, the general algorithms for the evolution of a system are relatively simple. However, the structural aspect of kMC is where the major distinctions are drawn between the available software, and systems of simulation. The reasons for this distinction lie in the assumptions, simplifications, and compromises made regarding the description of a system, in order to make a certain simulation possible. More precisely, the problem lies in the way atomic positions are treated, and the desired level of coarse-graining of the dynamics of the system.

For certain systems of simulation, an assumption (called the lattice assumption) can be made, which is that the atoms only occupy positions in a very small region around their reference position, given by the fixed crystal lattice. If this lattice assumption is true for all states that the system can possibly evolve through during its evolution, then a simplification of Lattice kMC (LkMC) can be done. LkMC constrains the atoms to only be allowed to move on a rigid lattice, which represents an idealization of the atomic positions. LkMC is further described in Sec. 1.2.

The structures involved in the reactions, or events in the kMC simulation, can also be coarse-grained. Meaning that a group of atoms representing a characteristic structure, can be thought of as a single object, with its own corresponding set of possible events. This is done by Object kMC (OkMC). Such grouping of the atoms, which can also be constrained by the lattice assumption, represents an idealization and abstraction of the kMC events. OkMC is further discussed in Sec. 1.3.

The idealizations of LkMC and OkMC bring simplicity into the structural aspect of kMC, and with it also the computational efficiency and speed. The assumptions of LkMC and OkMC can also be done for systems where they are not always exactly true, by forcing the system to evolve in a certain way. In those cases, an LkMC or OkMC simulation will be a tradeoff between the computational speed, and the precision in the description of a system in terms of the atomic positions.

When the lattice assumption is not applicable, off-lattice capabilities need to be enabled. The off-lattice kMC methods should make no approximations or idealizations of the structures, their "ground-truth" should be whatever is given by the events included in the simulation. In order to make this possible, and to grant portability to other systems, and a high degree of automatization, the procedures related to the structural aspect need to be general. The off-lattice kMC is further

discussed in Sec. 1.4.

Naming conventions

It often happens in the literature that a kMC code is written specifically for the material and problem being studied. The material, and the desired degree of coarse-graining of the simulation dictate the type of code that is needed: the assumptions, approximations, simplifications, and abstractions that are made. As a result, there are a large number of specific implementations of the kMC algorithm available, where the (dis-)advantages of each are not always clear, and their portability to other materials and problems is rarely questioned or discussed. As a possibly direct consequence of this, there appears to be a degree of freedom in the choice of the labels that are used to describe a kMC code.

Very often, the historically earlier works make the lattice assumption by default, but their approach is not specifically labelled LkMC. The lattice assumption is also often ommitted from the naming convention of the early definitions of Object kMC (OkMC). In order to differentiate from OkMC, the kMC variants that do not use object abstraction are sometimes called Atomistic kMC [29, 30, 31], or AkMC. The AkMC approaches also often omit the lattice assumptions in their naming. The adjectives "object" and "atomistic" appear to be related to the way the events are treated within the kMC, either as group, or single-atom moves.

Approaches which abandon the lattice assumption altogether are referred to as off-lattice kMC, which relates to the description of the system of simulation, but do not mention the specific treatment of the events (object vs. atomistic). But, off-lattice approaches traditionally include a self-learning on-the-fly approach to event exploration, which inherently includes a group of atoms. For this reason their events could probably be described as objects with atomic resolution, thus objects at the lowest level of abstraction, atomistic objects.

Approaches described as hybrid "atomistic-object kMC" have however been reported for example in Ref. [32], where certain events are treated as objects, and the others as atomistic. In Ref. [33] for example, which is also described as hybrid, the lattice assumption is abandoned, and different types of defects are grouped into different objects, such that the atoms near the border region of an object need to be in crystalline positions. The difference between their approach and an off-lattice kMC is that the latter typically uses a self-learning on-the-fly technique for exploring the events, while the former uses what the authors describe as "*prescribed set of correlated atomic moves*."

In summary, the naming conventions appear to be related to different aspects of the code. These aspects appear to be:

- the way atomic structures are treated: lattice, or off-lattice;

- the way events are treated: events including a single atom, events including a group of atoms, or events including abstract objects;
- the way event exploration is treated: self-learning on-the-fly approach, or events given pre-described in a catalogue (or hard-coded).

1.2 Lattice kMC

A well-grounded variation of kMC simulation is to impose a fixed grid onto the system of simulation, such that the atoms can only move on the grid points. The grid is often taken as the lattice of the underlying crystal, however other choices can be made, including intertwining of several lattices, or other options. Such kMC variation is in general labelled Lattice kMC (LkMC). Studies with LkMC methods typically target systems that do not undergo drastic changes in the morphology of the structures during certain processes, such as for example diffusion [6, 34], or certain simplified surface phenomena [35, 36, 37, 38] such as epitaxial growth [7, 39, 14], oxidation [8, 11], step morphology [40], or catalysis [13, 41].

By imposing the atoms to be on a fixed grid of points, there are a number of simplifications that can be done. Most importantly, compiling the full catalog of all possible events essentially becomes a problem in combinatorics in the number of possible initial and final states, given the possible grid sites. Secondly, each kMC move on a fixed grid is inherently a very precise transformation of an environment, in the sense that atoms only move on the grid points. Certain kMC software idealizes the events as-if they were on a fixed grid, however allows for relaxation of the forces after every move. Thirdly, the environments in the simulation can be efficiently characterized using rather simple descriptors. For example, whether a grid point contains an atom or not (or any user-defined description, as in Ref. [10]), all other environmental variables can easily be computed and stored in arrays like the neighbor list. An illustration is given in Fig. 1.3, where atoms are shown on the points of a grid, which is represented in grey. An atom of a different type is shown in yellow (which could possibly also represent an unoccupied lattice site), and some possible diffusion events associated to this site are shown with blue arrows. Typically, only one atom is assumed to move in an event.

Alternatively, in the software developed in Refs. [9, 42], on-lattice local atomic environments are encoded as simple graphs. Due to prescribed atomic positions on the lattice points, the space of possible states is discrete, and thus the simple graph description is unambiguous. Within the lattice philosophy, one can also interpolate certain information, for example the energy barrier as function of the number of first neighbours, as is done in the commercial Lattice kMC code reported in Ref. [43]. In some cases, LkMC can be parametrized in such a way

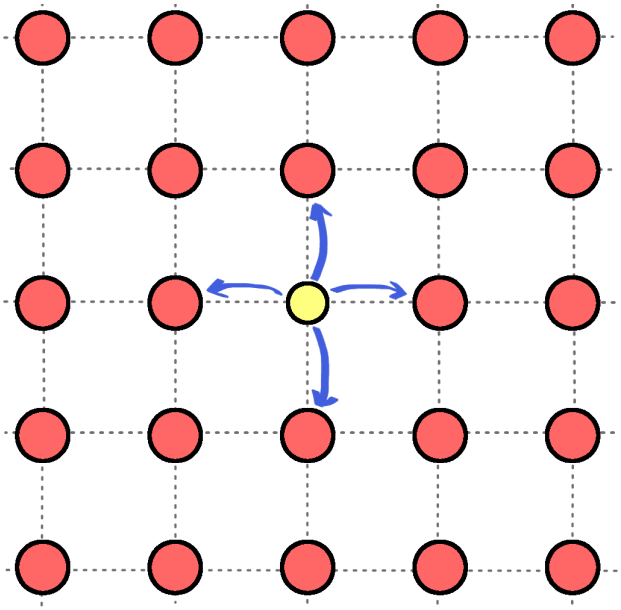


Figure 1.3: Illustration of LkMC situation. The fixed grid is represented in grey, an atom of different type is colored in yellow, and possible associated diffusion events are shown with blue arrows.

to effectively capture off-lattice effects [44]. LkMC has also been used to study certain bio-assemblies [45]. The performance analysis of LkMC in presence of external fields is reported in Ref. [46]. One of the most notable software employing LkMC is the SPPARKS code [12].

1.3 Object kMC

Another variation is to group atoms into objects, like for example adsorbate islands on a surface, or group of defects that are close together. Such kMC variation is generally labelled Object kMC (OkMC). An object is thought of as a single entity, and typically the events ascribed to such object can transform the object internally (including resizing), displace the object as a rigid structure (collective diffusion, dislocation diffusion), or split the object into two or more smaller objects. The atomic configuration in OkMC is generally encoded into a lattice of objects, as illustrated in Fig. 1.4, where different objects are represented as colored shapes, and the lattice in grey. In this case, an event would be represented by the transformation of one object into a different object, or a displacement of an object to a different lattice point.

If one thinks of the kMC method as coarse-graining of MD in time, then one

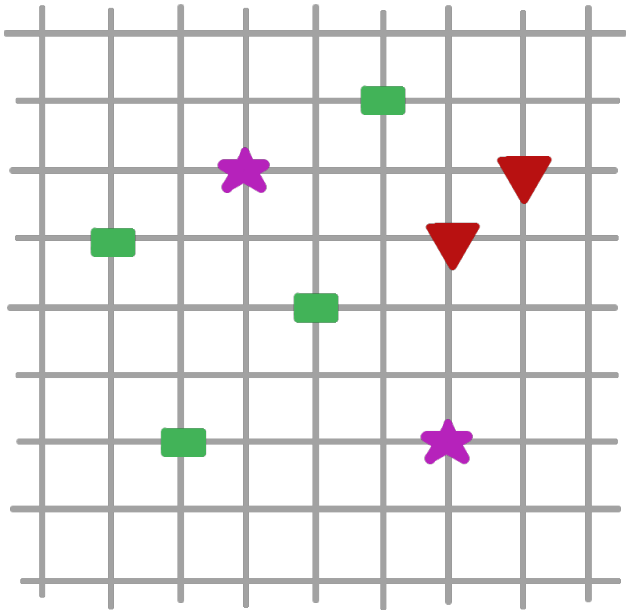


Figure 1.4: Illustration of lattice of objects in an OkMC simulation. The different objects are represented as colored figures, and the underlying lattice in grey.

can think of OkMC as coarse-graining of kMC in space, or in the type and number of structures or chemical species present. With abstraction through objects, the kMC method is greatly accelerated, capable of achieving long time-scales, however at the cost of losing the atomic scale resolution. The most notable use of OkMC is in the simulation of clustering of defects in materials, [47, 48, 49], in such case a cluster of defects is represented by objects.

Keeping in mind the object abstraction, one could ask questions regarding the rules of the mapping from atomic positions to objects. Such as, what is the limit for the number of atoms that can be grouped into an object, or what kind of considerations are made regarding the overall object size and structure, or quite importantly, is the mapping in the other direction of any meaning? That is, can the objects be mapped back to atomic structures, and if yes, are the object abstractions portable from one system to another? Or do we need to define them all from scratch when we change the system of study? Not all of these questions have a straightforward general answer. For the subset of materials where these questions have a favourable answer, the OkMC method is a powerful simulation tool, capable of reaching extremely long time-scales.

1.4 Off-lattice

At least four important problems need to be faced with, when designing a kMC software with off-lattice capabilities:

1. the system can evolve towards states that are unknown, that is states that are not present in the event catalogue;
2. need an unambiguous way of structural comparison for asserting the equivalence of two states;
3. the states need to be labeled in an unabiguous way in the event catalog;
4. need a way to find the correct reference frame in which an event selected from the event catalogue can occur.

One of the most important assumptions of a kMC algorithm is the complete knowledge of all possible events at every simulation step. This assumption can however be far from true, as we can never be sure that the events included in a simulation represent all possible physics. The assumption is important for two reasons. The first one is for an accurate computation of the total rate γ_{tot}, and subsequently for the computation of event times. The second, more important reason is that, in an extreme case when the event catalogue does not include any possible transitions from the current state, the simulation cannot continue. This is the scenario described by problem 1. To overcome it, a good off-lattice kMC method should be able to implement what is called a self-learning technique. This means that the kMC is able to perform an exploration of possible transition paths from the current state, and is thus able to find relevant transitions from a previously unknown state, and at the same time increase the probability of finding and including non-intuitive events in the catalog, that might have previously been missing.

The last three problems from the list are all in some way related to the reusability of the events in the event catalogue. Problem 2 occurs when the system is checked for possible events to occur at the current simulation step. Since only the subset of the events in the catalogue, whose initial state is present in the simulation, are possible to occur, an unambiguous way of structural comparison is needed, which does not rely on any underlying grid. Problem 3 occurs when searching the event catalogue for a possible event. If two (or more) non-equivalent events are labelled in the same way, this poses a problem since the choice of a wrong event can result in a seriously wrong transition. Problem 4 occurs at the stage of event execution, since we need to be sure that the event is applied within a proper reference frame, and in the proper direction. If any of these problems are not resolved to a sufficient degree, the simulation can go very wrong. Trivially, one could get rid of them by doing a full exploration of events from scratch at

every atomic site, for every step of the simulation, which however defeats the idea of re-usability of the event catalogue.

A number of implementations of this particular kMC variant, employing differing methods of explorations of transition states, differing structural comparison methods, and differing approaches to the reusability of events exist, some of them are briefly listed below.

One of the most notable off-lattice self-learning kMC software is the kinetic Activation-Relaxation Technique (kART) [4]. In (over-)simplified words, kART launches an exploration of possible transition paths from the current state each time it falls into a state that it does not yet know. All the found transitions and their rates get stored in the event catalogue for possible reuse later in the simulation. In order to enable the explorations, kART needs a way to compute the forces, which it does by using empirical potentials. As a solution to the structural comparison problem, kART relies on the isomorphism of simple graphs. The atoms within a radial cutoff region are taken as the local environment for which a graph is constructed, if two graphs are isomorphic then it is assumed that the local environments are equivalent. This assumption can be ambiguous, which kART can identify and deal with by enlarging or lowering the radial cutoff, until two graphs differ. It uses the same graph mechanism to label the events in the event catalog. In order to correctly execute the chosen event, kART follows what is called the canonical labeling of a graph, which gives a certain permutation of the graph nodes, in which the graphs are identical. This canonical labeling effectively gives the order of atoms which map the event to the local environment in the system, and to confirm it, kART relies on the computation of forces. In the case of rotation between the event configuration and the local environment in the system, kART uses a trial-and-error approach, based on calculation of forces of attempted moves, to map atoms corretly. It thus resolves all the four problems from the list to a sufficient degree, however at the cost of having to do many force calculations.

The adaptive kMC (akMC) [3, 50] method, uses the dimer technique [51] to explore the transition paths, including all atoms in the simulation. Alternatively, accelerated high-temperature MD simulations have also been used as transition state exploration method within akMC [52]. The akMC method is presently implemented in the EON software package [53].

Self-learning kMC (SLKMC) [15, 54] method uses the drag method as transition state exploration method, and a pattern recognition scheme for the reusability of the found events.

Self-evolving atomistic kinetic Monte Carlo (SEAK-MC) method [16, 55] introduces the concept of active volumes (AVs), such that the system evolution is done within these AVs. By monitoring the properties of boundary atoms, such as deviations in energy, strain, or stress, of each AV, possible inter-AV reactions

are taken into account through automatic AV boundary determination and redetermination. SEAK-MC primarily uses the dimer technique [51] to explore the transition paths localised to AVs, though other methods have been used [56].

The algorithm local-environment kinetic Monte Carlo (LE-KMC) [17] introduces a local environment descriptor, based on the local geometry, used to associate the possible events from the catalog to the system of simulation.

A notable approach to structural comparison in this context is the Kinetic DataBase, reported in Ref. [57].

In the following chapters we present the approach we have developed for an off-lattice kMC that combines the ideas of simple graph isomorphism, and an efficient geometric shape matching algorithm, to unambiguously resolve the three problems associated with re-usability of the event catalogue: the problem of structural comparison, the problem of labelling of the events in the catalog, and the problem of finding the correct reference frame for the execution of an event. All of this is done without the need of computing the forces. And since our approach does not rely on a pre-defined grid (lattice), it is portable to other systems and processes of simulation, including those with a changing number of atoms. Using the naming convention summarized in Sec. 1.1.2, the present state of our kMC can be labelled as: off-lattice kMC with a given catalogue of pre-described events, where each event includes a group of atoms.

Chapter 2

Shape representation and similarity

2.1 Introduction

Any geometrical pattern can be thought of as a *shape*. According to [58], there is no universal definition of what *shape* is. However the ability to recognize and compare shapes is of universal importance in everyday life, as well as in a more technical and scientific setting. The latter is our concern here. Recognition and comparison of shapes relies on the definition of a *meaningful* measure (degree) of similarity. Broadly speaking, there are two particular issues to be addressed.

Firstly, there are a number of things to consider regarding what constitutes a "shape", e.g. the overall geometry, its positioning and orientation in space, construction by specific constituents (parts), some particular properties related to its parts or whole, the order in which its parts are stored, etc. Therefore it should clearly be specified what one is talking about when talking about *shape*.

And secondly, the wanted level of resolution (precision) when comparing shapes might differ greatly for different measure definitions. In other words, the measure should be chosen according to how (in-)homogeneous the set of shapes to be compared is. This is a bit of a paradox since we would need to know the diversity in a set of shapes as a precondition to choosing a measure of similarity which would asses the diversity in that same set. Often we just choose a measure that is at hand. A critical situation is however when a chosen measure does not work well over the diversity of shapes in a set. A vivid example would be to compare a basket of oranges, would it suffice to simply say "they are all round", or would we need more precise words? And which measure do we use when there is also a potato in that basket, or a banana? Therefore, one should either choose the measure of similarity carefully and according to the diversity in the set compared, or precisely specify how to interpret possible mismatches and what to do in those cases.

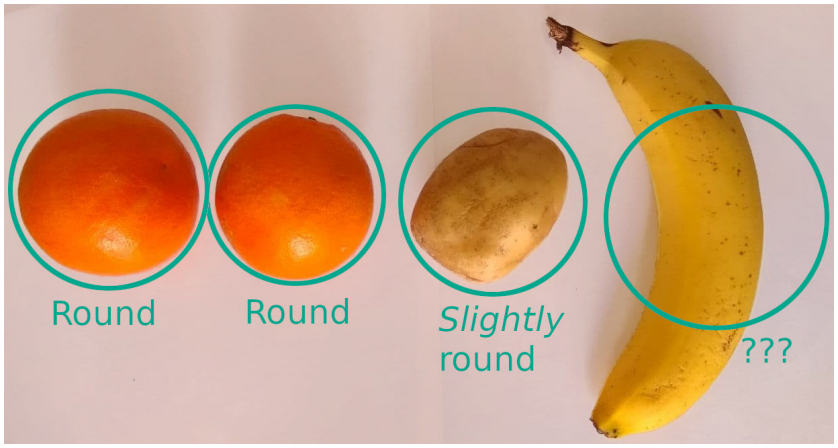


Figure 2.1: A set of oranges could be measured by how round they are. But what happens to the measure outcome when we include a potato, or a banana? Choosing a measure of similarity should be consistent with the diversity in the set of shapes. The interpretation of possible mismatches should be given.

Since the comparison of shapes by hand is a very tedious task, we would like to automatize the task via a computer. That means we need to formulate the problem of shape comparison into an algorithm. The type and implementation of any algorithm depends on its application. An application where a rough but extremely fast evaluation is needed, will use a different algorithm than some other application which needs a very precise evaluation with multiple checks and where time is not a particular constraint. Moreover since a shape can be any given geometrical pattern, algorithms and approaches will differ based on the form of the data given to an algorithm.

In some cases the important part of a shape is its surface, the interior does not matter. In that case, a shape can be given as a tessellation of smaller surfaces of certain shapes like triangles, in the form of their normal vectors for instance. However more interesting cases are where the data of a shape is given as a set of points, generally referred to as point sets. The number of possible applications of algorithms for shape similarity when data is given as set of points is very large, we will not discuss it further. Just as an example, data coming from imaging sensors is a 2D array of points. Applications such as image recognition immediately come to mind, for example fingerprint recognition [59]. Many imaging techniques can also provide data in 3D, for example LiDAR [60], with applications to landmark recognition, etc.

Atomic structures are generally given as point sets, where each atom is represented by a point in 3D space, which might have certain associated property(-ies): chemical specie, mass, charge, etc. There might also be properties relating spe-

cific atoms to each other, such as chemical bonds, networks of connectivity, etc. The most basic description of an atomic structure is a set of points, where each point has three coordinates whose values are prescribed by the reference frame in which the structure is written, and each point has a property of chemical specie. In the case of crystals, the ensemble of points in a structure might be part of a periodic lattice. In that case, the lattice is given by three non-collinear vectors that prescribe the pattern of periodicity.

The atomic structure data is usually generated as output of different computational approaches dealing with atomic structures. For example, structural optimization via force minimization (relaxation), or structural evolution via solving the equations of motion for a system of particles (molecular dynamics), or from a number of other (combined) approaches. Commonly, these approaches respect the invariance of physical laws with respect to the reference frame. This means that in absence of any externally imposed forces, the position and orientation of a relative reference frame of an atomic structure within the simulation box does not change the physics. This also means that there is no general standardized reference frame to perform simulations of atomic structures. This brings us to an important realization regarding the comparison of shapes within a set of atomic structures. Two identical structures that differ only by relative orientation, contain identical physics. Therefore, if physical properties relative only to that structure are of our interest, the orientation of its reference frame does not matter, and we can devise algorithms for shape similarity keeping this in mind. More about this scenario in Sec. 2.2. On the other hand, we can think of situations where the important information is not only the similarity of shapes, but also the relative orientation of one with respect to the other. It might be the case, for example, when several orientations of a structure appear during the same simulation, and we would like to quantify them in terms of their directions, or just simply map them onto a reference structure. Sec. 2.3 introduces some concepts and ideas related to that scenario, which is also the subject of the original work done during this PhD project, and is the topic of the subsequent Chapter 3.

2.2 Orientation is not important

When speaking about shapes in terms of atomic structures, often the relevant physical quantities vary only due to the overall geometry and associated chemical species. The issue of shape similarity can in that case be addressed by employing distance-like functions that evaluate the relationship (i.e. distance/similarity) between two abstract structural encodings, often called descriptors. These descriptors and the related distance functions can give meaningful comparisons of atomic structures in terms of their similarity in geometry and chemical compo-

sition, but not of the relation between their orientation/position in space (rotated state).

2.2.1 Representation

Descriptors are functions that encode structural information into high- or low-dimension vectors, in a way that is invariant to rotations, translations, and permutations of indistinguishable points. They can be classified by the size of their output. As for instance size 1 x 1 scalar, size 1 x N vector, and size M x N matrix. Graphs and their properties can also be utilised as descriptors.

Scalar descriptors:

Scalar descriptors can be for instance eigenvalues or functions of eigenvalues of certain tensors. For example the gyration tensor, defined as:

$$S = \frac{1}{N} \sum \begin{pmatrix} x_i^2 & x_i y_i & x_i z_i \\ y_i x_i & y_i^2 & y_i z_i \\ z_i x_i & z_i y_i & z_i^2 \end{pmatrix}$$

where the values of x, y, and z are coordinates of points from the structure, written relative to its geometrical center, and N is the total number of points in the structure. The gyration tensor S can be interpreted as "average spatial distribution" of points in a structure. Taking the eigenvalues of S in a descending order, $S_1 > S_2 > S_3$, a scalar called asphericity can be constructed as

$$b = S_1 - \frac{1}{2}(S_2 + S_3),$$

which returns value zero when the structure is spherical, or some Platonic solid (cube, tetrahedron, etc). The scalar b is a valid descriptor of an atomic structure. Other such scalars can be constructed, for example acylindricity,

$$c = S_2 - S_3,$$

which returns value zero when the distribution is cylindrical. Or another scalar called relative shape anisotropy,

$$\kappa^2 = \frac{3}{2} \frac{S_1^2 + S_2^2 + S_3^2}{(S_1 + S_2 + S_3)^2} - \frac{1}{2},$$

which returns a value in the interval [0,1], where 0 means distribution is spherically symmetric, and value 1 when all points are on a line. A tensor related to

the gyration tensor is the inertia tensor, which can be interpreted as "average distribution of mass". Its eigenvectors are often described as symmetry axes of a structure. Similar functions of its eigenvalues can be constructed, which can be exploited as structure descriptors.

Descriptors of this type have been used for example as collective coordinates in studies of metadynamics of peptides [61], and in analyses of shapes of polymer chains in attractive cages [62], where the authors exploited the gyration tensor, its eigenvalues and related shape descriptors mentioned above to construct a pseudophase diagram of the polymer-cage system.

Vector descriptors:

Descriptors can also be of vector type. Most commonly, the idea is to encode the structural information into a vector that in principle resembles the radial distribution function. In this way the descriptor vector is made invariant to rotation, translation, and permutation of identical atoms. The applications which generally use these kind of descriptors is in the Machine Learning (ML) approaches devoted to atomic structure simulations.

Among the many descriptors used and developed by the ML community, the most notable ones are the Behler-Parinello descriptor [63], the Smooth Overlap of Atomic Positions (SOAP) descriptor [64], and lately the Atomic Cluster Expansion (ACE) descriptor [65]. Some other descriptors of this type are also in use, for instance in Refs. [66, 67, 68, 69, 70]. The SOAP descriptor is briefly outlined in the following.

The idea of SOAP [64] is to write the atomic structure as a density function. To achieve this, atomic positions $\mathbf{x}_i$ are transformed into a sum of Gaussians, placed at the atomic positions, thus defining the "local density of atoms" $\rho_A(\mathbf{r})$, of structure A as:

$$\rho_A(\mathbf{r}) = \sum_{i \in A} \left(- \frac{(\mathbf{x}_i - \mathbf{r})^2}{2\sigma^2} \right).$$

This local density is invariant to atomic permutations, however it depends on the relative reference frame of the structure. To achieve rotational invariance, it is expanded in basis of spherical harmonics and orthogonal radial basis functions. The coefficients of this expansion are collected into a feature vector $\hat{\mathbf{p}}(A)$, which is a rotationally invariant vector, that carries encoded information of the atomic structure A.

Starting from the local density of atoms, another way to obtain a rotationally invariant quantity is to write a kernel function which takes into account all possible rotations given a symmetry group, see Ref. [71].

Matrix descriptors:

Descriptors giving a matrix output also exist, for example the Weyl matrix [72]. The Weyl matrix is a matrix of all pairwise scalar products between two structures. It is however not permutationally invariant.

Graph descriptors:

Another idea is to take the problem from a more general mathematical standpoint. That is, to map the problem into an abstract topological space, and think about it in terms of a graph.

The topological space is an abstract space with zero properties, only vertices connected by edges. An edge can represent any relationship between the vertices it connects. This structure is called a graph, denoted by $G = (V, E)$, where the graph G is an ordered pair of vertices V and edges E. V is the set of vertices also called nodes, and E is a set of edges also called links. If the set of edges E is ordered, the graph is called directed, and if set E is unordered, the graph is called undirected. The set of edges E can also have associated weights for each edge, in that case the graph is called a weighted graph. If an edge has endpoints that are equal, e.g. it connects a vertex to the same vertex, it is called a loop. If any two vertices are connected by more than one edge, the graph is said to contain multiple or parallel edges. A graph containing undirected, non-weigthed edges, no loops, and no parallel edges, is called a simple graph. Fig. 2.2 shows an example of a simple graph, and an example of a directed multigraph.

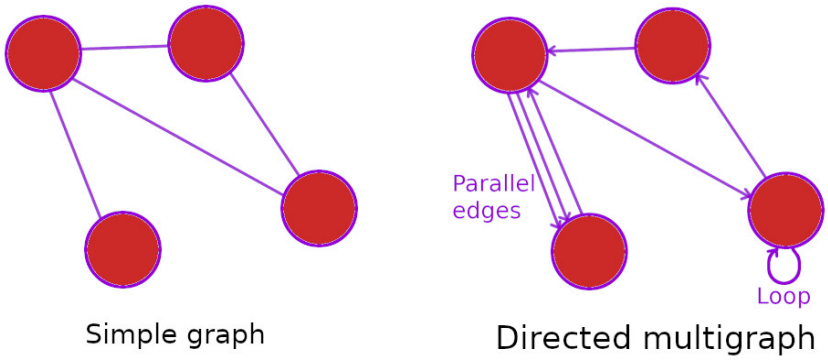


Figure 2.2: An example of a simple graph on the left, which contains non-weighted, undirected edges, no loops, and no parallel edges. On the right, an example of a directed multigraph, with non-weigthed, directed edges, one loop, and multiple (parallel) edges.

The relation between vertex labels and edges can be written in the form of an adjacency matrix, also called connectivity matrix. The connectivity matrix

contains elements c_{ij} which give the relation between vertices labelled i and j respectively.

In a simple graph, the connectivity matrix contains only elements 0 and 1.

$$c_{ij} = \begin{cases} 1, & \text{if } i,j \text{ connected} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The matrix is symmetric since all edges are undirected, $c_{ij} = c_{ji}$, which means an edge connects i to j and j to i. The diagonal values are all 0, since loops are not allowed, an element $c_{ii} = 1$ represents a loop on vertex i.

In a directed (multi)graph, the connectivity matrix is not symmetric, since the edges have a direction and an edge i to j does not automatically mean there is also an edge j to i. There can be values 1 on the diagonal, since loops are allowed.

The connectivity matrix of a weighted graph has elements $c_{ij} = w(i,j)$, where $w(i,j)$ is the weight associated to the relation between vertices i and j, and it can be any number.

Two graphs are said to be isomorphic when the set of vertices of one graph can be obtained by relabelling the set of vertices of the other graph. In other words, there exists a permutation of the vertices of one graph, such that its connectivity matrix equals the connectivity matrix of the other graph. If the connectivity matrix is a property of an abstract topological space W (graph), then any such space isomorphic to W must contain the same property. The connectivity matrix is a topological invariant in this sense.

If we map the points from an atomic structure into graph vertices, and relations between the points into graph edges, then graph isomorphism can be exploited for identifying equivalent shapes. A very simple approach is to map atomic vectors into vertices, and map the distance between atoms into an edge between those atoms, based on some distance cutoff. The connectivity matrix is then constructed from elements c_{ij}:

$$c_{ij} = \begin{cases} 1, & \text{if } d(i,j) \leq R_{cut} \\ 0, & \text{otherwise,} \end{cases} \tag{2.2}$$

where $d(i,j)$ is the Euclidean distance between atoms i and j, and the distance cutoff R_{cut} can depend on chemical species of atoms i and j. The connectivity matrix is by construction invariant to rotations of the atomic structures.

To calculate whether two graphs are isomorphic or not, the software NAUTY [73] can be exploited. Given an input in the form of connectivity matrix and the atomic types, NAUTY can (among other things) return a graph hash value, in a way that is invariant to permutations. In simple terms, the graph hash value can be thought of as a tag of that particular graph. If another graph produces the same graph hash value, the two graphs are isomorphic.

This straight-forward technique of mapping the atoms into graph vertices and distances between them into graph edges, and then looking at isomorphism between two graphs is successfully exploited in the off-lattice, self-learning kinetic Monte Carlo software kART [4], where graph isomorphism is utilised as a simple way of assessing similarity/equivalence of atomic structures.

A slightly more convoluted approach is developed in Ref. [74], where the authors present a so-called a universal fragment descriptor, for predicting properties of inorganic crystals within a ML scheme. In this study, the elements of the connectivity matrix are equal to 1 if atoms i and j share a Voronoi face, and the distance cutoff is based on covalent radii. The resulting graph is decomposed into subgraphs with paths of length l, which goes up to $l = 3$. Additional properties are given as a schema of reference properties. All the information is then concatenated and filtered for low-variance and high-correlation variables. The final feature vector then contains more than 2000 descriptors.

The graph connectivity matrix also possesses other properties, for example, we could look at its eigenvalue decomposition. This has been done by authors in Ref. [75]. They introduced topological coordinates called SPRINT, which combine the largest modulus eigenvalue and the corresponding eigenvector. The eigenvector is sorted from smallest to largest component, within sets of alike atoms. This sorting operation makes the coordinates invariant to permutations of identical atoms. The SPRINT coordinates were shown to be useful for identification of structures emerging from dynamic simulations, for example different phases of water ice in Ref. [76].

Different and more complex types of graphs can also be constructed, that can carry a range of properties. The field of chemical informatics is rich with studies developing and utilising approaches based on different types of graphs, and the related invariants. For example, the notion of a graph can be extended to a hypergraph [77], which brings additional capabilities and techniques. Another example where graphs are used in chemistry is the SMILES [78] approach, which encodes a molecular graph by a unique ASCII string.

2.2.2 Similarity

The similarity of structural encodings, or descriptors, can be evaluated by a distance function. A distance function, or a metric, is a mathematical function that gives a distance between two mathematical entities, e.g. points or shapes. In general, a distance function $d \to [0, \infty)$ is a real-valued, and non-negative function such that: $d(x, y) = 0$ implies equivalence of x and y; the function is symmetric $d(x, y) = d(y, x)$; and the triangle inequality holds $d(x, y) \leq d(x, z) + d(z, y)$. The similarity is then expressed through the notion of a *distance*, where similar entities are perceived as *close*, and non-similar ones are perceived as *far*. An-

other concept is that of a similarity measure, which can be thought of as simply inverse distance. Value zero of similarity $s(x, y) = 0$ then implies complete non-similarity.

The simplest example of a distance function is the Euclidean distance between points x and y: $d(x, y) = |x - y|$. For computing the distance between vectors, the Euclidean distance is $d(\mathbf{r}_i, \mathbf{r}_j) = ||\mathbf{r}_i - \mathbf{r}_j||$. An example of similarity function between two vectors is the scalar product $s(\mathbf{r}_i, \mathbf{r}_j) = \mathbf{r}_i \cdot \mathbf{r}_j$, which is zero when vectors are orthogonal. For computing distance between matrices or arrays, one could for example compute the sum of all pairwise distances from vectors of array A to array B, such as: $d(A, B) = \sum_i ||\mathbf{r}_i^A - \mathbf{r}_i^B||$. And as an example of similarity function between matrices or arrays A and B, compute the maximal scalar product of all pairwise scalar products, $s(A, B) = \max_i(\mathbf{r}_i^A \cdot \mathbf{r}_i^B)$. Many distance and similarity functions exist, see for example the book Dictionary of Distances [79].

In the case of SOAP descriptor mentioned earlier, the similarity measure between two feature vectors $\hat{\mathbf{p}}(A)$ and $\hat{\mathbf{p}}(B)$ is evaluated by a dot product between the two vectors.

2.3 Orientation is important

In some situations it is desired to compare two structures in terms of their orientation. In that case, the relevant quantity might be the angle between equivalent axes, or something similar. Generally speaking, two structures can be written in any possible reference frame, and in any rotated and/or translated state. The problem of comparing two rotated and translated structures therefore fundamentally depends on first writing them in a common reference frame, and defining a relative rigid transformation between them.

2.3.1 Rigid transformation

A rigid transformation is an isometric transformation, which means that it preserves distances of a vector space. Such transformations are rigid rotation and/or reflection, and rigid translation. In the Cartesian 3-dimensional space, a rigid rotation and/or reflection $\mathbf{R}$ can be written as an orthogonal 3×3 matrix, with $|det(\mathbf{R})| = 1$. More precisely, if $det(\mathbf{R}) = 1$ then $\mathbf{R}$ is a proper rigid rotation, and if $det(\mathbf{R}) = -1$ then $\mathbf{R}$ corresponds to a reflection. It also holds that $\mathbf{R}^T = \mathbf{R}^{-1}$. The application of a rigid transformation T to a vector a returns the vector a', and is written as:

$$T(a) = a' = \mathbf{R}a + \mathbf{t}, \tag{2.3}$$

and the inverse operation, keeping the same $\mathbf{R}$ and $\mathbf{t}$:

$$T^{-1}(a') = a = \mathbf{R}^T a' - \mathbf{R}^T \mathbf{t}. \tag{2.4}$$

The permutation P of indistinguishable atoms within an atomic structure can also be seen as a rigid transformation.

2.3.2 Atomic structure

Representation

An atomic structure A is represented by a set of vectors $a_i \in A$, representing the atomic positions written in the respective reference frame, and the corresponding atomic properties. An atomic structure A thus contains $3N$ coordinates, where N is the total number of atoms in A, plus the atomic properties. When a structure is periodic, the lattice vectors are also given, representing the unit of periodicity.

Order of atoms

In some cases the order in which atoms are written in a structure is important, and follows some pre-described recipe. Such as for example in Protein DataBank (PDB) files, where atoms are grouped together based on which residue of the protein they belong to, for example amino acid.

However more importantly, for a meaningful comparison of atomic structures A and B on an atom-by-atom basis, the order of atoms in a set is important since we need to compare an atom from one structure to an equivalent atom in the other structure. The specific order of atoms is given by a permutation P, which is generally an $N \times N$-dimensional matrix of integer elements 0 or 1, where N is the number of atoms in the structures compared. The comparison of atoms is usually restricted such that an atom $a \in A$ is only compared to one atom $b \in B$. In this case, the matrix P is such that each row i or column j contains exactly one element p_{ij} with value 1, which can be written as: $\sum_i p_{ij} = 1,\ \forall j$ and $\sum_j p_{ij} = 1,\ \forall i$.

In general terms, the order of points in two sets of points being compared is usually called the *assignment*. The problem of finding the correct assignment is called the *Linear Assignment Problem* (LAP). Algorithms for solving the LAP are known, and they generally work by operating on a given cost matrix. The elements c_{ij} of the cost matrix represent a relationship between a point i from the first set, and a point j from the second set. The assignment of points $i \rightarrow j$ is then found by minimizing or maximizing a given function of the cost matrix elements. For instance as formulated by Ref. [80], the assignment matrix x is found by:

$$\min \sum_{ij} c_{ij} x_{ij}, \tag{2.5}$$

where the elements of the matrix x are subject to $\sum_i x_{ij} = 1,\ \forall j$ and $\sum_j x_{ij} = 1,\ \forall i$. The most widely known general-purpose LAP algorithm is the Hungarian algorithm [81, 82].

When applied to atomic structures, the assignment matrix x has integer elements, and represents the permutation of atoms P. The Hungarian algorithm can be applied to solve the LAP in the context of atomic structures, as for example in Refs. [83, 84, 85].

Similarity

The similarity of atomic structures needs to be evaluated with taking into account the general orientation. For this reason, the similarity/distance function needs to be variant under rigid transformations, i.e. it needs to compare components that depend on the positions relative to the reference frame of a structure.

Such set-set distance function is for example the **Root-Mean-Square-Distance** ($RMSD$), between structures A and B, expressed as

$$RMSD(A,B) = \sqrt{\frac{1}{N}\sum_{i}^{N} d(a_i,b_i)^2}, \tag{2.6}$$

where N is the number of points, and $d(a_i,b_i)$ denotes an Euclidean distance between points $a_i \in A$ and $b_i \in B$. The function $RMSD(A,B)$ is variant on rigid transformation of either sets A or B, since a rotation expressed as a matrix $\mathbf{R}$, and translation by a vector $\mathbf{t}$, acting on set B would return

$$RMSD(A,B) = \sqrt{\frac{1}{N}\sum_{i}^{N} ||a_i - \mathbf{R}b_i - \mathbf{t}||^2}, \tag{2.7}$$

where $||..||$ denotes an Euclidean norm. It can also immediately be noted that the function $RMSD(A,B)$ depends on the order of points i in both structures. In other words, the distance $RMSD(A,B)$ is variant on permutations P of structures A and B. In Refs. [86, 87], authors have proposed a re-definition of $RMSD$ based on shortest distances, in order to make it invariant on permutations. It can be expressed as:

$$RMSD_{inv}(A,B) = \sqrt{\frac{1}{N}\sum_{i} \min_{j} d(a_i,b_j)^2}. \tag{2.8}$$

Another set-set distance function that is variant over rigid transformations is the **Hausdorff distance** $d_H(A,B)$ [88]. It is expressed as

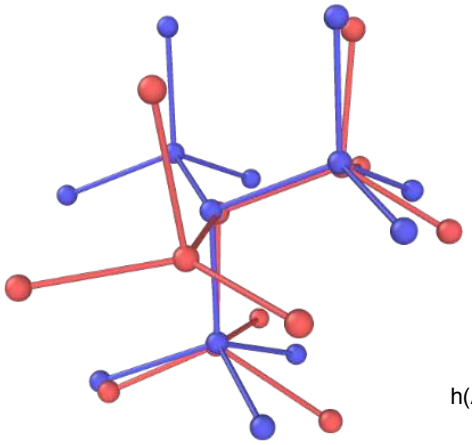
$$d_H(A,B) = \max(h(A,B), h(B,A)), \tag{2.9}$$

where $h(A, B)$ is

$$h(A, B) = \max_i \min_j d(a_i, b_j), \tag{2.10}$$

and $d(a_i, b_j)$ is the Euclidean distance between points $a_i \in A$ and $b_j \in B$. The Hausdorff distance is invariant over permutations by construction. Similarly as Eq. (2.8), the Eq. (2.10) takes the minimum distance over all atoms $b_j \in B$ for each atom $a_i \in A$. The difference is however that Eq. (2.8) makes a sum of all such minimal distances, while Eq. (2.10) takes only the maximal value. This subtility can be meaningful in some cases (see the last paragraph in Sec. 3.8.3). The definition of $d_H(A, B)$ in Eq. (2.9) takes the maximal value between values $h(A, B)$ and $h(B, A)$, which can differ when the sets A and B contain a different number of points. The distance h from Eq. (2.10) is thus not always commutative.

In Fig. 2.3, the calculation of the Hausdorff distance is shown for two atomic structures A and B. The atomic structures are similar, however slightly distorted, and rotated with respect to each other. The atomic indices are also permuted.



idx i	$\min_j$ d(i,j)	idx j
1	0.173	1
2	0.164	3
3	0.177	2
4	4.785	4
5	1.429	6
6	1.265	7
7	0.401	5
8	0.357	12
9	1.633	11
10	1.598	8
11	5.363	10
12	5.419	13
13	7.683	9

$h(A,B) = \max_i (\min_j d(i,j))$
$= 7.683$

Figure 2.3: Calculation of $h(A, B)$ for two atomic structures A in red, and B in blue. The structures are similar, but rotated relative to each other, and the atoms are permuted. The indices i are listed, with the minimal distances $\min_j d(a_i, b_j)$, and the corresponding indices j. The final value $h(A, B)$ is the maximal value among all $\min_j d(a_i, b_j)$.

The value of $h(A, B)$ in Eq. (2.10) is the largest value among the smallest distances from points $a \in A$ to points $b \in B$. The value of $d_H(A, B)$ in Eq. (2.9) takes the maximal value of $h(A, B)$ and $h(B, A)$. In other words, there exists a number ϵ such that each point in B is within ϵ-distance from a point in A, and each point in A is within ϵ-distance from a point in B; the closed ϵ-neighbourhood of A completely contains B, and the closed ϵ-neighbourhood of B completely contains A. The Hausdorff distance $d_H(A, B)$ is equal to the smallest such ϵ.

This is sketched in Fig. 2.4, where the value $h(A, B)$ is shown by a green line, and the ϵ-neighbourhood in shade of grey around the atoms of A in red. For the structures in this sketch, the values $h(A, B)$ and $h(B, A)$ are equal, thus the value $d_H(A, B) = h(A, B) = h(B, A)$. The sketch of $h(B, A)$ with the same structures would be identical, only the grey areas would be drawn around the structure B in blue. For an example when $h(A, B) \neq h(B, A)$, see Fig. 3.6 and the discussion

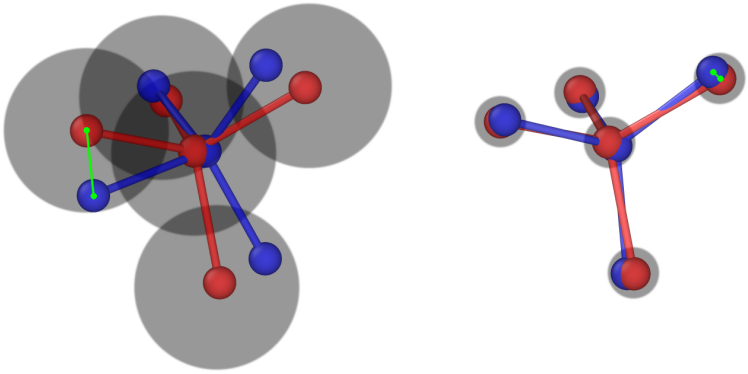


Figure 2.4: The value of distance $h(A, B)$ represented by the green line for two atomic structures A in red, and B in blue. The ϵ-neighbourhood around red atoms is represented by the area shaded in grey.

in that section around the commutativity.

The Hausdorff distance function is exploited in the original work done in this PhD project, it is further discussed in Sec. 3.4.

2.3.3 Reference frames

Representation

The coordinates of an atomic structure are expressed relative to some reference frame, which has an associated set of basis vectors, called the basis set, or simply a basis. A set of vectors β is called a basis of a vector space S if every element of the space S can be expressed as a linear combination of the vectors in β. In the case of Cartesian 3-dimensional space, the basis set β is formed by three orthonormal vectors $\hat{\mathbf{e}}_1$, $\hat{\mathbf{e}}_2$, and $\hat{\mathbf{e}}_3$, written as a 3×3 column matrix. The basis vectors can however also not be orthonormal, for example vectors expressing a primitive cell of a crystal are in general not orthonormal. In this thesis we assume Cartesian reference frames, unless indicated otherwise.

There exist some ways of finding a reference frame internal to a structure. For example finding the principal axes of inertia, or of the gyration tensor. The

problem with these two is that when a structure is isotropic/spherically symmetric, they are ambiguous and ill-defined. Another idea of a reference frame internal to a structure is the Eckart frame [89, 90], where the idea is that the reference frame is molecule-fixed, meaning that it rotates and translates together with the molecule, and the atomic movements expressed in Eckart frame are strictly due to atomic vibrations.

Similarity

Assume two orthonormal basis sets β and γ, which can be seen as two rotation matrices. In 3D, the rotations are the $SO(3)$ group, which has an intrinsic notion of the distance, and is expressed as an angle between the two rotations. The relative rotation between β and γ is represented by a rotation matrix $\mathbf{R}$, such that:

$$\mathbf{R} = \gamma^T \beta, \tag{2.11}$$

and the distance between β and γ is represented by the angle ϕ,

$$\phi = \arccos \frac{\mathrm{Tr}\mathbf{R} - 1}{2}. \tag{2.12}$$

Rotations in 3D can also be expressed as Euler angles or, in a more compact way, as quaternions [91, 92, 93]. A quaternion is a 4-dimensional vector giving the angle and the unit axis of the rotation. Quaternion algebra is well-defined and often convenient when dealing with rotations, we will however not discuss it further.

Chapter 3

Shape matching

Using the concepts introduced in the previous Chapter 2, more precisely in Sec. 2.3, the shape matching problem is properly defined in the present Chapter, and an algorithm that has been developed during the PhD project is presented. The implementation of this algorithm into our in-house kMC software is discussed in the subsequent Chapter 4.

Suppose the orientation of a structure is an important factor, and we would like to compare two structures A and B. We could compare them directly, meaning we could let A and B be written in their current reference frames, and compute some distance function, e.g. $RMSD(A, B)$ from Eq. (2.6).

$$RMSD(A, B) = \sqrt{\frac{1}{N} \sum_{i}^{N} d(a_i, b_i)^2} \qquad \text{(2.6 revisited)}$$

Such evaluation of similarity would be a bit meaningless, since the result depends on the reference frames which A and B are written in. Moreover, the result depends also on the permutation P of the elements i in sets A and B. To meaningfully compare A and B in this case, a common orthonormal reference frame and permutation should first be found. The problem of finding these two is in general called the shape matching problem.

The shape matching problem in general is further described and defined in Sec. 3.1. The Sections 3.2 through 3.7 describe our developed shape matching algorithm, and the different components of it. The Sec. 3.8 gives an evaluation of the algorithm, within different scenarios of its use.

3.1 Rewritten as an optimization problem

Formally, two sets A and B of vector elements, are considered congruent or equivalent if they are related by a transformation that preserves distances, i.e. isomet-

ric transformation. Such transformations are rigid translations $\mathbf{t}$, rigid rotations and/or reflections $\mathbf{R}$, and permutations P of indistinguishable vectors. The isometric transformation that fulfills the congruence relation between two structures gives a solution to the shape matching problem. The congruence relation can be written as:

$$P_B B = \mathbf{R}A + \mathbf{t}. \tag{3.1}$$

The problem of finding $\mathbf{R}$, $\mathbf{t}$, and P_B which best match structures A and B can be rewritten as an optimization problem,

$$\arg\min_{\mathbf{R},\mathbf{t}} \big\{ D(\mathbf{R}A + \mathbf{t}, B) \big\}, \tag{3.2}$$

in which D is a general distance function between two sets, that is i) variant under $\mathbf{R}$ and $\mathbf{t}$, ii) invariant under permutation P_B, and iii) returns value 0 when $\mathbf{R}$ and $\mathbf{t}$ are such that Eq. (3.1) is satisfied, i.e. when the best match is found. Due to the invariance of D on the permutation P_B, the problem of finding the permutations is taken out of this equation. It is important to highlight that D does not rely on an internal structural description (encoding), but rather it directly compares the "raw" state of the two structures, since $\mathbf{R}$ and $\mathbf{t}$ depend on their relative reference frames. When distortions and/or deformations are present, the transformation that minimizes Eq. (3.2), does not strictly return a 0 distance, but some minimum value. In that case, the relation between A and B is called a near-congruence, and the isometric transformation $\mathbf{R}$ and $\mathbf{t}$ is formally referred to as a near-isometry. This minimum distance value provides a measure of the quality of the congruence, i.e. a measure of the similarity between the structures.

The shape matching problem in the form of an optimization problem (Eq. (3.2)), is usually cast as the problem of finding a global minimum in the phase space of rotations, reflections, and permutations (neglecting for a moment the translations). In this formulation, using the $RMSD(A, B)$ from Eq. (2.6), as the distance function D in optimization Eq. (3.2), does not guarantee the existence of a single connected path from an arbitrary point in the phase space, to its global minimum. This can be seen via the following example. Take a structure of 8 points, placed at the corners of a cube as structure A, where the points are stored in a specific order, and take an identical structure as B. Then, keeping A fixed, act with a rotation $\mathbf{R}$ around the z-axis on B, and calculate $RMSD(A, \mathbf{R}B)$. Doing this for several permutations of the structure B results in the plot in Fig. 3.1. Notice that a change in the permutation can cause discontinuous jumps in the $RMSD$ values. Notice also the abundance of local minima. Taking these two into account, it is not clear that a path from an arbitrary point to the global minimum should exist.

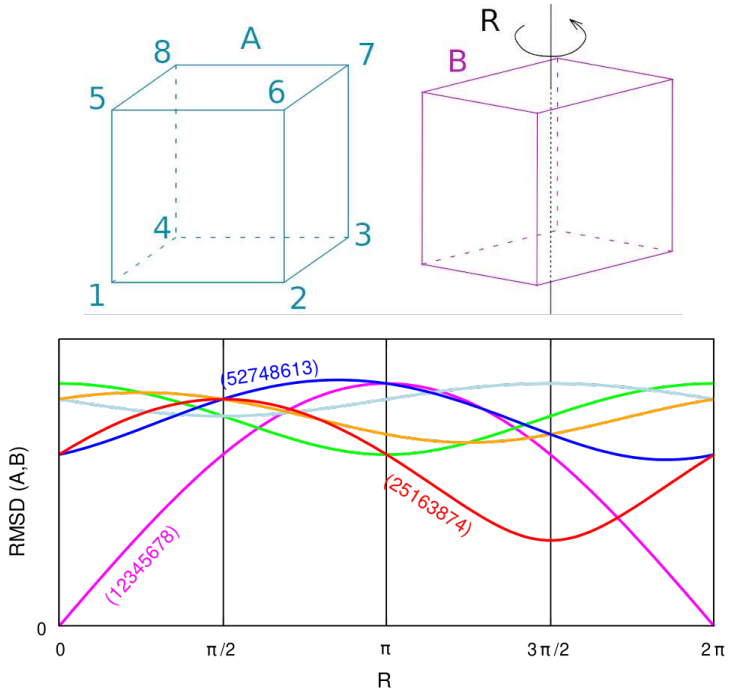


Figure 3.1: $RMSD$ as a function of rotations R and permutations between two identical cubes A and B, shown above the plot. Cube A is fixed while B is rotated around the z-axis only. Each color in the plot represents a different permutation of the rotated cube, some of them are explicitly labelled. Not all permutations are pictured, as there are in total $N_P = 8! = 40320$ possibilities.

3.1.1 In general terms ...

The main effort in solving the shape matching problem is in finding the optimal rotations and/or reflections $\mathbf{R}$, and the permutations P. For each of these problems separately, there exist successful and efficient algorithms.

In order to resolve rotations and/or reflections R among two sets of points A and B, the existing alogrithms rely on either symmetrization of a special matrix, or minimization of a cost function. Examples of the two ideas include the Lagrange multiplier method [94], matrix symmetrization [95, 96], decomposition of a matrix into orthonormal and positive semidefinite matrices [97], Singular Value Decomposition (SVD) [98, 99, 100], and quaternion eigensystem problem [101, 102, 103, 104] (a review of quaternions can be found in Ref. [91], and more recently in Refs. [92, 93]). Usually the cost function minimized is the $RMSD$ distance, which depends on the order of points (see Eq. (2.6)), therefore the rotation and/or reflection R computed depends on the given order of points. The problem of finding R is also called the orthogonal Procrustes problem [105].

Computing the order of points, or the permutation P, is called the assignment problem. It is introduced in Sec. 2.3.2 paragraph "Order of atoms". Restated briefly, it is a mapping from indices of one set to indices of another set, which minimizes a given cost in the form of a matrix. Solving the assignment problem might seem simple, but without the knowledge of any intrinsic relation between the points, the complexity increases very quickly as the total number of possible permutations N_P of indistinguishable points (atoms) in a structure grows as $N_P = \prod_{k=1}^{m} n_k!$, where m is the total number of different atomic types present, and n_k is the number of atoms of atomic type k.

The shape matching problem is very well known in the computer vision community, where it is generally known under the term point set registry, for some reviews on the problematic and associated algorithms, see Refs. [106, 107]. One of the most widely known algorithms in that community is the Iterative Closest Point (ICP) algorithm [108]. ICP exploits the idea of self-consistent iteration, where each step of the iteration combines an assignment procedure and consecutive rotation procedure, until a solution is found. A schematic of ICP procedure is shown by the plot on Fig. 3.2, where the space of rotations R is represented on the horizontal axis, the space of permutations is represented by different point styles, and the value of $RMSD_P(A, B)$ is represented by the vertical axis, where $RMSD_P(A, B)$ is the $RMSD(A, B)$ function of Eq. (2.6), evaluated for a given permutation P. For example, the first step of ICP algorithm in this plot is represented by computing a permutation P_1 at the starting rotation $R = 0$, then computing optimal rotation $R = R_1$ at fixed P_1, and then computing a new permutation P_2 at rotation $R = R_1$. The algorithm proceeds in this way until reaching a minimum of the $RMSD(A, B)$. On the plot of Fig. 3.2, the ICP procedure stops at rotation R_3, since it is then in a state where it cannot find a rotation or permutation that would further decrease the $RMSD(A, B)$, it has therefore reached a minimum. To compute the assignment of points, the originally proposed ICP algorithm uses the Hungarian algorithm [81], and the computation of optimal rotations is done using the quaternion approach [97]. It is however known that ICP might remain trapped in local minima of the transformation space. To mitigate, the original paper, Ref. [108] suggests using several different starting points, however other more efficient strategies and modifications exist [109].

3.1.2 Application to atomic structures

Applying the shape matching problem to atomic structures means that point sets now represent atomic structures, and each point has at least one property associated to it, which is the chemical specie. Some strategies of resolving the shape matching as an optimization problem (Eq. (3.2)) specific for atomic structures are briefly summarized in the following. The main problem is to find optimal atomic

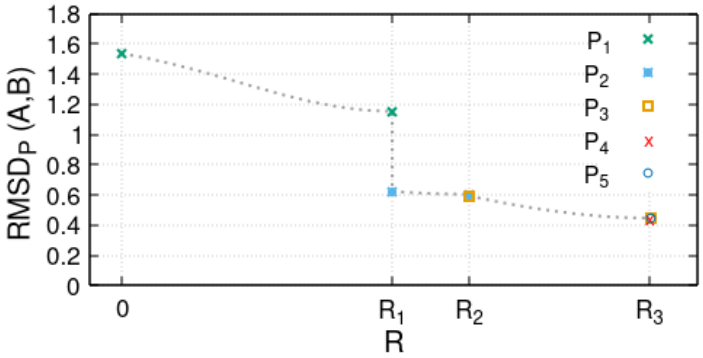


Figure 3.2: Schematic representation of the Iterative Closest Point (ICP) algorithm. The $RMSD_P(A, B)$ is the $RMSD(A, B)$ for a given permutation P.

assignments, and rotations and/or reflections.

The authors in Ref. [110] propose an approach for the alignment of molecules based on ideas from image recognition, which relies on filtering methods to obtain atomic assignments. Optimal rotations are later resolved by applying an SVD minimization.

Finding a rough equivalent reference frame (or Eckart frame[89, 90]) through, for example, principal axes of inertia, might provide a good-enough rotation for identifying reasonable assignments, which can in turn be used to find true optimal rotations, see for instance Refs. [111, 112, 113]. This idea is not suitable for isotropic or compact structures, and crystalline or bulk environments, since the principal axes might be ambiguously defined, due to the symmetry in the structures. The calculation of principal axes also involves weights, which usually correspond to atomic masses.

A successful Monte Carlo-based decision scheme for finding the global minimum of $RMSD$ has also been reported in Ref. [85].

An alternative idea to the self-consistent "minimization" of ICP could be to parse through the full rotation-space by brute-force, and compute optimal score for each possible rotation. The space of all possible rotations is however much too large to do that, so one would like to somehow reduce it into a smaller set of points. That is, to effectively partition the space of rotations into a number of points $\mathbf{R}$, which is sufficiently low such that an algortihm can try all of them, and determine the best. The immediate questions are: how many points are sufficient, and how to choose them such that the best match is included? The following works have proposed some resolution to these ideas.

Authors in Ref. [83] suggested an algorithm in which the space of possible rotations and reflections is discretized into a uniform grid of points. For each

grid-point $\mathbf{R}_i$ the optimal atomic assignment P_B is obtained as the optimal assignment of an inter-structure distance matrix with the Hungarian algorithm[81], which is then used to minimize rotations with SVD[99]. The grid-point $\mathbf{R}_i$ that brings the lowest final distance is chosen as the optimal rigid transformation, and the solution of the shape matching problem. The distance function that is minimized by SVD is inherently the $RMSD(A, B)$, thus this algorithm effectively minimizes the $RMSD$ distance. The unifrom-grid strategy is however difficult to optimize, as the number of grid points is not directly related to any property of the system.

A slightly different approach has been proposed in Ref. [84], with an atomic-centered grid of approximate rotations on both structures, meaning that the atoms present in the system define the possible grid-points $\mathbf{R}_i$. Specifically, the atoms farthest from the center are selected as the basis for points $\mathbf{R}_i$ and subsequently also to find the approximate rotation. The atomic assignments are obtained via finding optimal assignment of the inter-structure distance matrix with the Hungarian algorithm. In order to precisely define the atoms which give a possible basis, a parameter on similarity at atomic-level is introduced, which can be system-dependent. However, if the compared structures are identical, this method of partitioning the rotation space ensures the inclusion of best-match rotation point, or at least a point in its close proximity.

3.2 Our algorithm

The algorithm presented here follows the idea of partitioning the rotation space $\mathbf{R}$ into a set of discrete points $\mathbf{R}_i$. For each of those points, we calculate a distance $D(\mathbf{R}_i A, B)$, where D complies with the constraints given in Sec. 3.1. The point $\mathbf{R}_i$ which gives a minimum D is chosen as an approximate rotation $\mathbf{R}_{apx}$, which is used to compute the correct atomic assignments P_B, which is then in turn used to compute the optimal rotation which minimizes a given score.

A schematic of this idea is shown on Fig. 3.3, where the space of possible rotations has been partitioned into 12 points $\mathbf{R}_a$ on the horizontal axis. For each of these points, the value of distance $D(\mathbf{R}_a A, B)$ is represented on the vertical axis. The rotation point which returns the minimal value of D is in this case $\mathbf{R}_{apx} = \mathbf{R}_a^6$, the permutations are computed in this point and used to calculate the optimal rotation $\mathbf{R} = \mathbf{R}_{fin}$, which minimizes $D(\mathbf{R}A, B)$, showcased by the dashed line.

Our algorithm consists of two parts. The first part iteratively solves the approximate rotation $\mathbf{R}_{apx}$, which makes it possible to compute the correct atomic assignments P_B. The second part uses the atomic assignments to compute the final optimal rotation via standard SVD. We develop the approach Iterative Rotations

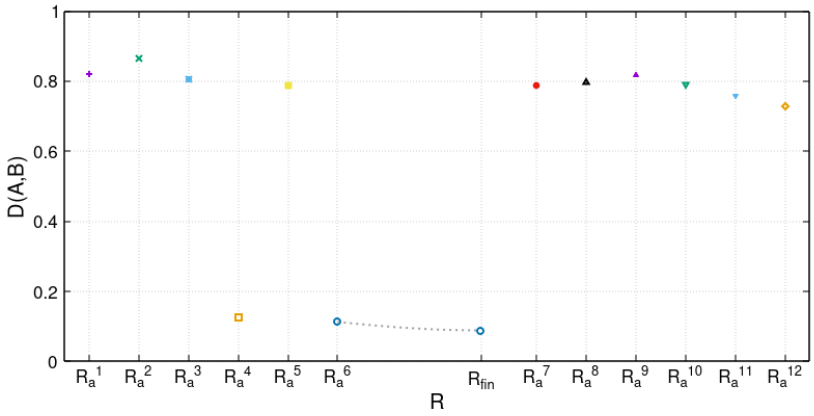


Figure 3.3: Schematic of the idea: the space of rotations $\mathbf{R}$ is partitioned into points $\mathbf{R}_a$ on the horizontal axis, the value of $D(\mathbf{R}A, B)$ is given on the vertical. The rotation point that gives the lowest D is chosen as the approximate rotation $\mathbf{R}_{apx} = \mathbf{R}_a^6$, the permutation is then computed and used to find the final optimal rotation $\mathbf{R}_{fin}$.

and Assignments (IRA), to obtain the approximate rotation $\mathbf{R}_{apx}$ in the first part of our algorithm. It is described in more detail in Sec. 3.3. To compute the atomic assignments P_B, we develop our own algorithm: Constrained Shortest Distance Assignment (CShDA), that solves the Linear Assignment Problem (LAP) under the one-to-one assignment constraint. It is described in more detail in Sec. 3.4. The ability of CShDA to deal with structures containing different numbers of atoms opens a door to a generalization of IRA to structures with different number of atoms, in Sec. 3.5. The Sec. 3.6 describes the Singular Value Decomposition-based technique of obtaining optimal rotations, in case when atomic assignments are known, that is used to obtain final rotation when structures are not exactly congruent.

The shape matching algorithm developed here has been published in Ref. [114].

3.3 Iterative Rotations and Assignments

One of the two main goals of our algortihm is to find the optimal rotation $\mathbf{R}_{fin}$, following the Eq. (3.3). This rotation should be such that one of the two structures (namely structure B) is kept fixed,

$$\mathbf{R}_{fin}A = B. \tag{3.3}$$

In order to find the rotation $\mathbf{R}_{fin}$ between the atomic structures A and B, the main idea is to find a basis set for each structure, such that the structures are equal

when expressed each in its basis. The relative rotation between these two bases then gives the rotation $\mathbf{R}_{fin}$. In order to explain this idea behind IRA in a clear way, we shall first assume that the two structures being compared are exactly congruent, in Sec. 3.3.1 and Sec. 3.3.2, where the solution of rotation gives an exact match. Then we extend the idea to near-congruent structures in Sec. 3.3.3, where the solution of rotation becomes approximate.

Henceforth we assume that all structures are written in a 3-dimensional Cartesian space with an orthonormal basis. In this assumption, any rigid rotation matrix $\mathbf{R}$ is composed of three orthonormal vectors. Similarly, any orthonormal basis set is equivalent to a rotation matrix $\mathbf{R}$. In other words, the rotation of a structure by $\mathbf{R}$ can be understood as a change of basis operation, and vice versa.

3.3.1 Congruent structures

Assume structures A and B are exactly equivalent (congruent), that is, the relation between them written as Eq. (3.1) can be exactly satisfied, but we do not know the rigid transformation (composed of $\mathbf{R}$, $\mathbf{t}$, and P_B) between them. Assume we can nullify the translation $\mathbf{t}$ by shifting both structures to a known common origin, such as their geometric centers. Let β be the 3×3 matrix containing three orthonormal basis vectors for a reference frame of structure A. Similarly, let γ be the 3×3 matrix containing three orthonormal basis vectors of a reference frame of structure B. When β and γ are such that structure A expressed in the β basis is equal to the structure B expressed in the γ basis, written as:

$$\beta A = \gamma B, \tag{3.4}$$

then the rotation $\mathbf{R}_{fin}$ from Eq. (3.3) can be written as:

$$\mathbf{R}_{fin} = \gamma^{-1}\beta. \tag{3.5}$$

Notice that since γ is orthonormal, the inverse is equal to its transpose $\gamma^{-1} = \gamma^T$, which means the Eq. (2.11), giving a relative rotation between two rotations (reference frames), is satisfied, and $\mathbf{R}_{fin}$ indeed gives the relative rotation between structures A and B.

$$\mathbf{R} = \gamma^T\beta, \tag{2.11 revisited}$$

3.3.2 Parsing the space of rotations; definition of a basis

In order to find the orthonormal bases β and γ (Eq. (3.4)), we partition the space of all rotations into a set of points. Each point represents a rotation, defined by a basis composed of orthonormalized vectors of atoms present in the structure. Each basis is constructed from two atomic vectors, and their cross product as follows. The

first basis vector $\hat{\mathbf{e}}_1$ is a normalized atomic vector of some atom, the second basis vector $\hat{\mathbf{e}}_2$ is the orthonormal part of some other atomic vector, noncollinear to the first one, and the third basis vector has two possible values: $\hat{\mathbf{e}}_3 = \hat{\mathbf{e}}_1 \times \hat{\mathbf{e}}_2$, and $\hat{\mathbf{e}}_3 = \hat{\mathbf{e}}_2 \times \hat{\mathbf{e}}_1$. The second choice of the third basis vector effectively represents a mirror-reflected basis. This choice allows the matching of mirror-reflected structures. Each rotation point, or basis, is thus completely defined by the atoms within the structure. A simplified 2D schematic of such partitioning for a hypothetical atomic structure is shown in Fig. 3.4, where the space of all possible rotations is drawn in dashed grey, and the points in green represent its partitioning. When A and B

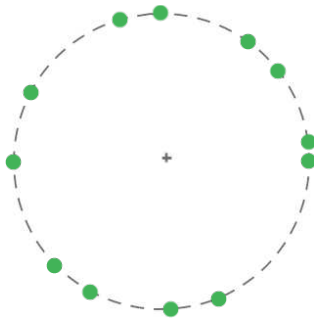


Figure 3.4: Partitioning of rotation space into a set of points, for some hypothetical atomic structure. In 2D for simplicity.

are exactly congruent, the partitioning of rotations for structure A will be identical to that of structure B. The task then is to find a pair of bases β and γ (points in rotation space), which satisfy Eq. (3.4). Since the partitions are equivalent, we can arbitrarily choose one basis β for structure A, and then iterate through all the possible points (bases) γ in the structure B, evaluating Eq. (3.4) for each, until a satisfactory γ is found. This procedure gives the name to our algorithm, Iterative Rotations and Assignments, abbreviated IRA.

3.3.3 Near-congruent structures

In the situation when A and B are not exactly congruent (they are near-congruent), meaning the relation in Eq. (3.1) cannot be exactly satisfied, in other words, there is some distortions between structures A and B, the partitioning of the rotation space is not anymore exactly equal for both structures. Since there are now some distortions in the atomic positions, this induces distortions in the bases constructed by our procedure, which define points in the rotation space. Thus we cannot anymore expect to find a pair of bases β and γ, such that condition in Eq. (3.4) is satisfied exactly. This means the rotation $\mathbf{R}_{fin}$ found by IRA procedure is at best an *approximate* rotation $\mathbf{R}_{apx}$. The approximate rotation would be constructed

as $\mathbf{R}_{apx} = \gamma^{-1}\beta$, from bases β and γ which are now such that the condition in Eq. (3.4) is satisfied as best as possible. This is done by computing the distance:

$$D(\gamma^{-1}\beta A,\ B), \tag{3.6}$$

for a chosen β, and each possible γ. The distance function $D(\gamma^{-1}\beta A,\ B)$ is the Hausdorff distance from Sec. 2.3.2. More precisely, the computation of D follows Eq. (2.10). In order to compute its value, we first need to compute the one-to-one atomic assignment.

$$h(A, B) = \max_i \min_j d(a_i, b_j) \tag{2.10 revisited}$$

3.4 Constrained Shortest Distance Assignment

To compute the atomic one-to-one atomic assignments we developed our algorithm called Constrained Shortest Distance Assignment (CShDA).

An atomic assignment, or mapping between atoms $a_i \in A$ and $b_j \in B$ is the set of pairs of two atoms $i,\ j$, such that each pair gets a minimum possible cost, under the constraint that each atom i can only have one and only one match j, so-called one-to-one assignment. The set of these pairs constitutes the atomic assignment, which can be written as a permutation matrix P.

The constraints of atomic assignment used in CShDA are described in Sec. 3.4.1. The link between CShDA and distance function D is described in Sec. 3.4.2. The comparison/contrast of CShDA with the Hungarian algorithm, which is one of the most widely used LAP solvers is given in Sec. 3.4.3. The possibility of CShDA to assign structures with different number of atoms is shown in Sec. 3.4.4.

3.4.1 Constraints

The idea of CShDA is that the distances from an atom $a_i \in A$ to all atoms $b \in B$ are used as a cost for computing the assignment of atom a_i, such that shortest distances are prioritized for each atom a_i locally. To showcase, an atom a_i gets assigned an atom b_j with the shortest distance $d(a_i, b_j)$ among all atoms b. However, if during the algorithm an atom $a_i \in A$ which is assigned an atom $b_j \in B$ with some distance $d(a_i, b_j)$, and another atom $a_{i'} \in A$ gets assigned the same atom $b_j \in B$ with a distance $d(a_{i'}, b_j) < d(a_i, b_j)$, the atom $a_{i'}$ will be prioritized for this b_j, and the atom a_i gets assigned a different atom. Symbolically, CShDA iteratively assigns a single atom $a_i \in A$ to a single atom $b_j \in B$ following:

$$a_i \to b_j \quad | \quad \min_{b_j \in B} d(a_i, b_j) \quad \forall a_i \in A \tag{3.7}$$

with the constraint that b_j has not yet been assigned with a distance lower than $d(a_i, b_j)$, where d is the Euclidean distance between the points. When applied to a general set of points, this kind of local assignment is sometimes referred to as bottleneck LAP [115], or more precisely, the inverse-bottleneck LAP.

3.4.2 Link between CShDA and distance D

One can realize that our CShDA algorithm corresponds to the min part of the Hausdorff distance in Eq. (2.10), with the additional constraint of one-to-one assignment. As distance function D we choose the Hausdorff distance in Eq. (2.10). The evaluation of Eq. (2.10) is then the maximal distance $d(a_i, b_i)$ among all points i, where the order of atoms b_i follows the assignment provided by the CShDA algorithm. The computation of distance D is therefore intimately linked to the CShDA assignment algorithm. In fact, each evaluation of D requires the computation of CShDA assignment, and each computation of the assignment with CShDA returns as result the distance D, along with the assignment.

3.4.3 Comparison of CShDA with the Hungarian

There are two main differences between our CShDA and the Hungarian algorithm.

Firstly, the criteria for the assignment of two atoms differ. The Hungarian algorithm [81] assigns indices such that the total sum of the cost is minimized, where the cost of assignment is the distance between two points. In CShDA, each assignment cost is minimized separately, under the one-to-one constraint, where the assignment cost is the distance between points. In order to illustrate the difference, we compute the atomic assignments for two structures in two rotated states (Fig. 3.5), using the Hungarian algorithm following the implementation by Munkres [82], and using our CShDA algorithm. What we observe when computing the assignments for two structures in different rotated states, is that the CShDA algorithm tends to concentrate the maximum deviations on a small number of atoms, contrary to the Hungarian algorithm that favours smaller deviations, but spread over several atoms. Prioritizing a smaller total distance cost over local distances, as done in the Hungarian algorithm, means that globally "distorted" solutions are preferred over rigid single mismatches, see Fig. 3.5.

The second difference is that the Hungarian algorithm requires two structures to have equal number of atoms, as the cost of assignment is computed from an all-to-all distance matrix, which needs to be square. While it is true that any square matrix can be made to be non-square by the addition of ghost rows or columns at specific indices, this is not trivial since it is not known a priori which should these indices be. Our CShDA algorithm does not have such a constraint. The only requirement for CShDA is that the number of atoms n_A in structure A is

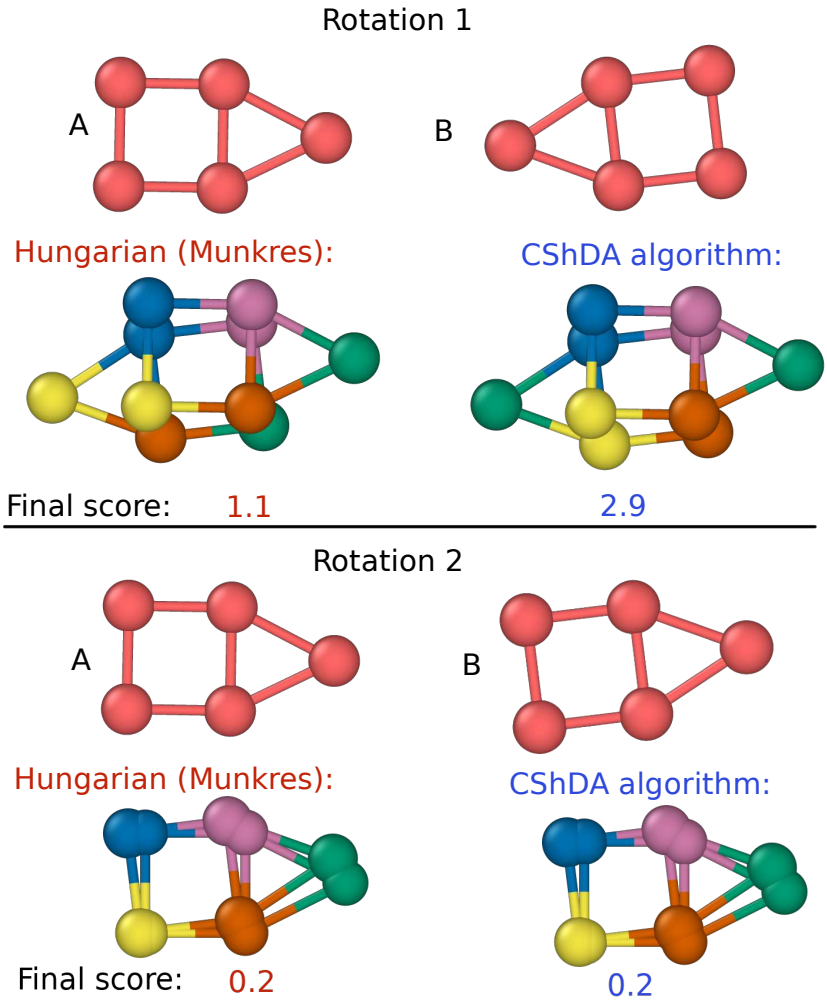


Figure 3.5: A schematic of the assignment problem, with structures A and B in two rotated states. On the left the assignment by the Hungarian algorithm following the implementation proposed by Munkres [82], and on the right by our CShDA algorithm. The colors show final assignments of atoms, e.g. a blue atom is assigned to a blue atom, yellow atom to yellow, etc. The final scores are computed as $max(d(a_i, b_i))$. The first rotated state could represent a particular intermediate step within the iterative rotations procedure (IRA).

$n_A \leq n_B$, where n_B is the number of atoms in structure B, as is explained further in Sec. 3.4.4.

3.4.4 Different number of atoms

In the IRA part of the algorithm (Sec. 3.3), the evaluation of distance function D in Eq. (3.6) is compliant with the one-to-one matching constraint of the CShDA, and strictly corresponds to the Hausdorff distance $h(A, B)$. Due to the relatively low number of atoms in the atomic structure matching, the usage and implementation of the Hausdorff distance needs some attention. The expression for $h(A, B)$ in Eq. (2.10) is only commutative when A and B contain the same number of points, which is the reason the expression for the Hausdorff distance is generally written in the form of Eq. (2.9), which penalizes the situation where some points are present in one structure but not in the other.

$$d_H(A, B) = \max(h(A, B), h(B, A)) \qquad \text{(2.9 revisited)}$$

Fig. 3.6 schematically shows the shortest distances between points of set A (triangles) and points of set B (circles) as arrows, where the largest among them is colored in red and represents the value of $h(A, B)$, and $h(B, A)$ respectively. As described in Sec. 3.4, the assignment of atoms is done under the one-to-one constraint, which poses a problem for the situation of $h(B, A)$ on the right side of Fig. 3.6, where B contains more atoms than A, since two atoms of B get assigned to the same atom of A. A mitigation for avoiding this problem is to systematically impose that the number of atoms $n_A \leq n_B$, which is the situation of $h(A, B)$ on the left side of Fig. 3.6. This imposition also opens up the possibility of matching structural fragments. But, it also adds another layer of complexity to the algorithm, since the choice of common origin is not anymore trivial: the geometric centers of structures can be far from equivalent when they contain different number of points. Which calls for a re-statement, and generalization of the algorithm in Sec. 3.5. When computing CShDA (or distance D) for structures A and B where the number of atoms $n_A \leq n_B$, the atoms $a_i \in A$ get assigned atoms $b_j \in B$. The unassigned atoms of B get ignored, and the distance D is computed as the max value of distances only among the assigned pairs of atoms, which is n_A number of pairs. We enforce that the permutation P_B of set B will in this case be such that the points of A will be assigned to the first n_A points of $P_B B$. The unassigned points of B will be permuted to the end of the set.

3.5 Re-statement of the algorithm: central atom

As has been shown in Sec. 3.4.4, the atomic assignment algorithm CShDA is not restricted to structures with equal number of atoms, and this opens up the possibility of matching structural fragments. However the additional complexity is that the common origin of two structures with different number of atoms is

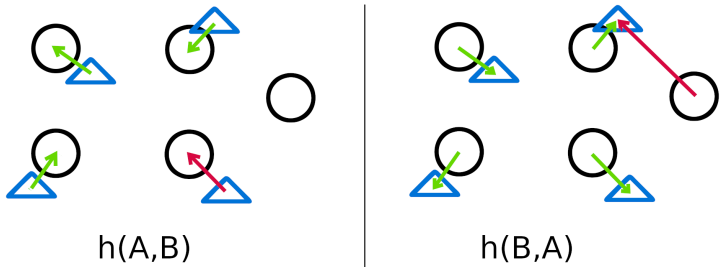


Figure 3.6: Schematic representation of the difference between $h(A, B)$ on the left, and $h(B, A)$ on the right, when A and B contain different number of points. Set A is represented by triangles, set B by circles. Arrows show the minimum distances between points in green, and the maximum value in red, $h(A, B)$ and $h(B, A)$ respectively.

not straightforward to determine. For this reason, the IRA algorithm needs to be restated such that the possibility of matching structural fragments is taken into account.

As mentioned in the previous section, the mitigation for non-commutativity of $h(A, B)$ is to always impose structure A to have less or equal number of atoms as structure B, $n_A \leq n_B$. In order that the one-to-one constraint for determining the assignments in CShDA is satisfied, all atoms from A must have a matching atom in B. Therefore, if the structures A and B are congruent, any atom that is present in structure A must also be present in the structure B. We thus use this idea to find a common origin of the reference frames, as one of the atoms of structure A. The algorithm thus becomes centered on a particular atom, called the central atom.

The particular choice of the central atom seems quite arbitrary at this point, so we choose it the atom that is closest to the geometric center of A (it is shown later in Sec. 3.8.4, that this might not always be the best idea). Let the atomic vector of the central atom in structure A be labelled $\mathbf{r}_C$. Since the basis β that is set in structure A is now also atom-centered, we call it Ω. The structure A in the new basis can the be expressed as:

$$A' = \Omega(A - \mathbf{r}_C). \tag{3.8}$$

The iterative search for basis γ in structure B must now take into account that it can potentially be centered on any atom $b_J \in B$, so an additional loop must be done. For each potential central atom J, all possible γ bases need to be constructed and checked. The atomic vector of atom J is labelled as $\mathbf{r}_C^J$, and the set of all possible γ bases, for all possible central atoms J is labelled $\{U_J\}$. The structure B in the new basis can then be written:

$$B' = U_J(B - \mathbf{r}_C^J). \tag{3.9}$$

In this notation, the IRA iterates over the set of $\{U_J\}$ bases, and for each computes

$$D(\mathbf{R}_{apx}A,\ B), \tag{3.10}$$

where $\mathbf{R}_{apx} = U_J^{-1}\Omega$. It chooses atomic ventor $\mathbf{r}_C^J$ and basis U_J where D is smallest. The approximate rotation $\mathbf{R}_{apx}$ and translation $\mathbf{t}_{apx}$ can then be constructed as:

$$\begin{aligned} \mathbf{R}_{apx} &= U_J^{-1}\Omega \\ \mathbf{t}_{apx} &= \mathbf{r}_c^J - \mathbf{R}_{apx}\mathbf{r}_c. \end{aligned} \tag{3.11}$$

At this point, the permutation P_B is computed using the CShDA algorithm. With all these information the shape matching algorithm can enter the last step, which is finding the optimal rotation $\mathbf{R}$ using the standard SVD algorithm, see Sec. 3.6.

3.6 Optimal rotations with Singular Value Decomposition

In the case in which the two structures are not equivalent, i.e. in the case of near-congruence, after finding the atomic assignments P by our IRA algorithm, the optimal rotations are found via an SVD-based algorithm [100] as follows.

Point sets A and B are shifted to their geometrical centers, obtaining $A' = \{a_i' = a_i - a^c\}$ and $B' = \{b_i' = b_i - b^c\}$ where a^c and b^c are the vectors of geometric centers of A and B respectively. A 3×3 matrix $\mathbf{H}$ is constructed from n_A points which are common to A' and B' (to enable the decomposition for sets with different number of atoms),

$$\mathbf{H} = \sum_i^{n_A} |b_i'\rangle\langle a_i'|, \tag{3.12}$$

with a_i' and b_i' the vector points of A' and B', and $|..\rangle\langle..|$ denoting outer vector product. The SVD returns three matrices, $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal rotation matrices, and $\mathbf{S}$ is a diagonal matrix containing the sinular values. These matrices are such that $SVD(\mathbf{H}) = \mathbf{USV}^T$. The rotation matrix $\mathbf{R}$ is then found as:

$$\mathbf{R} = \mathbf{UV}^T, \tag{3.13}$$

and if $det(\mathbf{R}) = -1$, then $\mathbf{R}$ is multiplied by $diag(1, 1, -1)$. The translation vector $\mathbf{t}$ is found as:

$$\mathbf{t} = a^c - \mathbf{R}b^c. \tag{3.14}$$

Rotation $\mathbf{R}$ and translation $\mathbf{t}$ found in this way, are such that the $RMSD(A, B)$ is minimized (details on SVD can be found in Ref. [100]).

It is commonly believed that SVD-based algorithms are not particularly suited for matching purposes, due to the ability of SVD to find non-proper rotations [91], i.e. rotation matrices with negative determinant. Such improper rotations correspond to reflections (sometimes also addressed as pseudorotations [116]), i.e. inversions, or mirroring over some axis, which changes the chirality of a vector set (which is not always desired, e.g. Ref. [110]). It has been suggested [100] to mitigate this issue by multiplying the rotation matrix by $diag(1, 1, -1)$, thus forcing a positive determinant. This strategy might however result in a completely wrong rotation, as the matrix $\mathbf{H}$ depends on the order of points (see Eq. (3.12)).

As our IRA approach is able to also suggest mirror-reflected reference frames (see Sec. 3.3.2), via inverting the third basis vector, it is always able to rigorously keep track of what has been suggested, and properly enforce the final rotation matrix to have $det(\mathbf{R}) = 1$ (corresponding to rotation), or by multiplying it by $diag(1, 1, -1)$ to obtain $det(\mathbf{R}) = -1$ (corresponding to reflection).

3.7 Algorithmic details

A flowchart of our shape matching algorithm is shown on Fig. 3.7, where the first part of the algorithm (IRA and CShDA) is colored in blue, the second in green (SVD), and final solution given in red. The Sec. 3.7.1 and Sec. 3.7.2 contain the

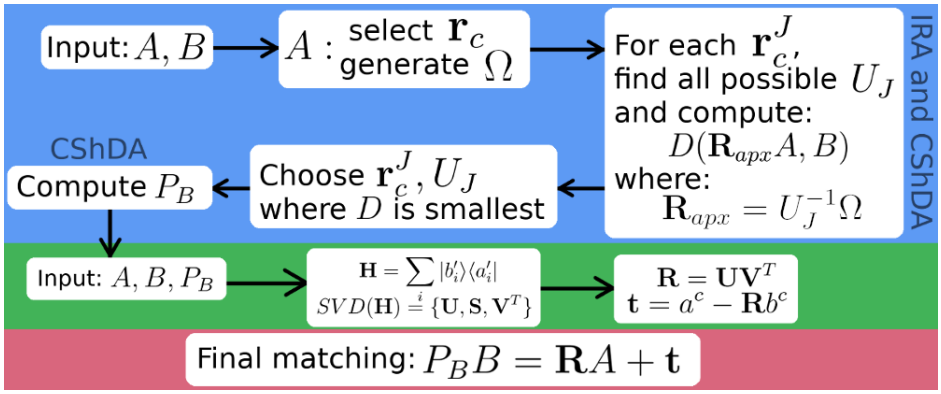


Figure 3.7: Flowchart of the algorithm. First part of the algorithm, colored in blue, gives an approximate solution to rotations and translations, and solution to the permutation P_B. The permutation is needed in the second part of the algorithm, colored in green, which finds the optimal rotation and translation by utilising the SVD method. The final solution of the matching algorithm is colored in red.

details on the implementations of IRA and CShDA. The idea is to lay out the ideas

such that they can be reimplemented by any reader. The subsection 3.7.3 contains the information on specific implementation as a library.

3.7.1 Iterative Rotations and Assignments (IRA)

The philosophy of IRA is to set some reference frame in A, and then iterate over B in order to re-find that same reference frame. The iteration procedure is given by the partitioning of the space of all rotations into a set of dicrete points, as described in Sec. 3.3.2. The number of points to be iterated over depends on the number of non-collinear atoms around the central atom of A, which can potentially be very large. However, if we choose the atoms which set basis Ω for structure A in a smart way, we can a priori exclude a huge portion of the iteration points. The pseudo algorithm for the general case when structures A and B contains different number of atoms is given in Alg. 4, and the case when they contain equal number of atoms in Alg. 5, the former is described in the following. The latter can also be used with different number of atoms, but there needs to be a known point, common to both structures, which replaces the geometric center as the central point. The main part of IRA, which is the iteration through possible rotations (orthonormal bases) is given in Alg. 3.

In order to distinguish the cases when A and B contain equal or different number of atoms, we use a different notation for the bases in each. In case when number of atoms is equal, we use β and γ, and when number of atoms is different we use Ω and U_J. The difference is purely in notation, to emphasize that U_J depends on the index of central atom J. For the main part of IRA in Alg. 3, this distinction makes no difference.

The central atom of A is chosen at the start, as the atom closest to the geometric center of A (Alg. 4 line 2). The atoms of A are then shifted and sorted by their norm, which is by their distance from the central atom (Alg. 4 line 5). Then the basis Ω is chosen with two noncollinear atoms, as close as possible to the central atom (Alg. 4 line 6). The norm of the larger of these two atomic vectors is stored as a "cutoff" distance `kmax`, used for terminating the iterations in structure B. In a sense, this distance determines the search space of rotation points U_J, around each potential central atom in structure B. The "cutoff" distance `kmax` is multiplied by default by 1.2, to slightly increase the search space, and thus accomodate possible small distortions, and numerical noise (Alg. 4 line 8).

The iterative loop over the structure B can thus begin (Alg. 4 line 9), each candidate central atom is selected, and the strucutre B shifted to that origin. Then the atoms are first sorted by their distance, such that the loop over rotations (Alg. 3, called by Alg. 4 line 13) can exit as soon as an atom is outside the `kmax` region (Alg. 3 lines 3 and 5).

This method profoundly cuts down the number of basis that are checked. It

Algorithm 3 The main part of IRA procedure: looping through possible bases γ, and selecting the one which gives minimum value of D.

Input: structures A, B, basis β, cutoff `kmax`
Output: rotation γ, distance D

procedure GETGAMMA(A,B,β, `kmax`, γ,D)
  for $b_i \in B$ **do**
    if($|b_i| >$ `kmax`) **exit**
    for $b_j \in B$ **do**
      if($|b_j| >$ `kmax`) **exit**
      for $m = 1$ **and** $m = -1$ **do**
        $\gamma \leftarrow$ SelectBasis(b_i, b_j, m)
        $A \leftarrow$ Rotate(A, $\gamma^{-1}\beta$)
        $h^{ijm} = D(A, B)$ ▷ Compute by CShDA
      end for
    end for
  end for
  select i, j, m such that h^{ijm} is minimum
  $\gamma \leftarrow$ SelectBasis(b_i, b_j, m, γ)
  $D \leftarrow h^{ijm}$ minimum
end procedure

can now be expressed as $n_C(n_C - 1)$, where n_C is the number of noncollinear atoms within the `kmax` region of the current central atom. Since the value of `kmax` is set by the nearest atoms to the central atom of A, it is most probably on the order of the first-neighbour distance. This means that the number of atoms n_C is given by the compactness and isotropy of the structure. The largest number of nearest neighbour for any known solid is 12, in the face centered cubic (fcc), and hexagonal close-packed (hcp) crystal structures. Thus, for fcc and hcp the number $n_c = 12$. Additionally, since each atom in fcc or hcp first-neighbour environment has one collinear atom, the total number of rotation points would be $n_C(n_C-2) = 120$. This number is however still not exact, due to the multiplication of `kmax` by 1.2.

A sketch is shown in Fig. 3.8 depicting a hexagonal structure of atoms, whose positions are marked by black crosses, and their respective indices are marked. The first basis vector $\hat{\mathbf{e}}_1$ marked in grey has been selected on the atom 2, the possible second basis vectors are given from orthogonal parts of vectors to atoms 3, 4, 6, and 7. They are marked in green as $\hat{\mathbf{e}}_2$ and $\hat{\mathbf{e}}'_2$. The orthogonal parts of vectors given by atoms 3 and 7 coincide, the same for atoms 4 and 6. The atom 5 is collinear with the first basis vector, thus the vector $\hat{\mathbf{e}}''_2$ cannot be selected for a basis. The number of iterations for this structure would thus be 4 per each atom

Algorithm 4 The IRA procedure for structures containing different numbers of atoms.

Input: Atomic structures A, B, stored in arrays

Output: Rotation matrix $\mathbf{R}_{apx}$, translation vector $\mathbf{t}_{apx}$, permutation P_B, distance D

1: vectors :: $a_i \in A$, $b_i \in B$
2: ChooseCentralAtom(A, a_c) ▷ closest to geometric center
3: $\mathbf{r}_c \leftarrow a_c$
4: $A \leftarrow$ Shift(A, $\mathbf{r}_c$)
5: $A \leftarrow$ SortByNorm(A)
6: $\Omega \leftarrow$ SelectBasis(a_i, a_j)
7: $A \leftarrow$ Rotate(A, Ω)
8: $\texttt{kmax} = max(|a_i|,\ |a_j|) * 1.2$
9: **for** $b_J \in B$ **do** ▷ candidate central atoms
10: $\mathbf{r}_c^J \leftarrow b_J$
11: $B \leftarrow$ Shift(B, $\mathbf{r}_c^J$)
12: $B \leftarrow$ SortByNorm(B)
13: **call** GETGAMMA$(A, B, \Omega, \texttt{kmax}, U_J, D_J)$ ▷ Get U_J
14: **end for**
15: select J such that D_J is minimum
16: $U_J \leftarrow$ SelectBasis(b_i, b_j, m)
17: $\mathbf{R}_{apx} \leftarrow U_J^{-1}\Omega$
18: $\mathbf{t}_{apx} \leftarrow \mathbf{r}_c^J - \mathbf{R}_{apx}\mathbf{r}_c$
19: $D \leftarrow D_J$ minimum
20: compute P_B from CShDA
21: **return** $\mathbf{R}_{apx}$, $\mathbf{t}_{apx}$, P_B, D

giving $\hat{\mathbf{e}}_1$, thus $6 \cdot 4 = 24$.

For each possible basis U_J constructed in B, the distance $D(A, B)$ is calculated (Alg. 3 line 9), as stated in Sec. 3.5. Each iteration thus stores in memory:

- an integer giving the index of candidate central atom J;
- two integers that are the indices of atomic vectors giving $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$;
- an integer m, signalling the choice of $\hat{\mathbf{e}}_3$ as: $\hat{\mathbf{e}}_3 = \hat{\mathbf{e}}_1 \times \hat{\mathbf{e}}_2$ when $m = 1$, and $\hat{\mathbf{e}}_3 = \hat{\mathbf{e}}_2 \times \hat{\mathbf{e}}_1$ when $m = -1$, and;
- the value D calculated with that basis.

At the end of the iteration, the lowest value D is chosen with other corresponding data in memory (Alg. 3 line 13), from which the basis U_J can be reconstructed.

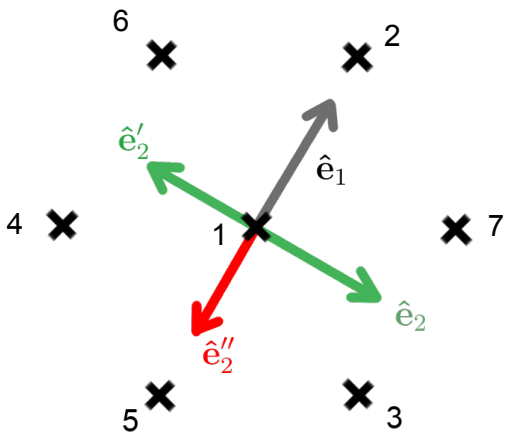


Figure 3.8: A sketch of the basis construction possibilities in a 2D hexagonal structure, atomic positions are given as black crosses, with their respective indices marked. The first basis vector $\hat{\mathbf{e}}_1$ is already chosen marked in grey, all possible choices for the second basis vector $\hat{\mathbf{e}}_2$ are marked. The choice of $\hat{\mathbf{e}}_2''$ is marked red, since it is not an acceptable vector, as it is collinear to $\hat{\mathbf{e}}_1$.

The structure B is shifted such that atom J is at the origin. The vectors $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$ are constructed from given atomic vectors, and $\hat{\mathbf{e}}_3$ is multiplied by m, which forms the basis U_J (Alg. 4 line 16). Bases Ω and U_J are combined to give $\mathbf{R}_{apx}$ (Alg. 4 line 17) according to Eq. (3.11). The atomic vector $\mathbf{r}_C^J$ of the central atom J, is combined with the atomic vector $\mathbf{r}_C$ of central atom in structure A, and rotation $\mathbf{R}_{apx}$ to give translation $\mathbf{t}_{apx}$ (Alg. 4 line 18) according to Eq. (3.11). Finally, the permutation P_B and distance $D(A, B)$ are computed by CShDA (Alg. 4 line 19 and 20).

The differences of Alg. 4 with respect to the case when structures A and B contain equal number of atoms in Alg. 5, are the choice of central point, and the loop over candidate central atoms of B.

3.7.2 Constrained Shortest Distance Assignments (CShDA)

The CShDA algorithm is outlined in Algorithm 6.

As the first step, the all-to-all distance matrix is computed, from atoms of structure A to atoms of structure B, see Alg. 6 lines 6-11. The matrix elements d_{ij} are such that

$$d_{ij} = ||\mathbf{r}_i - \mathbf{r}_j||, \tag{3.15}$$

where $\mathbf{r}_i$ is a vector of atom $a_i \in A$, and $\mathbf{r}_j$ is a vector of atom $b_j \in B$. If the atoms a_i and b_j are not of the same atomic type, then the distance is set to some high value $d_{ij} = 990.0$, to make the final assignment of this pair of atoms extremely

Algorithm 5 The IRA procedure for structures containing equal number of atoms. Also applicable with different number of atoms, but a known common point. In that case, the geometric center is replaced by that point.

Input: Atomic structures A, B, stored in arrays

Output: Rotation matrix $\mathbf{R}_{apx}$, translation vector $\mathbf{t}_{apx}$, permutation P_B, distance D

1: vectors :: $a_i \in A$, $b_i \in B$
2: ChooseCenter(A, $\mathbf{r}_c^a$) ▷ Geometric center
3: $A \leftarrow$ Shift(A, $\mathbf{r}_c^a$)
4: $A \leftarrow$ SortByNorm(A)
5: $\beta \leftarrow$ SelectBasis(a_i, a_j)
6: $A \leftarrow$ Rotate(A, β)
7: $\texttt{kmax} = max(|a_i|,\ |a_j|) * 1.2$
8: ChooseCenter(B, $\mathbf{r}_c^b$) ▷ Geometric center
9: $B \leftarrow$ Shift(B, $\mathbf{r}_c^b$)
10: $B \leftarrow$ SortByNorm(B)
11: **call** GETGAMMA$(A, B, \beta, \texttt{kmax}, \gamma, D)$
12: $\mathbf{R}_{apx} \leftarrow \gamma^{-1}\beta$
13: $\mathbf{t}_{apx} \leftarrow \mathbf{r}_c^b - \mathbf{R}_{apx}\mathbf{r}_c^a$
14: compute P_B from CShDA
15: **return** $\mathbf{R}_{apx}$, $\mathbf{t}_{apx}$, P_B, D

unlikely. The size of this distance matrix is $n_A \times n_B$, where n_A and n_B are the number of atoms in structures A and B respectively. As such, the distance matrix is generally neither square, nor symmetric. The Fig. 3.9 shows two structures A and B with a different number of atoms, containing atoms of two atomic species, blue and yellow. The indices i and j are marked on each structures with orange. A and B are such that all atoms are placed on a regular grid with the origin at atom index 3 in structure A, and at the vacancy in the center of structure B. The nearest-neighbour distance is 1.0. The all-to-all distance matrix for these two structures is shown at the bottom.

At second step, the assignments are computed according to Eq. (3.7), see Alg. 6 lines 12-30. Practically, the row index i of the distance matrix represents atoms a_i, while the column index j represents atoms b_j. The assignments $i \rightarrow j$ are then computed for each row i, as the index j of the lowest distance value in that row. However due to the one-to-one assignment constraint, each index j can be matched to only one index i. Meaning that if for some row index i, a column index j is selected with a value d_{ij}, and then for some other row index i', the same column index j is selected (Alg. 6 line 17) with a value $d_{i'j}$, the values d_{ij} and $d_{i'j}$ will be compared (Alg. 6 line 19), and the lowest one determines which

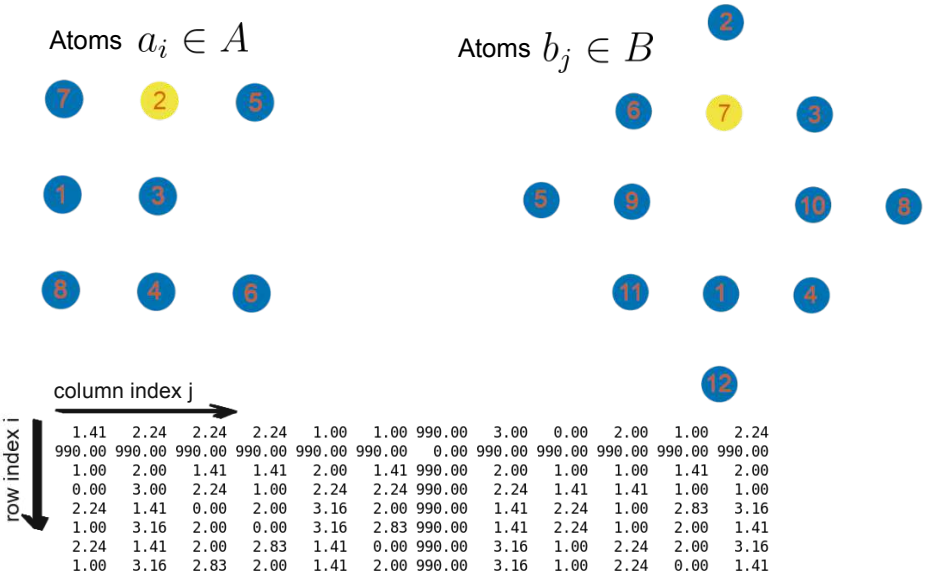


Figure 3.9: Structures A and B containing two atomic species: blue and yellow, with the atomic indices marked in orange. The all-to-all distance matrix at the bottom.

row index (i or i') will get assigned that column index j. The Fig. 3.10 shows the distance matrix for structures from Fig. 3.9, with entries circled in green and red. The green entries mark the final matching of indices $i \to j$, and the red ones mark the spots where CShDA had to search the index i again due to a conflict in assignment. The columns j that have been left unassigned are pointed by orange arrows, these represent atoms $b_j \in B$ which do not have a match in A, and are simply ignored.

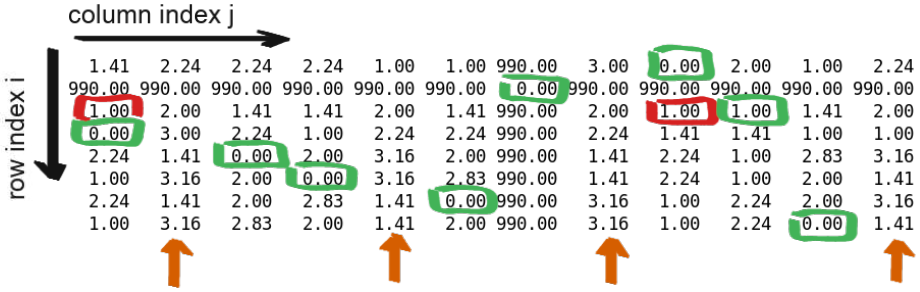


Figure 3.10: The assignment of distance matrix for structures from Fig. 3.9. Values circles in green mark final assignments $i \to j$, red circles mark the situations of conflicting assignments, and orange arrows mark unassigned column j.

The final assignments $i \to j$ of structures A and B, and their corresponding

distances d_{ij} are shown in Table 3.1. The value of distance function $D(A, B)$ used in the IRA algorithm is given by the largest value of the selected d_{ij} distances (Alg. 6 line 32). For the present example, $D(A, B) = 1.0$.

index i	index j	distance d_{ij}
1	9	0.0
2	7	0.0
3	10	1.0
4	1	0.0
5	3	0.0
6	4	0.0
7	6	0.0
8	11	0.0

Table 3.1: Final assignments $i \to j$ of structures A and B from Fig. 3.9.

The permutation matrix P_B which permutes the structure B is a $n_B \times n_B$-dimensional matrix, whose first n_A entries correspond to the indices found by CShDA, and the final $(n_B - n_A)$ entries are just on the diagonal, these last entries do not match any atoms from A (Alg. 6 line 31).

3.7.3 Standalone library

The shape matching algorithm containing IRA and CShDA is presently implemented as a standalone library of routines, written in Fortran. As such, it can be linked to any software. It is published as GitHub repository at `https://github.com/mammasmias/IterativeRotationsAssignments`, under a dual license: Apache v2.0, and GNU General Public License v3.0.

The main idea behind most routines is that the input structures should remain untouched. In language of Fortran, this means the atomic structures should have the attribute `intent(in)`. Anything returned by a routine thus has the attribute `intent(out)`. For example the IRA routine returns the rotation $\mathbf{R}_{apx}$ as a 3×3 rotation matrix, the translation $\mathbf{t}_{apx}$ as a 3-dimensional vector, the permutation P_B as n_B-dimensional array, and distance $D(A, B)$ as a real number, all these with the attribute `intent(out)`, and without actually applying them to a structure inside the routine itself. Giving the option to the program which called the routine, to use the information in any desired way.

A section of a hypothetical program, in which the shape matching problem is solved for two atomic structures `strucA` and `strucB` using the IRA algorithm, written as pseudocode would look as Alg. 7. Note that the structures are not transformed within the said program, the called routines leave them untouched.

Algorithm 6 The CShDA algorithm.

Input: Atomic structures A, B in arrays
Output: Distance $D(A, B)$, permutation P_B

vectors :: $a_i \in A$, $b_i \in B$
matrix :: d, dimension(n_A, n_B) ▷ Assignment matrix
list :: *found*, dimension(n_B) ▷ Contains the assignments
list :: *dists*, dimension(n_B) ▷ Contains assigned distances
list :: *search*, dimension(n_A) ▷ Search order
for $a_i \in A$ **do**
    for $b_j \in B$ **do**
        $d(i, j) = d(a_i, b_j)$ ▷ Euclidean distance
        if $\mathtt{typ}(a_i) \neq \mathtt{typ}(b_j)$ **then** $d(i, j) = 990.0$
    end for
end for
$i \leftarrow 1$
search(:) $\leftarrow 1$; *found(:)*$\leftarrow 0$; *dists(:)*$\leftarrow 0.0$
while *search(i)* > 0 **do**
    search(i)$\leftarrow 0$
    $j \leftarrow minloc(d(i, :))$ ▷ Location of the minimal value
    if j already in *found* **then** ▷ j is already assigned to some i
        $i\_old \leftarrow$ location of j in *found* ▷ which previous i
        if $d(i\_old, j) < d(i, j)$ **then**
            search(i)$\leftarrow 1$ ▷ Search this i again
            $d(i, j) \leftarrow 999.9$ ▷ Don't find this j again
        else
            search(i_old)$\leftarrow 1$ ▷ Search i_old again
            $d(i\_old, j) \leftarrow 999.9$ ▷ Don't find this j again
        end if
    end if
    found(i)$\leftarrow j$ ▷ assignment $i \rightarrow j$
    dists(i)$\leftarrow d(i, j)$ ▷ distance of assigned atoms
    $i \leftarrow$ next in *search(:)*
end while
fill the rest of *found* with indices of B without a match
return $D(A, B) = maxval($*dists(:)*$)$, $P_B =$ *found(:)*

3.8 Performance of the algorithm

The performance test of the IRA algorithm is done to examine its efficiency in finding the correct rigid transformation between two (near-)congruent structures.

Algorithm 7 A hypothetical program, solving the shape matching problem with IRA.

```
procedure MATCHSHAPE(strucA, strucB)
    # calculate rotation R_apx, translation t_apx,
    # permutation P, distance D:
    call IRA( strucA, strucB, R_apx, t_apx, P, D )

    # apply R_apx as matrix-vector multiplication, add t_apx
    strucA ← matmul( R_apx, strucA ) + t_apx

    # permute B
    strucB ← permute( strucB, P )

    # find optimal rotation R and translation t with SVD
    call SVD( strucA, strucB, R, t )

    # apply R, add t
    strucA ← matmul( R, strucA ) + t
end procedure
```

There are three specific scenarios to test.

The first is when two atomic structures are exactly congruent, they do not contain any distortions, and contain equal number of atoms. This scenario is tested in Sec. 3.8.1, which also describes the benchmark test of IRA against two other shape matching algorithms.

The second scenario is when two atomic structures are near-congruent, meaning they contain some atomic distortions, and have the same number of atoms. This scenario is tested in Sec. 3.8.2.

The third scenario is when two atomic structures are near-congruent, and have different number of atoms. This scenario is tested in Sec. 3.8.3.

The Sections 3.8.4 and 3.8.5 explore some details of the IRA algorithm. Namely how to mitigate some mismatches by providing more knowledge about the structures to the algorithm, in the form of known central atom, or any known common point in Sec. 3.8.4. And how to reduce the number of rotations tested in the iterative part of IRA in Sec. 3.8.5.

Furthermore, we test the particular scenario of IRA inserted into a kMC software in Sec. 3.8.6.

Note that the units of the $RMSD$ distance generally depend on the units in which the input structures are written. For the tests in Sections 3.8.1, 3.8.2, and 3.8.3, the units are irrelevant, since we are only interested in whether the distance

value equals to zero or not.

3.8.1 Exact congruence, equal number of atoms

In order to evaluate the efficiency of the IRA algorithm, we perform a test of finding the correct rigid transformation, consisting of $\mathbf{R}$, $\mathbf{t}$, and P_B, from any random initial rotation, reflection, translation, and permutation state. This is done as follows. An atomic structure A is copied into structure B, then B is operated on with a random rigid transformation consisting of rotation, reflection, translation, and permutation. The random rigid transformation is described in the next paragraph. The structures A and B are thus exactly congruent, contain equal number of atoms, and are related by some rigid transformation. The IRA algorithm is run for these two structures A and B. The transformation found by IRA is applied to B, and then the distance $RMSD(A, B)$ from Eq. (2.6) is calculated. If IRA has successfully found the correct transformation, the distance $RMSD(A, B)$ will be exactly zero, or within the floating point precision error. $RMSD(A, B)$ from Eq. (2.6) is a good distance measure for this case, since it is variant on all the rigid transformations applied, including permutation.

A random rotation is done by choosing a random rotation axis and a random rotation angle on the range [0,2π]. The rotation matrix corresponding to this operation is generated and applied to structure B. To randomly perform the reflection, a random number on the range [0,1] is drawn, if the number is greater than 0.5, then the current z-axis of the structure B is mirrored. Random translation is done by drawing random 3-dimensional vector, with a norm chosen randomly on the range [0,10]. The srtucture B is then translated by that vector. To randomly permute the indices, a valid random permutation matrix is generated, meaning that each index appears in it only once. The structure B is permuted with this permutation matrix.

The random transformation test has been done a number of times for each structure, to gather some statistics of the algorithm efficiency when starting from different initial states. Two sets of structures A used for this test are from the Cambridge Cluster Database [117], more specifically we have used the TIP4P water clusters with $n = 2$ to $n = 21$ molecules of water in the cluster, the Lennard-Jones (LJ) clusters of sizes $n = 3$ to $n = 150$ and from $n = 310$ to $n = 1000$ atoms. We have also used an amorphous Si structure with $n = 64$ atoms. Some sample structures are shown in Fig. 3.11. The test is done 10000 times for each of the water cluster structures, 100 times for each LJ structure, and 10000 times for Si structure. The result of every single one of the tests was a distance below the floating point precision error, which implies that the correct transformation has been found in all the cases.

For the purpose of the IRA article [114], we have also performed a comparison

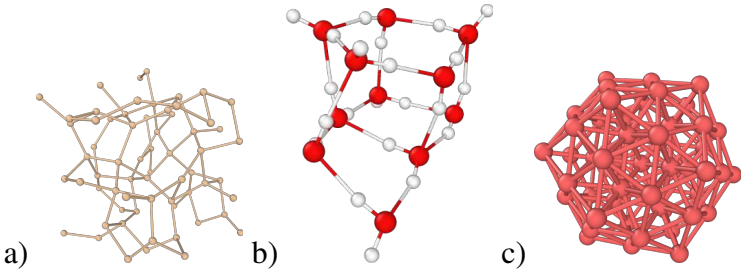


Figure 3.11: Some sample structures used to test the reliability of the overall algorithm: a) amorphous bulk silicon, b) $n = 11$ TIP4P water cluster, and c) $n = 52$ Lennard-Jones cluster.

of the efficiency to two other software, namely the ArbAlign from Ref. [113], and fastoverlap from Ref. [87]. This has been done with the same procedure of randomizing a structure, and then trying to re-find that transformation, as in the previous paragraph. The structures chosen for the comparison were such that they represent different possible shapes, namely isotropic ones, such as compact, spherically, or cylindrically shaped clusters, as well as non-isotropic ones, such as water clusters and peptide molecules. The reasoning behind this choice is to show that the efficieny of IRA is independent of the structure shape. This statement is not trivially true for the other two algorithms in this test. The datasets chosen are the following.

Datasets used:

- From the Supplementary Materials of Ref. [113], Neon clusters with number of atoms $n = 10$, $n = 50$, $n = 100$, $n = 150$, $n = 200$, $n = 300$, $n = 500$, and $n = 1000$, each with two distinct configurations, thus making $N_s = 16$ structures altoghether;

- From the Supplementary Materials of Ref. [113], water clusters with number of water molecules $n = 2$ to $n = 21$, $n = 25$, $n = 40$, and $n = 60$, each in a number of different configurations, making $N_s = 70$ configurations altogether;

- From the Supplementary Materials of Ref. [113], conformers of FGG peptide, with $n = 37$ atoms of 4 different atomic types, in $N_s = 15$ different configurations;

- From the Supplementary Materials of Ref. [113], hydrates S1-MA-W1 with $n = 17$ atoms of 5 different atomic types, in $N_s = 20$ different configurations;

- From the Supplementary Materials of Ref. [118], Al clusters of number of atoms between $n = 63$ and $n = 160$, in steps of 1, and from $n = 160$ to $n = 310$ in steps of 5 or 10, making $N_s = 93$ structures altogether;
- From the Supplementary Materials of Ref. [119], GaN clusters with number atoms between $n = 12$ and $n = 96$, in steps of 2 or 4, making $N_s = 31$ structures altogether;
- From the Supplementary Materials of Ref. [120], clusters of Au with a total of $n = 26$ atoms, with a varying number of atoms of a different specie, making $N_s = 6$ structures altogether.
- From the Cambridge Cluster Database [117], Lennard-Jones clusters of number of atoms from $n = 5$ to $n = 150$ and from $n = 310$ to $n = 520$, making $N_s = 357$ structures altogether.

A representative structure from each dataset is shown in Fig. 3.12. Note the diversity of shapes of the structures used.

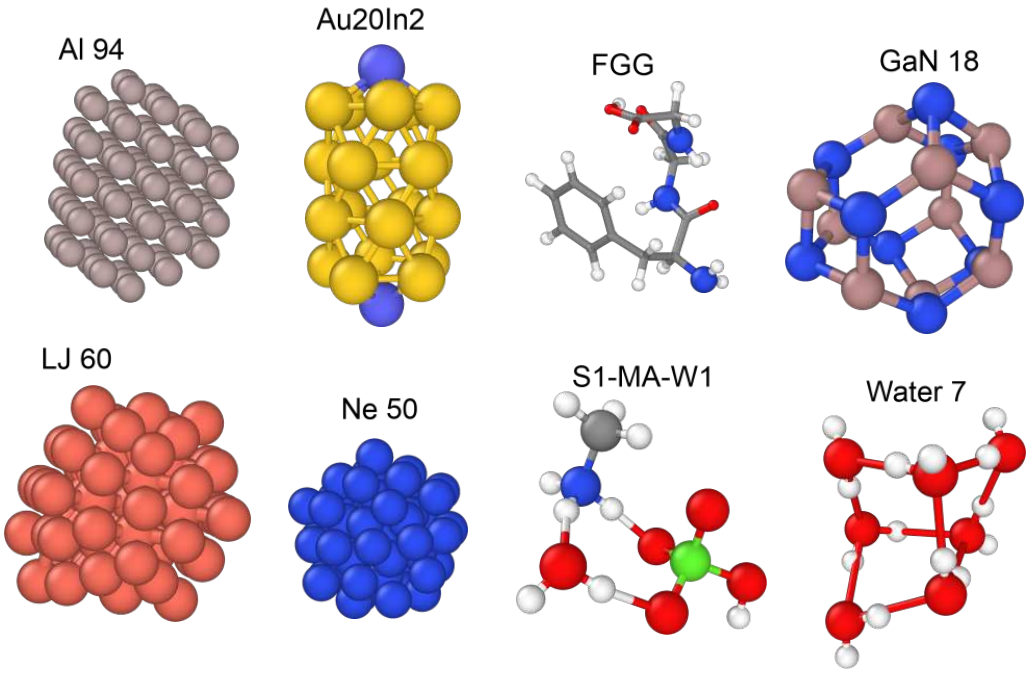


Figure 3.12: Representative structure of each dataset used, note the diversity in the general shape of structures.

A random transformation was done 50 times for each structure. The succes of each trial is given by the value of the final $RMSD(A, B)$, if it is above

the threshold of 0.001, then the trial is deemed as failure, since the correct rigid transformation should give distance zero. This threshold was chosen as the least common precision in the default output of all three codes, which is three decimal places.

The results of the test are given in Table 3.2, and Fig. 3.13. The values in Table 3.2 are given in the form m/n, where m is the total number of failures, and n is the number of structures in which the failures occured. The plots in Fig. 3.13

Dataset	N_s	ArbAlign [113]	fastoverlap [87]	IRA
Al [118]	93	0/0	613/34	0/0
Au26 [120]	6	186/4	*0/0	0/0
FGG [113]	15	0/0	*0/0	0/0
GaN [119]	31	50/1	*294/14	0/0
LJ [117]	357	45/1	1177/113	0/0
Neon [113]	16	100/2	82/8	0/0
S1MAW1 [113]	20	0/0	*0/0	0/0
water [113]	70	0/0	*217/11	0/0

Table 3.2: Results of the efficiency test of the three algorithms. Each dataset is referred to by its name, N_s is the number of different structures in each dataset. Each structure from each dataset was tested with 50 random initial transformations. The tabulated values are in the form m/n, where m is the total number of failures, and n is the number of structures in which the failures occur. Values marked with *: the structures in this dataset include several atomic types, which fastoverlap cannot distinguish.

report the final values of $RMSD$ for cases when the matching has failed. The horizontal axis on the plots gives the name of the particular structure where a failure has occurred, the vertical axis is the number of current trial, the color of a point gives the final value $RMSD$, and the shape of a point is related to the particular software which returned the failure.

Before commenting the results, brief descriptions of the ArbAlign [113], and fastovelap [87] algorithms are in order.

The algorithm ArbAlign [113] uses similar ideas as the well-known algorithm Iterative Closest Point ICP [108]. ICP consists of iterative steps combining an assignment procedure, and a consecutive rotation procedure in each step, until a solution is found in a self-consistent way. However, the algorithm ArbAlign proposes only one such step. It is well known that ICP can remain trapped in local minima [109]. This trapping can be mitigated in several ways, one of them is finding a good-enough initial rotation and permutation state. The original article of the ArbAlign algorithm never mentions local minima explicitly. The first step of Ar-

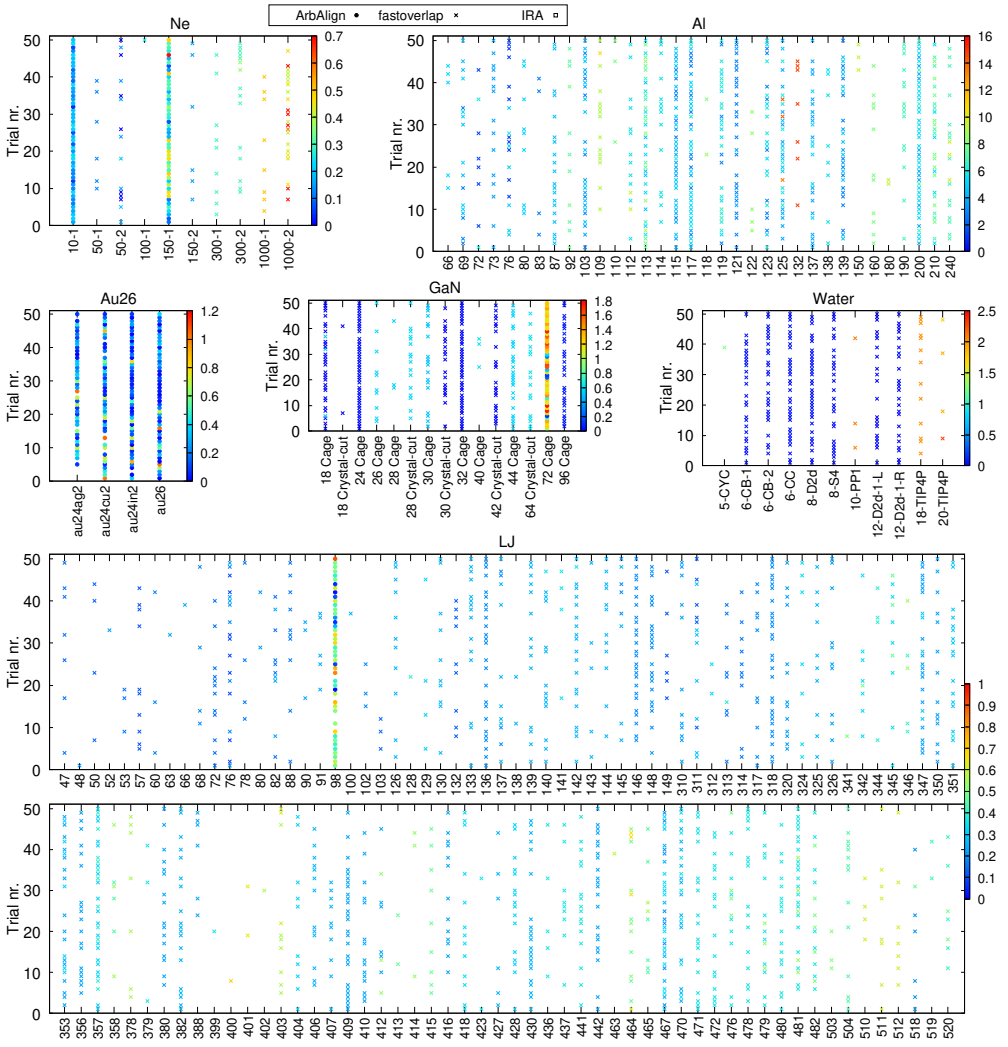


Figure 3.13: The final values of $RMSD$ for the cases of failure, for structures and datasets where failures have occured. The color of each point gives the value of $RMSD$, the horizontal axis gives the name of the structure, the vertical axes is the number of current trial (50 trials per structure), the results coming from different algorithms are shown by the shape of points: ArbAlign by circles, fastoverlap by crosses. IRA did not fail even once.

bAlign is however to rotate the structures into their principal axes of inertia, which can be understood as the mitigation strategy just mentioned. The algorithm then attempts 48 pre-defined symmetry operations, and does the one-step assignment-rotation procedure for each attempt. As briefly commented in the Sec. 3.1.2, principal axes of inertia are problematic when the structures are isotropic or compact, since the principal axes might be ambiguous due to the symmetry of the struc-

tures, which might result in the subsequent assignment-rotations procedure to get trapped in local minima, no matter the 48 symmetry attempts. Moreover, the computation of principal axes of inertia requires the knowledge of associated weights, i.e. atomic masses. To solve the LAP, ArbAlign uses the Hungarian algorithm [81], and to optimize the rotation it uses the SVD-based approach by Kabsch [99].

The algorithm fastoverlap [87] is based on kernel correlation. It uses Fourier transform to find maximum correlation between density representations of both structures. As such, it needs some input parameters, namely the width of Gaussian kernels, and the angular resolution defined by an angular momentum cutoff. Moreover, fastoverlap does not distinguish different atomic types. The underlying philosophy of solving the matching problem is slightly different than what is used in the IRA and ArbAlign algorithms. For the test performed here we use the default parameters set in the fastoverlap software.

From the results of our benchmark test (Table 3.2), we can conclude the following. The algorithm ArbAlign [113] has problems to find the correct rigid transformation in structures where the principal axes of inertia are ambiguous. This is very clear from the Au26 dataset from Ref. [120], which includes cylindrical shape structures, where the axis along the cylinder is well defined, however the other two are not. We note that since each structure was tried 50 times, the result of 186 failures in 4 structures (see Table 3.2) indicates that on average, there were 46 failed attempts out of 50 trials per structure. Similarly also for the other datasets, there are a high number of failures for few structures in the dataset. This indicates that ArbAlign mostly works well, however in cases of failure, it fails very consistently. This is also seen by the final distance values obtained in structures where it fails (Fig. 3.13), which are distributed quite randomly. The origin of this dispersion of values is in my opinion the huge number of local minima present due to structure symmetry, which ArbAlign effectively walks into with its one-step procedure.

On the other hand, the algorithm fastoverlap [87] shows a higher overall rate of failure, but its failures are more disperse, in the sense that we do not observe such stark contrast between complete success and complete failure, as in ArbAlign. Moreover, the final values of distance from fastoverlap (Fig. 3.13) show clustering around several distinct values, which might be a signature of the algorithm getting stuck in a small number of distinct local minima.

Our proposed IRA algorithm does not show any failure among any of the structures tested. We can say with high confidence that it is fully reliable at finding any rigid transformation between two congruent structures.

3.8.2 Near congruence, equal number of atoms

To test the performance under conditions of near congruence, i.e. the structures contain some deformations - we first perform a short NVT-ensemble Monte Carlo (MC) simulation for a LJ-20 cluster from the Cambridge Database [117] at two different temperatures. The specific temperatures used are $T = 0.02$ and $T = 0.3$ in the reduced units. These two values have been chosen as corresponding to "low" and "a bit higher", and are only used to induce some atomic vibration.

We take the equilibrium configuration of the LJ-20 cluster as reference structure A. At each step of the MC simulation, the current structure is taken as B, and the distance $RMSD_{ini} = RMSD(A, B)$ is calculated. During the MC, the structure undergoes some distortion, translation, and rotation, but not permutation of atoms. We can thus readily apply the SVD method to obtain rotation that minimizes $RMSD(A, B)$ at current step, and store this $RMSD$ value as $RMSD_{ref}$. Then we apply random rotation, reflection, translation, and permutation to structure B, and run our shape matching algorithm on it, to obtain B' aligned to A, and calculate distance $RMSD_{fin} = RMSD(A, B')$. The distance $RMSD_{fin}$ should be equal to $RMSD_{ref}$ if our algorithm has successfully found the right transformation. The results are shown on Fig. 3.14. The difference $RMSD_{ref} - RMSD_{fin}$ on every step is on the order of floating point precision error (*i.e.,* zero), for both test temperatures, confirming the ability of the IRA algorithm to find the correct matching transformation efficiently.

The non-zero value of $RMSD_{fin}$, can be seen as a measure of the congruence between the structures.

3.8.3 Near congruence, different number of atoms

To test the algorithm in case of near congruence, where the structures have a different number of atoms, we use a trajectory of replica-exchange molecular dynamics simulation of the cynanine molecule. The data has been provided by the authors of Ref. [121]. In order to showcase the ability and performances of the IRA and CShDA algorithms in matching structural fragments, we select two kinds of fragments, a connected one shown in Fig. 3.15, and a non-connected one shown in Fig. 3.16.

The atoms of the molecule move and distort the molecule during the trajectory, but they do not permute. Thus, we can make a similar test for reliability as in the previous subsection. We choose as structure A a reference fragment, and compute the optimal rotation of molecule B using SVD, giving $RMSD_{ref} = RMSD(A, B)$, where the sum in $RMSD$ goes up to n_A number of atoms, over the known atomic indices that make up structure A. Then we randomly rotate, reflect, translate, and permute structure B, and then use our shape match-

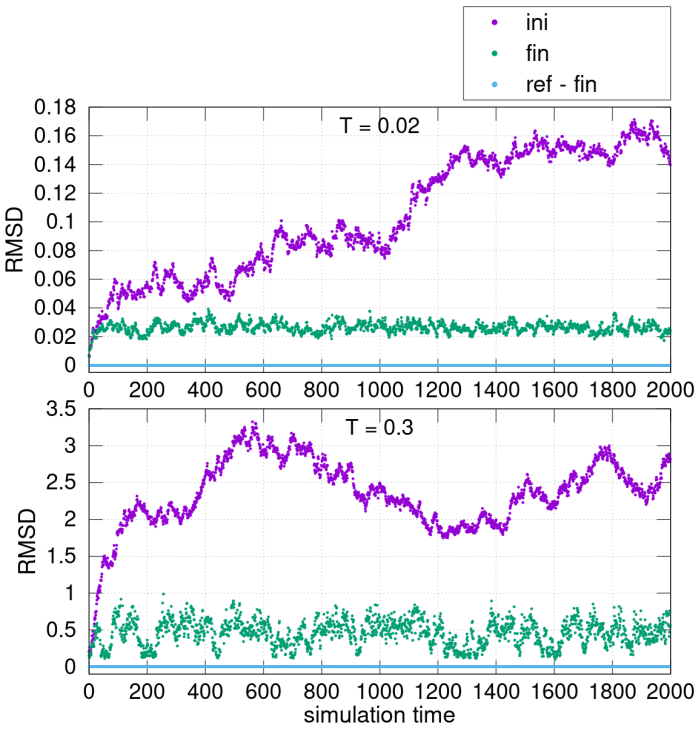


Figure 3.14: Result of the efficiency test of the IRA algorithm for near-congruent structures. Plot of $RMSD_{ini}$, $RMSD_{fin}$, and the difference $RMSD_{ref} - RMSD_{fin}$ for temperatures (top) $T = 0.02$, and (bottom) $T = 0.3$.

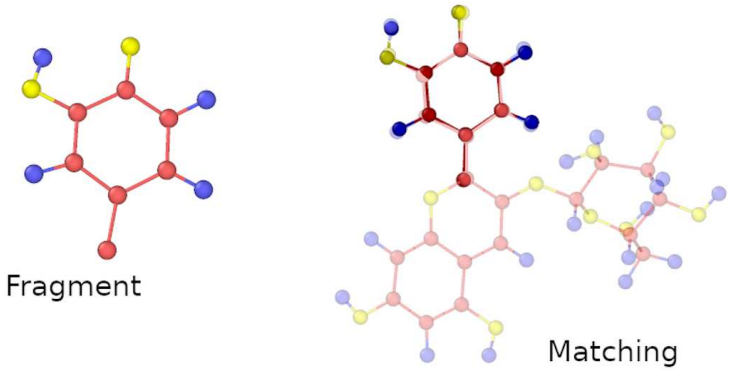


Figure 3.15: The fragment to be matched, and the final matching of the molecule, the atoms of the fragment are shown with a darker shade for better distinction. Red, blue and yellow atoms correspond to Carbon, Hydrogen and Oxygen atoms respectively, the same color code is used in the following.

ing algorithm to obtain B' aligned to fragment A, and calculate $RMSD_{fin} = RMSD(A, B')$. The distances $RMSD_{ref}$ and $RMSD_{fin}$ should be equal if the

right transformation has successfully been found.

The result when structure A is the connected fragment from Fig. 3.15, is that out of the 80000 configurations in the trajectory, there are 313 instances of the difference $RMSD_{ref} - RMSD_{fin}$ being above the floating point precision value, *i.e. nonzero*. These instances represent structures where the algorithm has mismatched the fragment. Some of the reasons for this behaviour are explored later in this section.

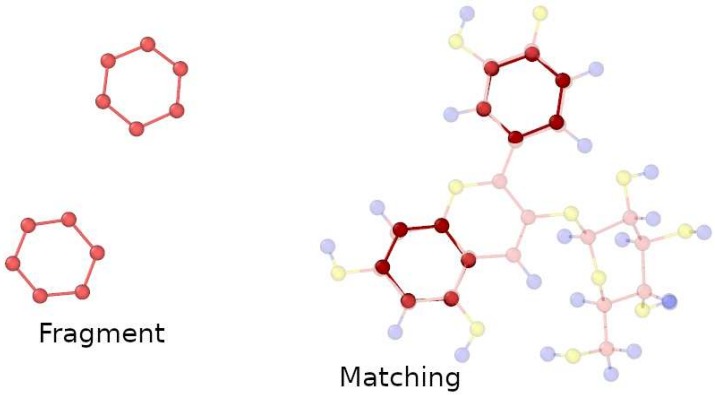


Figure 3.16: A disconnected fragment, and matching of a molecule.

Tracking the number of mismatches when structure A is the non-connected fragment from Fig. 3.16 is not straightforward, since the two hexagons do not move rigidly. As a consequence, $RMSD_{ref}$ as defined previously is ambiguous. One could think to trace the particular steps in the trajectory where the two hexagons are found with a distance below some threshold, and call those particular configurations equivalent to the fragment. But actually there is no clear step in the distance values in this case, the values are quite continuous from small to large, so it is not straightforward to set such threshold.

Since the value of h in Eq. (2.10) only takes the value of maximal distance, it only contains information about one specific atom/point. This particularity can be advantageous in cases of small distortions between two structures. In that situation, the value of h is low, meaning that all atoms are within this low-distance h of the reference positions. Larger distortions lead to higher h value, which can hide the behavior of any specific atom. A high h value can be due to a distortion in a single atom, which completely obscures any information on other atoms. This property of the Hausdorff distance is often described as high susceptibility to noise. It opens the possibility of a situation in our algorithm, where a "wrong" assignment gives a transformation U_J^{-1} whose distance $D(A_{\{\hat{\mathbf{e}}\}}, B_{\{\hat{\mathbf{e}}'\}_J})$ is lower than the distance D when the transformation is given by the "correct" assignment, which then leads to a wrong final assignment and transformation. In other words,

the algorithm finds a way to map the atoms to the reference structure which satisfies the algorithm, but is not actually the mapping we would expect. To mitigate this, we tried to replace the $h(A, B)$ with a sum of minimal distances $d_{sm}(A, B)$, given by Eq. (3.16), which is similar to $h(A, B)$ from Eq. (2.10), but d_{sm} takes the sum of all minimum values, instead of just the maximal value as in h.

$$d_{sm}(A, B) = \sum_{a \in A} \min_{b \in B} d(a, b). \tag{3.16}$$

The distance d_{sm} should capture a more "collective" behaviour of the atoms, but it has not shown any significant changes in the performances with the highly distorted cyanine molecule tests. The mismatches still happen at large set-set distance values. The choice of a particular set-set distance function therefore does not seem very crucial, as long as it complies with the permutational invariance, and translational and rotational variance, imposed by Eq. (3.2). The "mismatches" are rather due to attempting to match structures that are beyond near-congruence, or in other words, far from equivalence. Which ultimately brings us back to the question of comparing oranges, potatoes, and bananas from Fig. 2.1. In such a situation, a careful interpretation of the matching transformation obtained from the algorithm is needed. It is true that the final transformation comes from a well-defined mathematical operation, but this does not mean that it is always meaningful.

3.8.4 Mitigating the mismatches

By assuming some prior knowledge on the system, it is possible to reduce the number of mismatches that happen in cases of larger distortions in the structures. One piece of such prior knowledge is the knowledge of a definitive common point for the two structures, which could be in form of a known common atom, deemed central atom.

The first step of our IRA algorithm (Sec. 3.5) selects a central atom in structure A by the criteria of closeness to the geometrical center of A. The second step is to select a basis $\{\hat{\mathbf{e}}\}$ for a reference frame in A, which is based on positions of atoms around the central atom. Then the structure B is searched for the equivalent basis $\{\hat{\mathbf{e}}'\}_J$. When large distortions are present in structure B, there is no guarantee that the basis found in B is equivalent to the basis found in A, or that it even exists. If we assume that there still exist local environments in the two structures that are congruent to each other, then the central atom of A could be chosen as the atom for which its local environment is the most similar to any local environment in B. Imposing the central atom in A according to that criterion in the case of cyanine for instance, reduces the number of mismatches by an order of magnitude (313

originally, 30 with this choice). The imposed central atom for structure A has been chosen by hand in the present example, however ideas for automatizing this could be to use simple structure descriptors on short-range (first-neighbour) local environments, such as the ones described in Sec. 2.2.

The implementation of IRA in our kMC code (see Sec. 4.4.1) uses graph isomorphism of local environments as additional per-atom knowledge of the structures, which can be seen as a filtering method on possible central atoms.

3.8.5 Number of rotations tested

As already mentioned in Sec. 3.7.1, the total number of rotations tested is greatly dependent on the structure. Fig. 3.17 shows the number of rotations N_R tested by IRA during the matching of the structure with an exactly congruent randomized version of itself as done in Sec. 3.8.1, against the number of atoms in the structure, for the Al dataset[118]. The Al dataset, along with LJ and Ne datasets from the benchmark test in Sec. 3.8.1, represents a worst-case scenario for IRA as all atoms are of the same atomic type, and the structures are close-packed. As it can be seen on Fig. 3.17, the number of rotations tested is on the range [2, 154] and there is no apparent rule. The number of tested reference frames is related to the structure surrounding the origin point as mentioned in Sec. 3.7.1, which in the case of non-equal number of atoms is a central atom, and in the case of equal number of atoms is the geometrical center (or possibly any known common point). The higher number of tested rotations occurs when the geometrical center of the structure coincides with an atomic position. In that case, the distance to nearest atoms is the highest. A large number of atoms is therefore included in the radial cutoff region, such increasing the number of possible reference frames to be tested. When the geometrical center falls in between atoms, the distance to nearest neighbors is shorter (lower number of atoms), and thus less reference frames have to be tested. In the case of matching structures with non-equal number of atoms, the origin point of rotation is a central atom. In that case the number of possible rotations N_R is directly linked to how many neighbours the current central atom has, as described in Sec. 3.7.1, and shown on Fig. 3.8. The total number of rotations tested in the scenario of non-equal number of atoms would then be the number of rotations N_R per possible central atom, times the number of possible central atoms. We see that in this case the computational cost can increase fast, with the number of possible central atoms. It can thus become very beneficial to spend some computational effort in reducing the set of possible central atoms, using any method that is deemed effective. This gives another reason for the discussion in Sec. 3.8.4.

In situations when we know that the two structures being matched are sufficiently similar, the multiplication factor 1.2, used for the cutoff can be reduced,

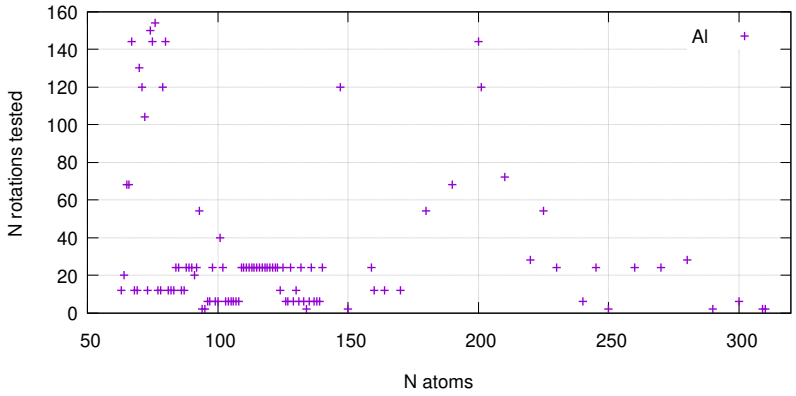


Figure 3.17: Number of rotations tested versus the number of atoms, for structures in the Al dataset [118].

but the value should in any case remain above 1.0. This effectively reduces the search space of rotations, and the algorithm can be faster as a result. The speedup comes at the cost of possible mismatches when structures matched are actually not as similar as we thought.

In situations where the equality of two structures is being tested with a certain known threshold for equality, heuristic approaches can be used on top of the logic of the IRA and CShDA algorithms, to exit certain loops as soon as certain criteria are met. This method has the potential to speed up the algorithm considerably, however at the cost of generality.

3.8.6 A specific situation: the central atom is known

In this subsection, we test the efficiency of IRA in a specific situation where the structures A and B contain a different number of atoms, however the central atom is known a priori. The main effort of this test is to increasingly distort the structure A, and find at which point IRA fails to find the correct transformation. The value of the largest deformation at which IRA still finds the correct transformation can set the maximal reference value for a threshold of equivalence when comparing two structures.

The particular efficiency test is then as follows. We take a simulation box of 217 atoms of Si with a single interstitial atom as structure B, and a local environment around the interstitial atom as structure A with 27 atoms. The typical first neighbour Si-Si distance in both structures is 2.34 Å. We know exactly which atomic index is the central atom in A, and which is the atomic index of central atom in B, namely it is the interstitial atom in both cases. Then we perform some deformations on the structure A, and attempt matching it to structure B. The eval-

uation of the final matching will be evaluated for each case separately, in terms of what the random deformation is, and the value of h after the matching. The structures A and B are shown on Fig. 3.18.

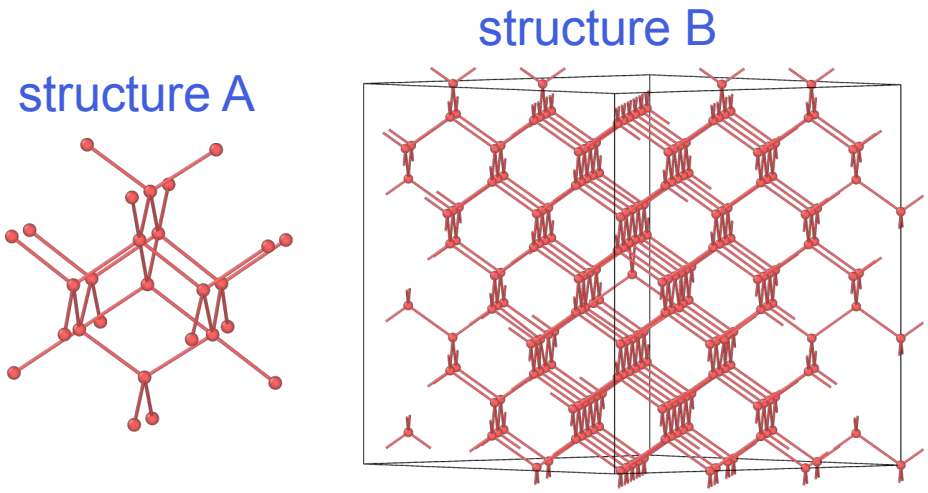


Figure 3.18: Example structure A on the left, and B on the right, used in the test. The interstitial atom is at the center of both structures.

The deformations on A are performed as randomly generated displacement vectors, with random norm in the range [0, d_{max}], which are applied to N_d number of atoms, chosen at random. The structure A is then also randomized, with a random rigid transformation including rotation, reflection, translation, and permutation. After that, IRA is used to match A to B with a known central atom. If the correct transformation is found, the final value of h should be on the order of d_{max}. The expectation is that if the number of displaced atoms N_d is small, for example 5, then the maximal distortion d_{max} can get quite high and IRA will still be able to find the correct matching. On the other hand, if N_d is higher, for example $N_d = 27$, which displaces all atoms of A , then there might be mismatches for lower d_{max} values. In order to test this, we launch 500 such tests for each combination of N_d and d_{max} values, where $N_d = \{5, 10, 20, 27\}$ and $d_{max} = \{0.1, 0.3, 0.5\}$ Å. The first mismatch happens for $N_d = 5$ and $d_{max} = 0.5$ Å, the final matching is shown on Fig. 3.19, with a successful match for comparison. It is worth noting that the distortion of $d_{max} = 0.5$ Å actually breaks the graph isomorphism for the central atom. Subsequent mismatches happen 3 times at $N_d = 10$ and $d_{max} = 0.5$ Å, 11 times at $N_d = 20$ and $d_{max} = 0.5$ Å, and 25 times at $N_d = 27$ and $d_{max} = 0.5$ Å. There are no mismatches for lower d_{max} values. One could conclude that for this example, a threshold distance for h of around 0.3 Å could represent a meaningful value for structural comparison,

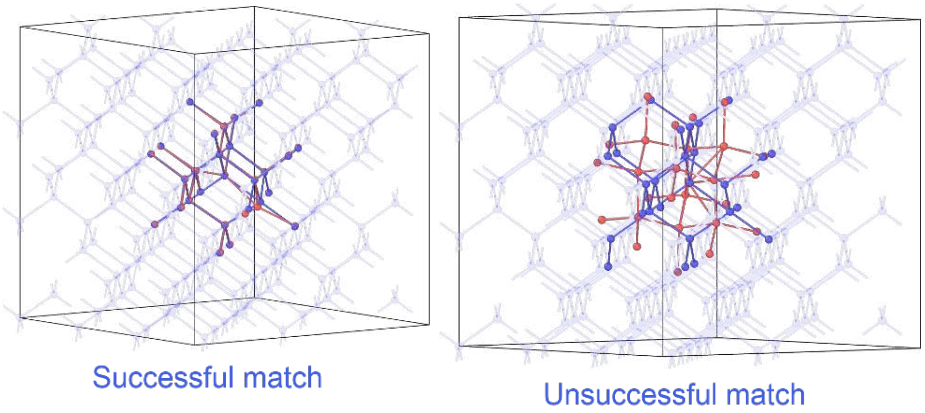


Figure 3.19: A mismatch at $N_d = 5$ and $d_{max} = 0.5$ Å on the right, and a successful match for reference on the left. The structure A is represented by red atoms, the structure B by the blue atoms, where the atoms of B that are not matched are shown in transparent.

since we observe that distortions as large as $d_{max} = 0.3$ Å do not cause IRA algorithm to find "alternative" solutions for the matching. Whether that particular value represents anything meaningful in relation to the atomic distortions in the particular example, or if there would be a way to come to this value from any other potentially simpler test, has not been investigated.

From a more empirical inspection of the results of this test, I can say that the probability of a mismatch is higher when the displaced atoms are coincidentaly the ones which set the basis β in structure A. This observation can be connected with the Sec. 3.8.4, in which the knowledge of a central atom is discussed as a strategy to mitigate mismatches. Here, we could say that a further strategy of mismatch mitigation could be to input the knowledge on particular atoms that move the least from one structure to the other, and use those to set the basis. This is however slightly paradoxical, since the atomic information of this kind is available only after the matching procedure.

Chapter 4

Implementation of shape matching into kMC

A fundamental process within the kMC simulation is the association of possible events from the event catalog to the atomic sites within the system of simulation, throught the IDENTIFY_POSSIBLE_EVENTS() routine in Alg. 1 line 3. In order for an event to be deemed possible for execution, three things need to be known. Firstly, the local environment defined around an atomic site in the system of simulation must be sufficiently similar (or equal) to the initial state configuration of a given event. Secondly, the rigid transformation which satisfies the equivalence of these two configurations must be found. And thirdly, each of the possible symmetries of the event must be checked if they exist in the local environment of the system, to grant the correct statistics of the move directions. These three things are used later at the stage of event application, APPLY_EVENT() routine in Alg. 1 line 5.

As has been shown in Chapter 3, all of these tasks can be done simultanously in an efficient way with the IRA shape matching algorithm. Throughout the Sections of the present chapter, the details of the implementation of that algorithm inside our kMC simulation code are given. In the final Section 4.6, we discuss some ideas that are used for the implementation.

4.1 Program workflow

The general idea for this kMC is to be able to read the ensemble of given events from input, in the form of atomic coordinates. In order to pass it to our kMC, the event data is first rewritten into an event catalogue. The event catalogue is then read by the kMC code at the initial stage of the simulation. As such, we can think of the whole procedure as composed of two stages, the generation of the event

catalogue, and the kMC simulation. This process is shown by the schematic in Fig. 4.1.

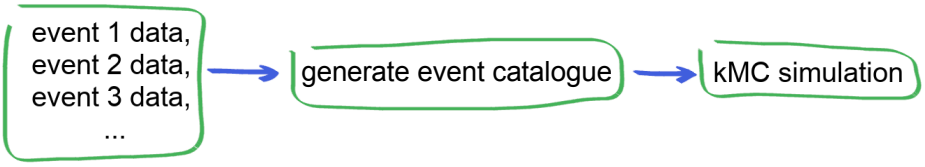


Figure 4.1: Schematic of the two-stage process: creation of the event catalogue, and the kMC simulation.

Within the kMC simulation, atomic structures from the system of simulation get compared to the atomic structures of initial states of events, that are written in the event catalogue. This is done in two steps.

The first step is a fast pre-screening of the possible event sites in the system of simulation, by isomorphism of simple graphs using their canonical hash values, computed by the NAUTY [73] software. In order to do this step more efficiently, the graph hash values of each event configuration are pre-computed and stored in the event catalogue, while the hash values of local environments of system sites are computed on-the-fly in the kMC simulation.

The second step of the comparison of atomic structures is done by our IRA shape matching algorithm. IRA itself can be seen as a two-step process when matching structures A and B. From this point of view, the setting of reference frame β on structure A is the first step, while the iteration to find γ reference frame on structure B and subsequent SVD minimization is the second step of the shape matching algorithm. The two steps of IRA matching are independent, in the sense that once β is known, it can be stored in the memory quite simply as three integers. The structures that are matched in the kMC are: the structure A that is the initial state configuration of an event that is stored in the events catalogue, and structure B that is the local environment around an atomic site in the system of simulation. Absolutely no knowledge of structure B is needed to complete the first step of the IRA algorithm, thus the algorithm is split into two parts accordingly. The first part is done in the event catalogue generation, acting only on structure A, and the second part is done when knowledge of structures B is given (in the kMC simulation).

In the Section 4.2, some preliminary definitions and explanations are given, regarding atomic environments. The Sections 4.3 and 4.4 describe the details of the two stages of the program, namely the generation of event catalogue, and the kMC simulation. Their contents are listed in Table 4.1, with references to the algorithms that they interact with. The main connection between the catalogue of events and the kMC simulation is at stages of identification of possible events,

and application of a chosen event. In the general kMC algorithm (Alg. 1), these are the procedures of the structural aspect of the kMC: IDENTIFY_POSSIBLE_-EVENTS(), and APPLY_EVENT().

Table 4.1: The two stages of the program, and the different algorithms they interact with.

Event catalogue:		Sec. 4.3
select type of event, and local environment:		Sec. 4.3.1, 4.3.2, 4.3.3
generate graph hash values:		Sec. 4.3.4
IRA first part: generate β and symmetries θ:	Alg. 8	Sec. 4.3.5
final format:		Sec. 4.3.6

kMC:		Sec. 4.4
Identify possible events:	Alg. 1, line 3	
- check graph hash values:		Sec. 4.4.1
- IRA second part: generate γ, check all θ:	Alg. 5	Sec. 4.4.2, 4.4.3
apply event:	Alg. 1, line 5	Sec. 4.4.4
local update:		Sec. 4.4.5

4.2 Atomic environments

4.2.1 System of simulation

Any configuration containing the full set of atomic positions present in the simulation box shall be designated with the letter R. The system of simulation is designated R^{sys}. It is composed of N atoms, each with its xyz-coordinates, and associated atomic type.

$$R^{sys} = \{typ_i,\ x_i,\ y_i,\ z_i\}$$

4.2.2 Central atom

In the kMC simulation, the idea of central atoms is used. The central atom of an event is placed at the origin of the reference frame of atomic configuration. It is the atom around which an event will happen.

At the start of kMC simulation, any atom in the system is a potential central atom for an event to occur. In the first step of the simulation, all sites are parsed, but only those where an event is deemed possible are potential event sites. In the subsequent steps, only a subset of the sites are checked, as given by the local update (see Sec. 4.4.5).

4.2.3 Local atomic configuration

A local atomic configuration, designated S, is any un-ordered subset of system configuration R^{sys},

$$S \subset R^{sys};\ S = \{typ_k,\ x_k,\ y_k,\ z_k\}$$

where typ_k, is the atomic type of atom index k, and x_k, y_k, z_k are its xyz coordinates. $\{k\}$ is the list of indexes invloved in the local configuration. The list $\{k\}$ is typically found such that all atoms k satisfy a certain condition. In our kMC, this condition is the distance from a designated central atom being below some pre-defined cutoff. An example local atomic configuration S is shown in Fig. 4.2.

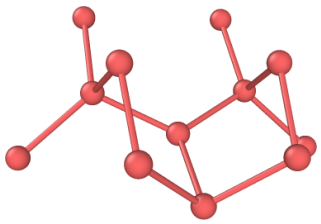


Figure 4.2: Example local atomic configuration S.

Atomic events in the kMC are composed of two states: the initial state, and the final state. The full configuration of the input data of the event initial state is labelled as R^{evt}_{ini}, and similarly for the final state R^{evt}_{fin}. The local atomic configurations corresponding to these states are labelled S_{ini} and S_{fin}, such that:

$$S_{ini} \subset R^{evt}_{ini};\ S_{ini} = \{typ_k,\ x_k,\ y_k,\ z_k\} \tag{4.1}$$

and

$$S_{fin} \subset R^{evt}_{fin};\ S_{fin} = \{typ_k,\ x_k,\ y_k,\ z_k\}. \tag{4.2}$$

4.2.4 Extended local configuration

To make atomic configurations more flexible, we define the extended local configuration S^E, which does not have a fixed size, and is not necessarily isotropic.

Any local atomic configuration S is called an extended local configuration S^E when it is made of two parts: primary part, and extension. The primary part is the local atomic configuration S from Sec. 4.2.3. The extension is made up of atoms that follow some other imposed rule.

An atom of the extension can be any atom which is not automatically included in the local atomic configuration S around a central atom, but it satisfies one or more of the three conditions:

- the atom is not included in the local environment around the central atom in the initial state S_{ini}, but is included in the final state S_{fin} of the event;
- during the event, the atom moves for a distance larger than some threshold;
- the atom is included manually by the user.

Fig. 4.3 shows an example extended environment S^E where the primary part is colored in green and the extension in yellow.

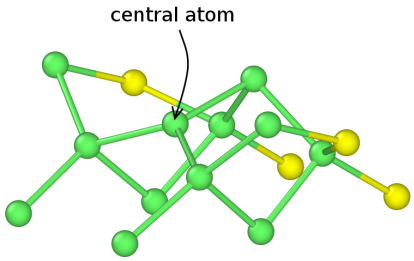


Figure 4.3: Example extended environment S^E, primary part colored in green, extension in yellow. The yellow atoms of extension are found by the three conditions.

4.3 Generation of event catalogue

Generation of the event catalogue (also called event library) is done by the program `generate_library.x`, which needs two input files, see Fig. 4.4. One

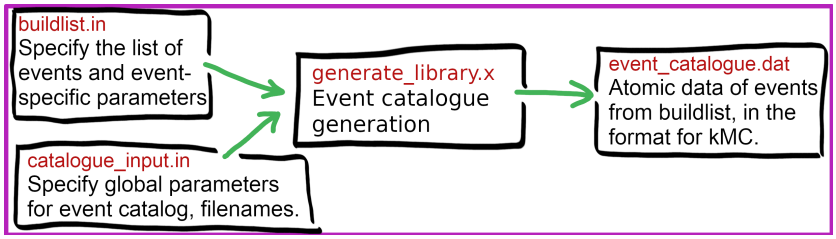


Figure 4.4: Schematic of the creation of the event catalogue.

that is generically called `catalogue_input.in`, and one called `buildlist.in`. Actually, only the `catalogue_input.in` file is given as standard input to `generate_library.x`, and it contains the exact filename of the `buildlist.in` file, and the exact filename of the final event catalogue to be written in. It also

```
events_filename = buildlist.in
library_filename = event_catalogue.dat

rcut_mode = neig
small_move_thr = 0.05
dist_thr = 0.5
ntyp = 1

at_typ
1 Si

color_cutoff
1 1 2.8
```

Figure 4.5: Example of `catalogue_input.in` file.

contains some other parameters used globally during the generation of the events catalogue. An example is shown on Fig. 4.5.

Due to our idea that kMC events should be given at input, the expected format of the events is an output calculation of some other computational method, for example a first-principles code. The file `buildlist.in` contains data entries specific to each event. Each data entry contains:

- the value of cutoff `rcut`;
- the data related to the energetics of the event:
  - the prefactor `f0`, which is the vibrational frequency ν_0 from Eq. (1.2);
  - the change in energy `dE` (optional);
  - the energy barrier `Eacc` from Eq. (1.2);
- some additional options, such as the imposition of the central atom, or the manual inclusion of certain atoms into the event;
- and the paths to original data files containing the full initial and final state configurations R^{evt}_{ini} and R^{evt}_{fin}, and the format of those files.

An example data entry of an event is shown on Fig. 4.6.

4.3.1 Selection of event type

Based on what happens during the event, we classify three types of events: diffusion, adsorption, and desorption. The program `generate_library.x` selects the type of event based on the total number of atoms present in the input initial and final state data R^{evt}_{ini} and R^{evt}_{fin}. If the number of atoms is equal, then the event type is diffusion. If the number of atoms in final state is larger than in the initial state, then it is an adsorption event, and conversely for desorption. This decision on the

```
nevt = 6


ievt = 1
# dumbbell to hexa
rcut = 2
f0 = 1e+13
dE = 0.014
Eac = 0.278
central_atm = 0
check_sym = true

read NEB_1I_Si_pur/NEB_Si_217_1I_dumbell110_to_hexa_charge_0/initial.xyz xyz
read NEB_1I_Si_pur/NEB_Si_217_1I_dumbell110_to_hexa_charge_0/final.xyz xyz
```

Figure 4.6: An example event data entry in the file `buildlist.in`.

type of event is important for how to treat the event configurations. A schematic is shown on Fig. 4.7.

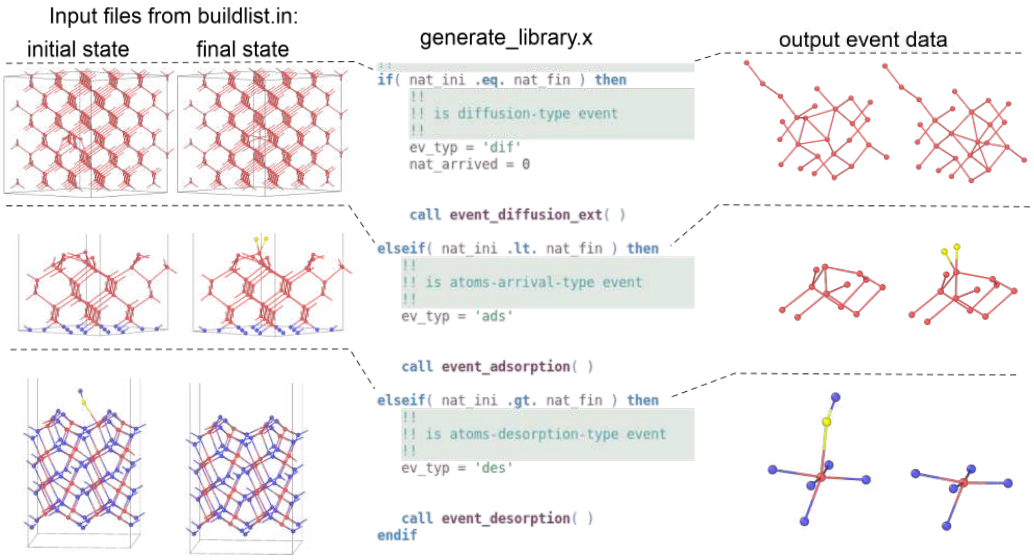


Figure 4.7: A schematic of the process of generating event configurations from input data.

4.3.2 The central atom of an event

The central atom of an event specifies the origin of the local reference frame, and acts as central/input node of the graph of the local structure. It is imposed by the

command `central_atm=`c in the event data entry (Fig. 4.6), as:

$$c = \begin{cases} 0, & \text{choose } c \text{ automatically,} \\ c, & \text{otherwise.} \end{cases}$$

In the case $c = 0$, the program `generate_library.x` will choose the central atom automatically, based on different criteria according to the type of the event.

In the diffusion-type event, the central atom c is chosen as the atom that moves the most during the event.

In the adsorption-type event, the central atom cannot be any of the atoms which arrive during the adsorption event, and is thus chosen to be the atom neighbour to any of the adsorbed atoms.

In case of desorption-type event, the central atom is chosen as one of the atoms neighbouring the atom that gets desorbed during the event.

In the case when more than one atom gets adsorbed/desorbed during an event, there are more than one atoms which are the potential central atoms (central/input nodes). In this case, the local environments of all potential central atoms are combined into the extended local environment. One of those atoms acts as the central atom.

4.3.3 Selection of the atomic environment

The list of atoms which participate in an event is chosen around the central atom, based on the criteria of cutoff, displacement during the event, and user input. Both the initial and final state configurations are taken into account for those criteria.

Each event specified in the `buildlist.in` has an associated cutoff value given by the parameter `rcut`, which can be in form of radial distance, or counting the neighbours (connections of the local graph). The choice between the two is imposed in the input file `catalogue_input.in` as the parameter `rcut_mode`, which has possible values `dist` and `neig`:

- in the case when `rcut_mode=dist` is used, the `rcut` value gives the real-valued distance around the central atom, specifying the atoms included in the local environment;
- in case `rcut_mode=neig`, the value `rcut` is an integer, specifying how many neighbours from the central/input node of the local graph we count (see Appendix A).

The `rcut` is used to generate the local configuration S around the central atom, composing the initial state S_{ini} and the final state S_{fin}.

As described in Sec. 4.2.4, there are three conditions for an atom to be part of the extended local environment. By default, the atomic indexes that are present in the S_{ini} and S_{fin} configurations are combined to form the extended environments S^E_{ini} and S^E_{fin}. The reasoning here is that if the central atom moves during the event, such that its initial state local environment S_{ini} is composed of atoms different than the atoms in its final state S_{fin}, then the event is composed of all atoms present in the initial state S_{ini}, and final state S_{fin}.

The input file `catalogue_input.in` also specifies a parameter called `small_move_thr`, which is used as a threshold value for displacement in any event. If any atom in the event moves more than this specified threshold, it will be automatically included in the extended event configurations S^E_{ini} and S^E_{fin}.

The user can also include certain atoms to the event environment by force, by specifying the command `considered_atoms`, and then listing the indexes of atoms that should be added to the extended environments S^E_{ini} and S^E_{fin}.

To summarize, let the set of integers $\{k\}_{ini}$ represent the atomic indexes of the input initial state data R^{evt}_{ini} of the event, which are present in the local configuration S_{ini} around the central atom. Similarly, let the set of indexes $\{l\}_{fin}$ represent the atomic indexes of the input final state data R^{evt}_{fin}, that are present in S_{fin}. Let the set of atomic indexes $\{m\}_{sm}$ represent all atoms which move more than the threshold `small_move_thr` during the event. And let the set of atomic indexes $\{n\}_{usr}$ represent the atomic indexes that are included in the event by the user's imposition. Then the extended environments S^E_{ini} and S^E_{fin} are formed by atoms from the input data R^{evt}_{ini} and R^{evt}_{fin} with atomic indexes $\{i\}$, such that:

$$\{i\} = \{k\}_{ini} \cup \{l\}_{fin} \cup \{m\}_{sm} \cup \{n\}_{usr}, \tag{4.3}$$

where the atoms $\{k\}_{ini}$ form the primary part of the extended environments S^E_{ini} and S^E_{fin}, and the other atoms form the extension.

The adsorption/desorption events can have more than one potential central/input node, as stated in the previous subsection. In those cases, the atomic environments S are selected around each of the potential central/input nodes, and combined to form a single extended local environment S^E. The primary part of this extended environment is the local environment S around the atom that is the designated central atom.

The order in which atoms of the extended environments S^E_{ini} and S^E_{fin} are written is important to the extent that the atoms belonging to the extension part must be written after the atoms belonging to the primary part of the environment. Atoms that get adsorbed are written at the end. The exact order of atoms within each of these sections is however not important.

4.3.4 Generation of graph hash values

As has been described in Sec. 2.2.1, the software NAUTY [73] can be used to calculate the graph isomorphism, through graph hash values. If two graph hash values are identical, then the corresponding graphs are isomorphic. Reducing the problem of structural similarity to the isomorphism of graphs greatly reduces the dimensionality of the problem, and can be resolved in a fast and efficient way with NAUTY [73]. However, fully relying solely on the simple graph isomorphism can be problematic, since a simple graph does not contain the information on specific geometry. For this reason, our kMC uses simple graph isomorphism only as fast pre-screening of the atomic sites where an event is possible, and isomorphic graphs only indicate a potential similarity in the geometry of structures.

We set up simple, undirected graphs from the atomic environment. In this sense, atomic positions get mapped to graph nodes, and the distances between atoms are mapped to the connectivity matrix, as in Eq. (2.2).

$$c_{ij} = \begin{cases} 1, & \text{if } d(i,j) \leq R_{cut} \\ 0, & \text{otherwise} \end{cases} \qquad \text{(2.2 revisited)}$$

Any graph hash can be calculated independently. For the cutoff distance R_{cut} in Eq. (2.2) which decides whether two nodes are connected or not, we use specie-dependent cutoff values given as input parameter `color_cutoff` in the `catalogue_input.in` file (see Fig. 4.5), and passed to the kMC simulation through the header of the event catalogue (see Fig. 4.8).

In order to be able to use the mode `rcut_mode=neig` for selecting the local atomic environments, we first generate the global connectivity matrix of the whole system of simulation, and then use a simple graph traversal algorithm based on the breadth-first search idea (see Appendix A), to find all nodes (atoms) connected up to `rcut=`n neighbours away from the central/input node. This subset of atoms is taken as the local atomic configuration S around a central atom. The `rcut=`n parameter is given for each event in the `buildlist.in` file, see Fig. 4.6.

The graph hash values are generated for the initial state of the event, and for the final state of the event.

In order to allow the events to each have a specific cutoff `rcut`, that is independent from the other events, we introduce the concept of common cutoff radius. The common cutoff radius `rcut_common` is the smallest common cutoff value among all events. This value will be used when parsing the system of simulation. If graph isomorphism is found with `rcut_common`, then the specific `rcut` of that event will be set, and the local configuration of the system site will be re-generated and checked with `rcut`. Therefore, a hash value of the primary local environment S_{ini} is written into the event catalogue, in the common cutoff `rcut_common`, and in the specific cutoff `rcut` of the event.

From the final state, only the graph hash value in the specific event cutoff `rcut` of the primary environment S_{fin} is generated.

4.3.5 First part of IRA; symmetries in events

Once the initial and final state configurations S^E_{ini} and S^E_{fin} of an event are known, the first part of the IRA algorithm can be done. This consists of finding a basis β for the configuration of the initial state of an event.

At each simulation step of the kMC, the possible events are those whose initial state can be found within the system. In other words, the possible events are those whose initial state configuration can be matched to some system site, with a distance below the threshold of equivalence. We make an assumption that for two configurations that match below a certain threshold, the particular choice of basis β will not affect the matching process. The choice of atoms forming the β basis is thus made simply by choosing two non-collinear atoms as close as possible to the central atom of the initial state of the event configuration S_{ini}.

At this stage, we also make a search over the possible symmetries in an event. The event symmetries give possible unique directions in which an event can occur, and are important for the statistics of a simulation.

The event symmetries θ are such that:

$$\theta S_{ini} = S_{ini}, \tag{4.4}$$

and

$$\theta S_{fin} \neq S_{fin}, \tag{4.5}$$

are satisfied simultaneously. In other words, the symmetry operation θ leaves the initial state S_{ini} of the event unchanged, while giving a different final state S_{fin}. See also Fig. 5.16, for a practical example.

The second, iterative part of the IRA algorithm (Alg. 3) provides a procedure of iterating through some candidate bases, which we can use to find the symmetries θ. However, each candidate basis in that procedure is given by relative positions of the atoms in S_{ini}. This means that if the set of candidate bases generated in the IRA procedure contains symmetry operations, they are by-design given in a reference frame internal to the atomic structure S_{ini}. In order to transform them to the general reference frame of S_{ini}, we need a transformation relating the general and internal frames. This transformation is exactly the β basis.

Thus we rewrite the structures S_{ini} and S_{fin} in the internal reference frame β, as S^{orig}_{ini} and S^{orig}_{fin},

$$S^{orig}_{ini} = \beta S_{ini} \tag{4.6}$$

$$S^{orig}_{fin} = \beta S_{fin}, \tag{4.7}$$

and insert that into Eq. (4.4) and Eq. (4.5). Then, we modify the algorithm such that at each iteration the conditions in Eq. (4.4) and Eq. (4.5) get tested. The modified algorithm is outlined in Alg. 8. Any transformation θ found to fulfil both of the conditions, Eq. (4.4) and Eq. (4.5), is taken as a unique symmetry operation of the event, and stored in form of two atomic indexes which give the θ basis, plus the integer m. In the original algorithm (Alg. 3), the integer $m = -1$ signifies that the third basis basis vector $\hat{\mathbf{e}}_3$ should be constructed as $\hat{\mathbf{e}}_3 = \hat{\mathbf{e}}_2 \times \hat{\mathbf{e}}_1$, which is just the negative vector of $\hat{\mathbf{e}}_1 \times \hat{\mathbf{e}}_2$. In order to explore a larger portion of the transformation space for θ, we extend the iteration over candidate bases by including also the negative vectors of each basis vector, as possible basis vectors. Like this, the possible values for m are -3, -2, -1, and 1 (Alg. 8 line 5). Specifically, the value $m = -3$ means that the basis vector $\hat{\mathbf{e}}_3$ is multiplied by -1, the value $m = -2$ means that the basis vector $\hat{\mathbf{e}}_2$ is multiplied by -1, and similarly the value $m = -1$ means the basis vector $\hat{\mathbf{e}}_1$ is multiplied by -1, while the value $m = 1$ does not flip any basis vector (see Alg. 8 line 7).

Algorithm 8 Modified iterative part of IRA, used to find symmetries in events.

Input: structures S_{ini}, S_{fin}, basis β
Output: integers m, i, j

1: $S_{ini}^{orig} \leftarrow \beta S_{ini}$
2: $S_{fin}^{orig} \leftarrow \beta S_{fin}$
3: **for** $i \in S_{ini}^{orig}$ **do**
4: **for** $j \in S_{ini}^{orig}$ **do**
5: **for** $m \in \{-3, -2, -1, 1\}$ **do**
6: $\theta \leftarrow$ SelectBasis(i, j, m)
7: $\theta(|m|, :) \leftarrow sign(m) * \theta(|m|, :)$ ▷ Flip the $|m|$-th basis vector
8: $S_{ini} \leftarrow$ Rotate(S_{ini}^{orig}, θ)
9: $d_{H,ini} = D(S_{ini}^{orig}, S_{ini})$ ▷ Compute by CShDA
10: **if** ($d_{H,ini} < thr\_sym$) **then** ▷ Condition Eq. (4.4)
11: $new \leftarrow$ `True`
12: $S_{fin} \leftarrow$ Rotate(S_{fin}^{orig}, θ)
13: Check against all S_{fin}^{memo} in memory:
14: $d_{H,fin} = D(S_{fin}^{memo}, S_{fin})$ ▷ $\forall S_{fin}^{memo}$ in memory
15: **if** ($d_{H,fin} < thr\_sym$): $new \leftarrow$ `False` ▷ Eq. (4.5)
16: **if** (new): add m, i, j, S_{fin} to memory
17: **end if**
18: **end for**
19: **end for**
20: **end for**
21: **return** all m, i, j in memory

4.3.6 Format of the catalogue

The header of the final catalogue of events includes some global parameters of events, such as number of events, number of atomic species, the threshold for equivalence of two structures, the specie-dependent cutoff for the connectivity matrix (see Sec. 4.4.1), and the common cutoff radius `rcut_common` of all events in the catalog. An example of the header is shown on Fig. 4.8.

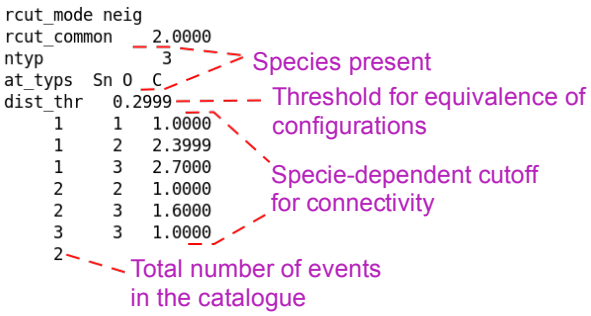


Figure 4.8: Example of the events catalogue header lines with global parameters used for all events in the catalogue.

Each event written in the final event catalogue includes information regarding the topology of the local environment of the event, the energetics data of the event, and the exact configurations of the initial and final states in the xyz-format, written in the reference frame of the original input file of the event. Due to the use of extended environments, additional information is needed as to how many atoms there are present in the primary environment, and how many atoms are present in the extension. As mentioned before, the atoms belonging to the primary environment are written first, then the atoms of the extension. At the end, information regarding the basis β and possible additional symmetries is written in the form of indexes m, i, j from Alg. 8, such that the first triplet m, i, j defines basis β, and the others give additional symmetries θ. An example diffusion-type event entry is shown on Fig. 4.9, with explanations of the event-specific information, the primary local environment marked with red, and the extended local environment marked with green. The example event has two possible symmetries, written in the last two lines containing the integers m, i, j.

In the adsorption-type event, the atoms that get adsorbed are written on the final spots of the final configuration. We also need an additional integer to tell how many get adsorbed. An example of adsorption-type event entry in the final catalogue is shown on Fig. 4.10. The adsorbed atom are lined with blue. In desorption-type event, the desorbed atoms are written at the end of initial configuration. An example is shown on Fig. 4.11.

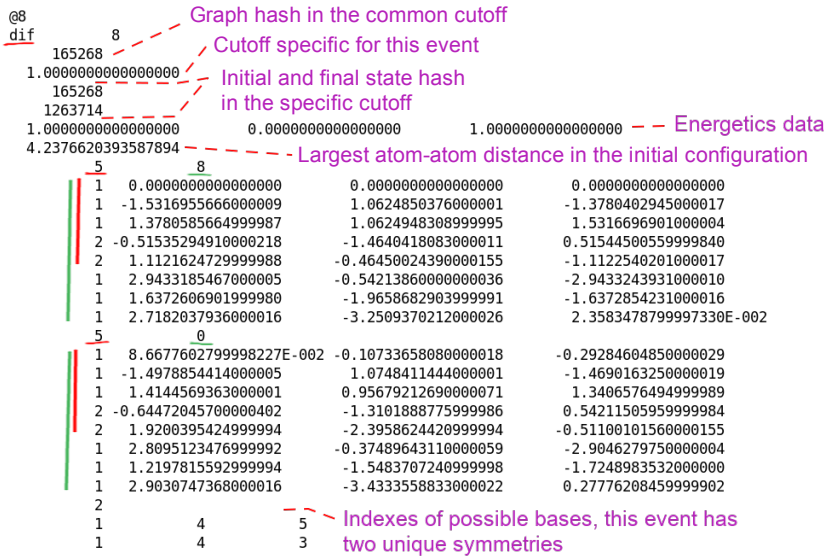


Figure 4.9: Example of a diffusion-type event entry in the final event catalogue. The primary local environment is marked with red, while the extended environment is marked with green. The central atom is at the first spot. The event has two unique symmetries, written in the form of two triplets of integers at the end. The first triplet gives the β basis, and the second triplet gives one additional symmetry θ of the event.

4.4 Inside kMC: identify possible events, apply chosen event

After the event catalogue has been generated, it is passed to our kMC software by specifying the corresponding filename into the kMC input script `input_kmc.in`. A schematic of the process from generation of event catalog to the final simulation is shown in Fig. 4.12.

The first thing that kMC program does is to check the graph hash value of each event in the common cutoff `rcut_common`, and make a table of which events are associated to each of them. Like this, when a certain graph hash is found in the simulation box during the step of identification of possible events, we know directly which events are the candidate events for this topology. Then each of the candidate events is checked within its own specific cutoff `rcut`. If isomorphic graphs are found, then the system site configuration and the event data undergo the shape matching procedure, namely the second part of our IRA algorithm. If the shape matching procedure gives a distance between atomic structures that is below the threshold of equivalence, then the event is deemed possible. At this

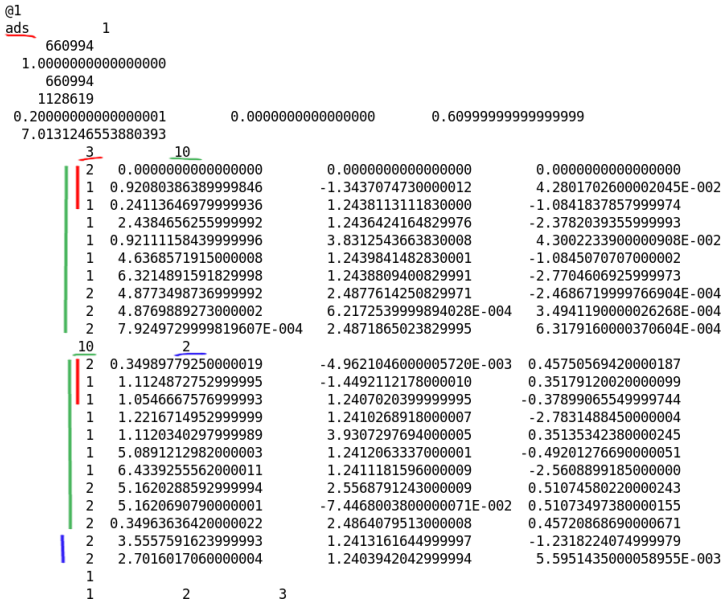


Figure 4.10: Example of adsorption-type event entry in the final catalogue of events. The adsorbed atoms are lined with blue at the end of the event final state.

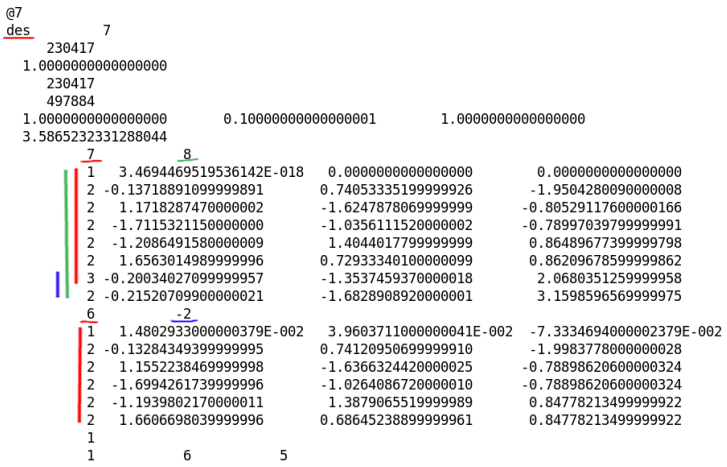


Figure 4.11: Example of desorption-type event entry in the final catalogue of events. The desorbed atoms are lined with blue at the end of event initial state.

stage, all symmetries of the event are also checked if they exist in the system.

After the procedure of identification of all possible events, and the subsequent choice of an event to be executed next, the transformation prescribed by the chosen event needs to be applied in the system of simulation.

After the application of an event, the status of the structures within the system

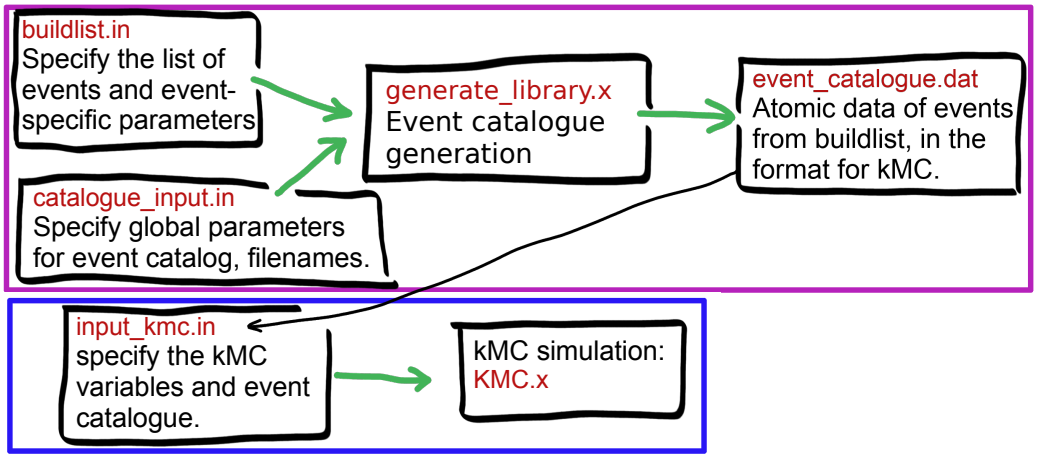


Figure 4.12: A schematic representation of the process from generation of event catalogue to the final kMC simulation.

of simulation needs to be updated for the next step.

In Sec. 4.4.1, the details of checking the graph isomorphism within the system of simulation are given. In Sec. 4.4.2 and Sec. 4.4.3, the details on the second part of the IRA algorithm application are given. In Sec. 4.4.4, the details of the application of an event are given. The update of the status of the system is described in Sec. 4.4.5.

4.4.1 Topology check

When parsing over the system of simulation R^{sys}, a local configuration is generated for each atom in the system, first with the common cutoff value `rcut_common`. Graph hash value with this environment is calculated, and if there are any corresponding events with the same common hash value (Sec. 4.3.4), each of them is checked with its specific cutoff value `rcut`. If the graph hash value in the specific cutoff indicates isomorphic graphs, then the current central atom can be a potential candidate atom for an event, and it undergoes a full geometric check for the local environment, through the second part of our IRA algorithm.

4.4.2 Second part of IRA

The task of the second part of the IRA algorithm is two-fold. Firstly, it provides a way of unambiguous evaluation of the geometric similarity between two configurations. And secondly, it calculates the exact rigid transformation of one atomic configuration which matches it to another atomic configuration.

An event written in the event catalogue can be written as a transformation T. T prescribes how an initial state S^E_{ini} should transform into a final state S^E_{fin} in a generic way, without specifying any particular reference frame, a *relative* atomic move. It can be written as:

$$S^E_{ini} \quad \overset{T}{\rightarrow} \quad S^E_{fin}. \tag{4.8}$$

However, what needs to happen in the kMC simulation is a very precise application of this event, a transformation of the initial state R^{sys}_{ini} into a final state R^{sys}_{fin}, which can be written as:

$$R^{sys}_{ini} \quad \overset{T'}{\rightarrow} \quad R^{sys}_{fin}, \tag{4.9}$$

where T' is the transformation prescribed by an event, written in the correct reference frame of R^{sys}. The correct reference frame can be found by finding a relative rigid transformation F between S^E_{ini} and R^{sys}_{ini}, such that:

$$FR^{sys}_{ini} = S^E_{ini}, \tag{4.10}$$

where F represents rigid rotation and/or reflection, translation, and permutation of indexes. When F is known, the transformation T' is written as:

$$T' = F^{-1}T = F^{-1}(S^E_{fin} - S^E_{ini}). \tag{4.11}$$

The relation between all equations from this section can be written as a closed path:

$$\begin{array}{ccc} S^E_{ini} & \xrightarrow{T} & S^E_{fin} \\ {\scriptstyle F}\uparrow & & \downarrow{\scriptstyle F^{-1}} \\ R^{sys}_{ini} & \xrightarrow{T'} & R^{sys}_{fin} \end{array} \tag{4.12}$$

where finding the transformation F is done by our IRA algorithm.

4.4.3 Geometry check

Since we know the central atom of the event initial state configuration S^E_{ini}, and we know the current atomic site from R^{sys}, the translation part of F can easily be computed by a simple shift of reference frames. After that, the transformation F is equivalent to the approximate rotation $\mathbf{R}_{apx}$ from Sec. 3.11. It thus consists of the bases β and γ, such that

$$F^{-1} = \gamma^{-1}\beta. \tag{4.13}$$

As has been described earlier, β can be found independently, and is already known at this point of the kMC. Namely, it is written in the event catalogue in the form of a triplet of integers under each event (see Fig. 4.9).

The task is thus to find the γ basis. There are some algorithmic advantages when calculating γ in the particular setting of the kMC, with respect to the situation when matching generic structures, described in Chapter 2. Firstly, the central atom is known for both configurations. And secondly, we are not particularly concerned about mismatches, since a mismatch in this case only means that two configurations are not sufficiently similar, and thus an event is not possible at current site (see also test in Sec. 3.8.6). If atomic environments are similar, IRA will be able to find the match (see also the 100% efficiency rate in Sec. 3.8.1).

Since there are atoms present in S^E_{ini}, which cannot be found by prescribed cutoff check in R^{sys}, namely the atoms of the extension of the environment, we take a larger portion of the simulation box around the current atomic site, and exploit the capability of IRA to match with different number of atoms. The exact specification of which portion we should take is given by the largest atom-atom distance in the event, which is given for each event entry in the event catalogue, see Fig. 4.9. This number is increased by factor 1.2, and taken as radial distance cutoff from the central atom in R^{sys}. All atoms within this distance are then taken as tentative local environment S_{sys}, and matched to S^E_{ini} to find basis γ. Note that S_{sys} chosen like this contains more atoms than S^E_{ini}, but the central atom for both is known. Thus we can use the Alg. 5 of IRA, where we replace the geometric center by the position of the central atom, to find the basis γ.

When the best γ basis is found, we can evaluate the matching between the atomic configurations. This is done by calculating the distance h from the IRA algorithm, Eq. (4.14).

$$h = h(F^{-1}S^E_{ini}, S_{sys}) \tag{4.14}$$

If the evaluation of h is favourable (is below the threshold of equivalence), then the event is deemed possible at this site, and the atomic indexes that set the γ basis are recorded in memory. At this point, all the additional symmetries of events are checked by evaluating the equation:

$$\gamma^{-1}\theta\beta S^E_{ini} = S_{sys}, \tag{4.15}$$

where θ are the additional unique symmetries (see Sec. 4.3.5), given by the triplets of integers m, i, j, written at the end of an entry in the event catalogue (see Fig. 4.9). Each symmetry with a favourable evaluation of Eq. (4.15) is added as a separate event into the temporary array of possible events to occur at the current simulation step, since each of them represents a unique possible direction of that event.

Once all atomic sites in R^{sys} with a common hash value associated to any event have undergone such topological and eventual geometry checks, the list of all possible events at the current simulation step is known. Each event from this list has an associated atomic site index, event tag/index, event rate, atomic indexes

that specify the γ basis, and the indexes that specify the θ basis. Now it is back to the generic kMC algorithm in Alg. 1, to choose the next event to apply, based on the transition rates (procedure CHOOSE_EVENT() in Alg. 2).

4.4.4 Apply event

When the kMC chooses an event from the list of events, almost all information regarding the structure/matching of S^E_{ini} and S_{sys} can be deduced from the information already in the memory: the atomic site, event tag/index, the indexes that specify the γ basis, and the indexes that specify the θ basis. The only piece of information missing is the permutation P which gives the permutation of atoms in S^E_{ini} which maps them to atoms of the simulation box R^{sys}. The rigid transformation F is thus known, minus the permutation. The permutation P found by first applying the known transformation F^{-1} to S^E_{ini}, and then computing P using the CShDA algorithm (see Sec. 3.4).

Once the permutation P is also known, the transformation T' which will transform the system of simulation from its initial state R^{sys}_{ini} to its final state R^{sys}_{fin} can be written in the form of a $3N$-dimensional array, where N is the total number of atoms in the system, which contains values zero for atoms that do not participate in the event, and for the others:

$$T' = F^{-1}P_i(S^E_{fin} - S^E_{ini}), \tag{4.16}$$

where P_i gives the mapping of index of atom i from R^{sys}_{ini} to an index from S^E_{ini}, and F^{-1} is the rigid transformation

$$F^{-1} = \gamma^{-1}\theta\beta. \tag{4.17}$$

With all this information, the event can be applied in two different ways:

$$R^{sys}_{fin} = R^{sys}_{ini} + T' \tag{4.18}$$

or

$$R^{sys}_{fin} = F^{-1}PS^E_{fin}. \tag{4.19}$$

The Eq. (4.18) will propagate and accumulate any distortions that might be present in the structure R^{sys}_{ini}, since T' contains the difference $(S^E_{fin} - S^E_{ini})$. Meanwhile, the Eq. (4.19) imposes the final state of the event S^E_{fin} onto the system configuration, without regard to what was previously there.

There are some advantages and some disadvantages in applying events by Eq. (4.18), or by Eq. (4.19). As already mentioned, with application of event following the Eq. (4.18), some possible distortions that are present in the system before application of an event will get propagated and accumulated. This means

that the kMC evolution propagates subtle distortions that might appear during its history, into the future. In our case of the kMC with a fixed catalogue of events this is a drawback in the situations when many events can happen close to each other, and in fact interfere with one another. In that case the application of one event can effectively destroy the application of a close-by event due to subtle atomic movements in its proximity.

By application of event following Eq. (4.19), we effectively ignore all atomic distortions that are present in the system before the current event, and impose the final atomic positions to be exactly equal to those given by the event in the catalogue. As such, an event application might destroy subtle atomic structures (usually near the border of the event), that can be essential to the recognition of some other, nearby event. Nonetheless, the system locally ends up in a well-defined final state, which means that it can always continue the propagation with an appropriate next event. However, possibly at the cost of all the rest of the box being destroyed.

To a great extent, the problems of Eq. (4.18) and Eq. (4.19) are related to the atomic distortions present in the original event data, which are possibly inconsistent among each other, and to particular cases when a kMC simulations is performed with too few events in the catalogue. The differences between Eq. (4.18) and Eq. (4.19) disappear when the relaxation of the forces is included in a simulation, or the events are explored with an on-the-fly approach. However, in an attempt to mitigate the propagation of distortions as much as possible, we mix both Eq. (4.18) and Eq. (4.19). This is done as follows.

The event is first applied with Eq. (4.18). After the application, the coordinates of R^{sys}_{fin} are compared to the final state coordinates $F^{-1}S^{E}_{fin}$. If any atom in R^{sys}_{fin} is within a distance larger than some threshold away from its associated atom in $F^{-1}S^{E}_{fin}$, then an interpolation scheme is launched, which interpolates the coordinates R^{sys}_{fin} and $F^{-1}S^{E}_{fin}$ with some factor $m \in [0, 1]$, such that the case $m = 0$ corresponds to applying the event purely following Eq. (4.18), and the case $m = 1$ essentially overwrites R^{sys}_{fin} with $F^{-1}S^{E}_{fin}$, which is equivalent to applying the event following Eq. (4.19). The distance threshold for comparison, and the interpolation factor m can be set in the kMC input script. The interpolation scheme used is the Image-Dependent Pair-Potential (IDPP), described in Sec. 4.5.

When an adsorption-type event is applied, all arrays are increased by the number of arrived atoms, and the arrived atoms are placed at the end of the arrays. In case of desorption-type event, the indexes of atoms that need to be removed are first calculated, and then removed from atomic positions in order of descending index value. All arrays are resized appropriately.

4.4.5 Local update

After an event is applied, we could scratch all the information about possible events, and recompute it all from beginning. This is however computationally expensive, so a better idea is to do what we call a local update.

The idea of local update is that we can save all information about possible events from the previous step, except for a region of the simulation box where the latest event has occured. Thus saving potentially quite a lot of computation. We can reuse the information of the permutation P from the event application step, which gives the indexes of atoms in R^{sys}, which have taken part in the applied event. The search for possible events in the next step will then recycle all the old information, and only update these atomic sites. There are however more sites in R^{sys} which can potentially change their geometry due to an event being applied in their vicinity, namely the atoms at the border of the applied event. A schematic of this idea is shown on Fig. 4.13, where the grid represents the global connectivity matrix, the node marked in green is the central atom of the applied event, the blue nodes are the atoms involved in the event, and the pink atoms are the atoms with potential change in their geometry, due to an event applied in their vicinity. A first-neighbour cutoff is assumed for the schematic. The atoms in blue might have changed their hash value due to the event. However if any blue atom that is close to the border of the region changes their position, the change would be *seen* by atoms in their cutoff range, thus all the atoms within the cutoff region of all blue atoms need to be checked/updated in the next step. In order to obtain the atomic indexes of these atoms, we take the value of the common cutoff `rcut_common` of all events, and use the breadth-first graph traversal algorithm on the global connectivity matrix (Appendix A).

Once the list of all the atoms which need to be updated is known, the corresponding parts of the global connectivity matrix are computed, and the simulation goes into the next step.

We occasionally see problems with the local update procedure. Most notably in simulations with a large number of events possible to occur close-by, where the events have a low specific cutoff `rcut`, but they contain lots of atoms in the extended environment S^E. The problem is that the atoms that get updated by the local update procedure, are not all the atoms that should get updated. In other words, there are atoms that are left non-updated. This makes sense, since the updated region only encompasses the region within common cutoff value around the applied event, and the atoms belonging to the extended environment are not included in that region. This could be resolved by increasing the range around the region of applied event, but this potentially means lots of redundant computations in the next step of event identification. Thus, we currently resolve the problem as follows. At the stage of event application, the event configuration is transformed

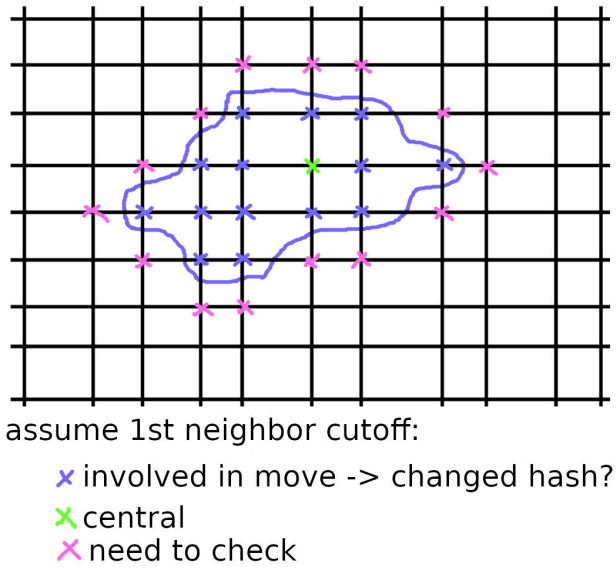


Figure 4.13: A schematic of the local update idea. The grid is an illustration of the global connectivity matrix, the node in green marks the central atom, and the nodes marked in blue represent atoms which have been involved in the event application, the pink atoms are atoms with potential change in geometry due to an event in their vicinity.

by F in Eq. (4.17), the distance $h(F^{-1}S^E_{ini}, R^{sys})$ can then be readily evaluated. If it is beyond the similarity threshold, then we know something has happened in that atomic site environment, and thus the event should be aborted and this atomic site added to the update list for the next step, along with all its neighbours within the common cutoff region.

Note that this local update procedure only updates the information regarding the structures in the simulation. After this local update, the algorithm passes to the UPDATE_SYSTEM() procedure in Alg. 1, which continues the generic kMC.

4.5 Image-Dependent Pair Potential (IDPP)

Image-Dependent Pair Potential (IDPP) is a method that has originally been developed for guessing a good initial path for a transition between given initial and final configurations, it is presented in Ref. [122]. The method presents a better alternative to the simple interpolation of Cartesian coordinates, and does not rely on any external force or energy computation.

In our kMC, IDPP is used as interpolation scheme in order to mix the Eq. (4.18) and Eq. (4.19), as explained in Sec. 4.4.4. As such, it can be seen as a mitigation

strategy for the accumulation of distortions, in place of a real relaxation of the forces with a realistic potential. Concretely, it prevents the final state configuration in the kMC simulation to deviate too much from the final state configuration of an event in the catalogue.

4.5.1 Original algorithm

The initial path for a transition between two states typically consists of a number of so-called images of states that are between the given initial and final states. Traditionally, the k-th image R^k is obtained by directly interpolating the Cartesian coordinates of the initial state R_{ini} and final state R_{fin}, by the Eq. (4.20).

$$R^k = R_{ini} + \frac{k}{p}\big(R_{fin} - R_{ini}\big), \tag{4.20}$$

Where p represents the total number of images to be created, and k runs from 1 to $p-1$.

The main problem of direct linear interpolation on the Cartesian coordinates is that it generates a *straight line* between the initial and final configurations. For example, if two atoms exchange their positions from R_{ini} to R_{sys}, then Eq. (4.20) will generate images such that the two atoms will go straight through each other. Other vivid example of this problematic are curved-arc paths, for which the Eq. (4.20) generates a straight path which can potentially mean atoms coming extremely close to each other.

For the reasons above, IDPP [122] method has beed developed, which is based on the idea of interpolating atomic distances, instead of directly on the coordinates. The atomic distances in the sense of all atomic distances d_{ij}^A within a structure A. As such, the interpolation can be written as Eq. (4.21).

$$d_{ij}^k = d_{ij}^{ini} + \frac{k}{p}\big(d_{ij}^{fin} - d_{ij}^{ini}\big), \tag{4.21}$$

where d_{ij}^k, d_{ij}^{ini}, and d_{ij}^{fin} are the all-to-all atomic distance matrices of the interpolated image k, the initial configuration, and the final configuration, respectively. The variables k and p are the same as in Eq. (4.20). The difference between interpolating on coordinates and interpolating on distances is shown on Fig. 4.14, where interpolation on coordinates is shown in dashed lines, and interpolation on distances in full color.

In order to pass from a representation in the form of interpolated distance matrix d_{ij}^k to an atomic structure R^k, represented by a $3N$-dimensional array, an objective function is defined by Eq. (4.22):

$$S_k^{IDPP}(R) = \sum_i^N \sum_{j>i}^N w(d_{ij}) \left(d_{ij}^k - d_{ij} \right)^2, \tag{4.22}$$

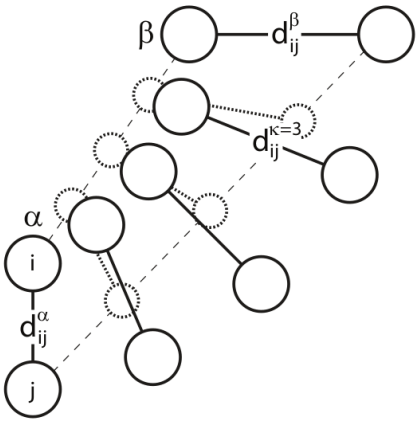


Figure 4.14: Schematic of paths between initial structure α and final structure β. In dashed line: path generated by interpolation on coordinates, by Eq. (4.20). In full color: path generated by interpolation on distances from Eq. (4.21). Image taken from Ref. [122].

where d_{ij}^k is the interpolated distance for image k from Eq. (4.21), d_{ij} is the Cartesian distance between atoms i and j in the current structure R, and $w(d_{ij})$ is a weight function used to put more weight onto shorter distances, such that the value rises when two atoms come close together. The form of the weight function is $w(d) = 1/d^4$. The objective function $S_k^{IDPP}(R)$ can be seen as a sort of pair potential energy function, that depends on the image k. This is where IDPP got its name. Then an image-dependent "force"-like quantity on atom i of structure R is given by Eq. (4.23).

$$F_i^k(R) = -\nabla_i S_k^{IDPP}(R) \tag{4.23}$$

The next step is to minimize the image-dependent "force" in each image k. In the original article, this is done by imposing a Nudged Elastic Band (NEB [123]) algorithm where the starting point are images computed by the linear inerpolation from Eq. (4.20), and the "forces" are computed by Eq. (4.23). Note that the gradient in Eq. (4.23) can be computed analytically.

4.5.2 Our modifications

The modifications that we made to the original IDPP algorithm are only minor. Instead of taking as input the number of images k to be generated, we take as input directly the ratio $m = \frac{k}{p}$, which is a factor in the range [0,1]. Like this, IDPP generates only one image, which is located at factor m between the states R_{ini} and R_{fin}.

The second modification we make is to not impose a NEB minimization on the generated images, but instead do a steepest descent algorithm with image-dependent "forces" defined as in Eq. (4.23). We find satisfactory results with step size 0.05, and exit criterion when maximal "force" on any atom has a norm below 0.1.

4.6 Why so complicated?

In this section, we look back at the current chapter from a larger perspective, and question some ideas that are used.

The IRA algorithm presented in the previous Chapter 3 is parameterless, independent of the shape of a structure, and does not need any specific distance cutoffs. Then why is the matching process implemented in the kMC, as described in the current Chapter, seemingly quite complicated?

In principle, the events could be without a specific cutoff. Within that idea, the only criterion for deciding which atoms take part in an event could be the distance that atoms move during an event. The atomic configuration of an event would then be some configuration of atoms, without any particular order or defined shape. The procedure of identifying possible events could rely solely on the IRA matching procedure, without the need of graph isomorphism. This would simplify the creation of the event catalogue. In particular, there would be no need for any cutoffs, which are actually needed only for generating the graphs, and consequently, there would also be no need for the concept of extended local environments. The stage inside the kMC simulation would also be simplified, for the same reason. Within that idea, the process of identifying possible events, which is comparing and matching structures, would get much heavier computation-wise. Here is why.

Let us imagine a simple system of simulation R^{sys}, with 100 atoms in total. Assume one single event, which is possible at 5 sites in the system. Assume that the initial state configuration of the event S^E_{ini} includes 15 atoms. Let the initial state configuration of the event S^E_{ini} be labelled structure A, and the entire system of simulation R^{sys} be labeled structure B. The process of identifying possible events in kMC is equivalent to the IRA algorithm when matching structures A and B with nonequal number of atoms (see Alg. 4), where the potential central atoms of B are the potential event sites in the system. The difference is that, instead of searching for a central atom $b_J \in B$ for which the distance D_J is minimum (as in Alg. 4), we search all atoms $b_J \in B$ where D_J is below a threshold value, which means the site has a possible event in kMC. In the present imagined example, there are 100 atoms in structure B, thus 100 potential central atoms $b_J \in B$, or potential event sites that need to be checked. As has been discussed in Sec. 3.8.4,

and Sec. 3.8.5, it is very beneficial for the IRA algorithm, in terms of speed, to spend some computational time on reducing the set of possible central atoms. For our imagined example this means reducing the set of potential 100 central atoms $b_J \in B$, preferrably down to 5, or as close to that number as possible. In our kMC, the reduction is done by graph isomorphism. The graphs are constructed in an automatic way, which means specific cutoffs are needed. Firstly the `color_cutoff` parameters, needed to generate the global connectivity (see Fig. 4.5), and then the event-related common cutoff `rcut_common` (Fig. 4.8), and the event-specific `rcut` (Fig. 4.9), which specify the generation of local graphs, and thus local atomic structures S. Checking the graph isomorphism of local structures represents a means of fast-filtering of the possible event sites in the system R^{sys}, and is the main reason why the matching process implemented in the kMC seems quite complicated.

There are also other benefits (not only the computational speed), associated to having some cutoff distances. The most notable being the "sphericity" of the primary local environment S, that results from the environment being generated with a certain cutoff. This "sphericity" makes its use in the symmetry-finding algorithm (see Sec. 4.3.5). If the structures S_{ini} were without any order and shape, the criterion for a symmetry θ in Eq. (4.4), would be impossible to evaluate in the form currently written.

The graph isomorphism condition, as means of fast-filtering of the structures, could also be replaced by any other similarity descriptor. The main reason for using graphs in our current implementation is the fact that NAUTY [73] associates an integer hash value to each graph. The graph isomorphism thus becomes a binary yes/no operation, which checks if two hash values are equal or not. We typically set the specie-dependent R_{cut} values (called `color_cutoff` in our kMC), which specify the connectivity matrix of a graph (see Eq. (2.2)), to typical first-neighbour distances. As such, the graphs are a direct mapping of the atomic structures, and the first-neighbour bond network. If we choose the R_{cut} values such that they are slightly beyond the typical bond-length values (a few percent above, e.g. 5-10%), then the graph construction remains unchanged upon small distortions of the atomic positions. This brings some distortion tolerance into the graph isomorphism process, and gives the user some control over it via the relatively intuitive R_{cut} values. Similarly, some distortion tolerance could be achieved with other shape descriptors, which often come in real-valued scalars (or vectors), and with a similarity function that gives a range of possible values. However in that case, we would need to decide a threshold value for deciding if two structures (or their descriptors) are perceived as equal or not, which might not be as intuitive.

Chapter 5

Examples

This chapter presents some examples done with our kMC software, based on IRA algorithm. The different examples are used to showcase the capabilities of the present stage of our kMC development, as well as to point to the difficulties and potential problems that have been encountered.

The example in Sec. 5.1 is a simple toy model, used to show the necessity of the use of extended local environments for the event configurations. It also serves as simple demonstration of the application of symmetries in events.

The example in Sec. 5.2 is slightly more realistic, since it uses real Density Functional Theory (DFT)-produced data of O and O_2 diffusion in Si crystal. It explores the artificially induced distortions in the system of simulation, as function of the event size.

The example in Sec. 5.3 presents a successful simulation of an interstitial atom diffusion in Si crystal, using events generated from DFT data, where our symmetry checks reveal a more complete set of events. It is also used to discuss the calculation of the mean displacement of the diffusion.

The example in Sec. 5.4 is used to showcase the capability of our kMC to deal with events that change the total number of atoms in the simulation, namely adsorption and desorption events. The example used is a working principle of a CO gas sensor, on a SnO crystal.

The example in Sec. 5.5 presents an example where all the capabilities of our kMC are utilised. It is the simulation of O_2 adsorption on the Si(100) surface. Some advantages of using IRA are shown, namely the capability of recognition of the different Si sites on the buckled surface. Some problems are also presented, that are connected to elastic distortions in events, event size, and interference between several events happening close together on the surface. The example serves as the main motivation for the on-the-fly approach, not implemented yet presently.

5.1 Toy model: simple cubic crystal

To showcase some of the essential parts of the algorithm, we shall use a toy model system. That is a system of simple cubic crystal, periodic in all directions, composed of two atomic types. This system is not meant as an analogy to anything physical, it is chosen solely for demonstration purpose.

Toy model system is shown on Fig. 5.1. The two atomic species are A and B, colored red and blue respectively on Fig. 5.1. There are 500 atoms in total, 491 of type A and 9 of type B. Atoms of type A are shown in smaller size for reasons of better visualization.

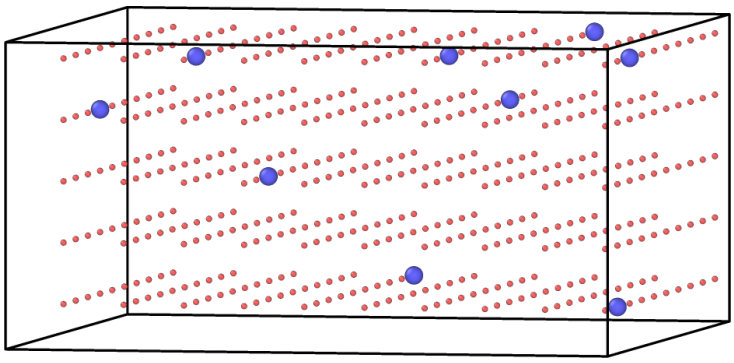


Figure 5.1: The toy model system used to showcase the different parts of the kMC.

5.1.1 Test 1

For the beginning, we include one single event, which is exchange of atoms of type A and B in their first-neighbour positions. The event is shown on Fig. 5.2, with all corresponding directions. The event is centered on the atom B (blue), the exchange of atoms happens in such a way that the atom of type B (blue) does not *see* the first-neighbour local environment of the site where it is diffusing into. And the atom of type A (red) does not *see* the local environment it is diffusing from. From our symmetry checks (see Sec. 4.3.5) we find that the event is possible to execute in 6 possible directions, consistent with the expectation. This is the first, and very simple confirmation of the symmetry checking algorithm.

Set in this way, the kMC finds an event possible on each blue atom, and the event at each blue site is initially possible in 6 directions. This is a simple confirmation that also the kMC can read, find, and execute the different symmetries of an event. As the simulation progresses, blue atoms meet other blue atoms, which makes their local environment different from the initial state of the event, and they stop moving since there is no event corresponding to these structures in the catalog.

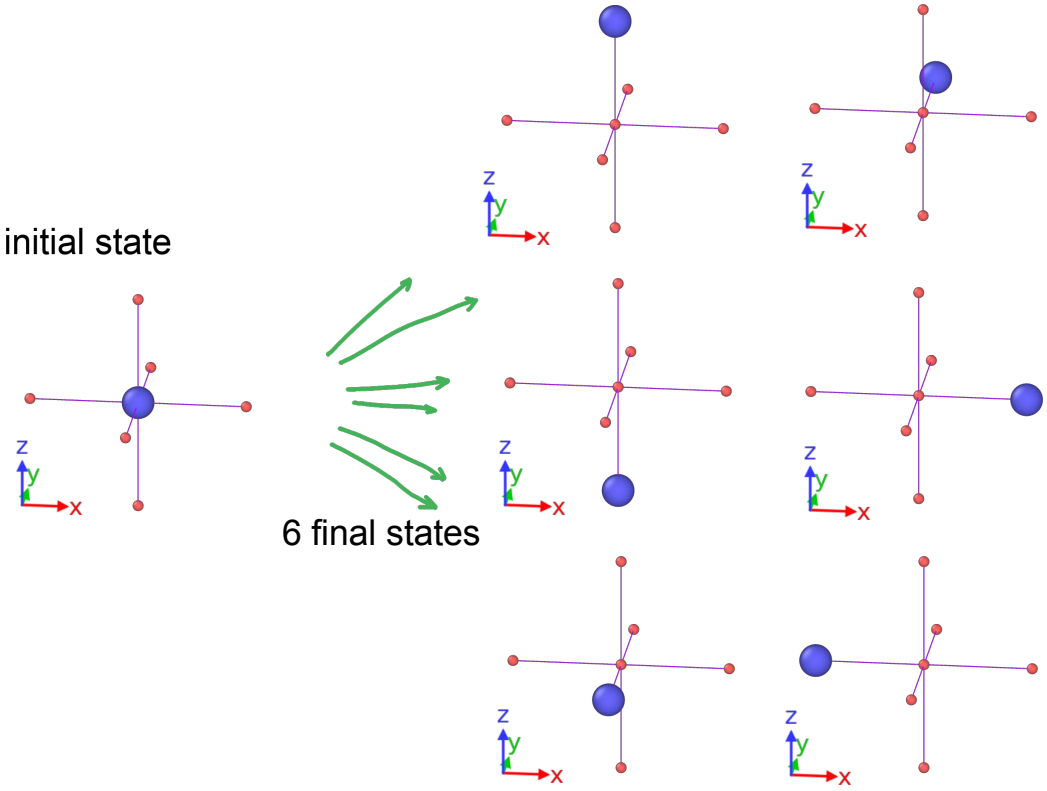


Figure 5.2: Exchange of atoms of type B in blue, and type A in red, centered on atom of type B. The six possible directions of the event are shown.

Thus as expected, what happens in the long kMC simulation with this setup is that the atoms of type B stop moving as soon as they become the first neighbour of another atom of type B. Some specific final states of the simulation box are depicted on Fig. 5.3, all starting from the same initial state depicted in Fig. 5.1. The bonds are drawn among atoms B when they are in the first neighbour position. We can observe that the atoms of type B (blue) have formed some clusters, and the simulation stops at this point because there is no relevant event present in the event catalog.

Essentially, the blue atoms diffuse randomly in all directions, until they meet another blue atom at one of their first-neighbour positions. At that point, they are blocked and cannot move anywhere. From the point of view of the event catalog, it is impossible to avoid this scenario without modifying the event configurations.

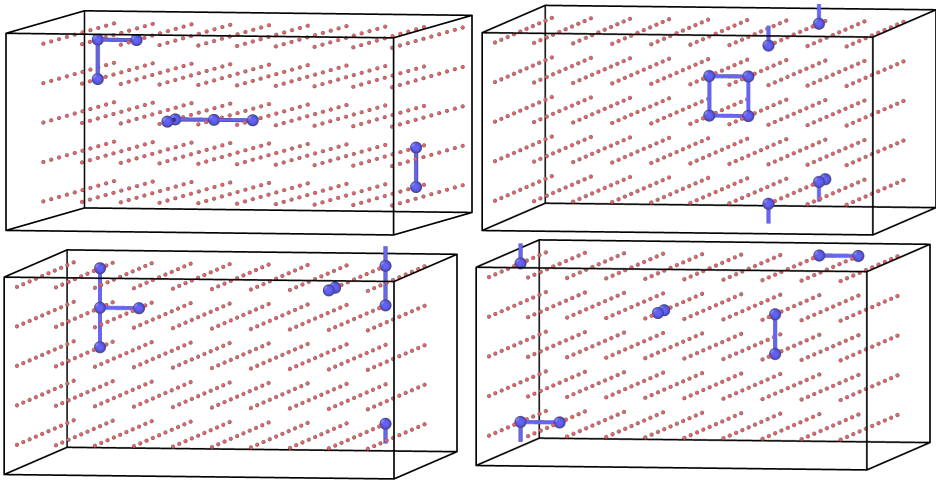


Figure 5.3: Four specific final states of toy model simulation with the event from Fig. 5.2. The atoms of type B (blue) have formed some clusters, and the simulations stop due to no possible further event.

5.1.2 Test 2

Another possibility for an event of identical mechanism - exchange of atoms A and B - is to center the event on atom A, as shown on Fig. 5.4. This time the atom A does not *see* the local environment of the state it is diffusing into, and the atom B does not *see* its local environment at the initial state. In this kind of setup, the kMC finds possible event on the 6 red atoms neighbouring to a blue atom. There is thus no need for having all different possible directions stored in

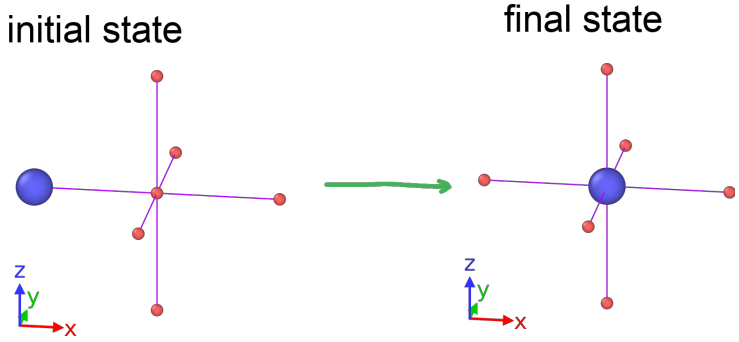


Figure 5.4: Exchange of atoms of type A and type B, centered on atom of type A. There are no additional directions of the event, since it is found at each of the 6 A atoms around a B atom, with corresponding rotation.

the event catalog.

The kMC simulation of this setup never ends up in a situation where two B

atoms come to first-neighour sites, since that would require an event allowing this kind of move. Some specific states of kMC simulations are shown on Fig. 5.5, all starting again from state in Fig. 5.1, but now with the event catalog consisting of single event from Fig. 5.4. The kMC evolution of this system can run for an indefinite number of steps.

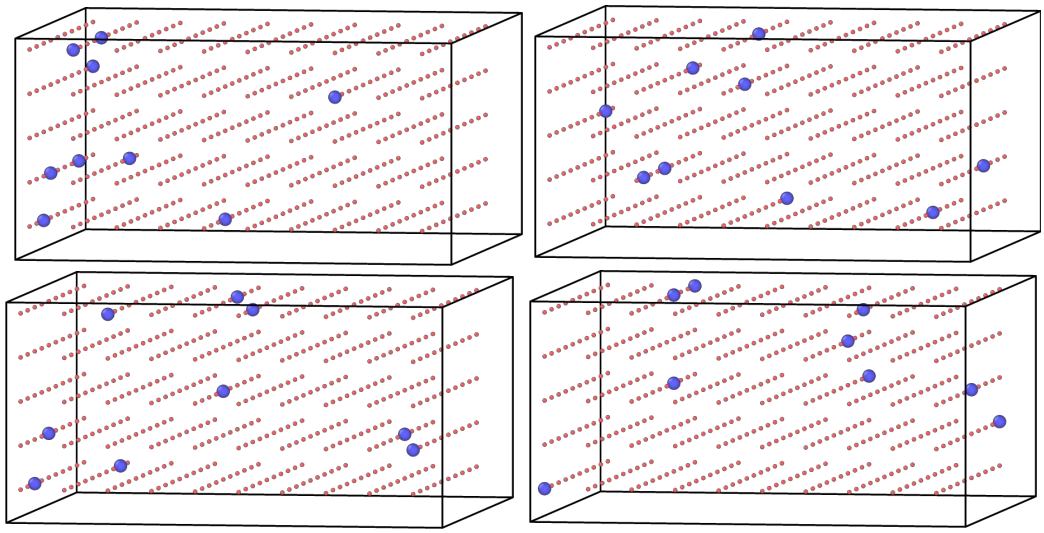


Figure 5.5: Four specific states of toy model simulation with the event from Fig. 5.4. The blue atoms never end up on first neighbour positions since the event corresponding to that does not exist. This simulation can run indefinite number of steps.

5.1.3 Test 3: justification of extended environment

From the Test 1 in Sec. 5.1.1 we can see that from the point of view of the event catalog, there is no real way to control the clustering mechanism of the atoms B. One might suggest to increase the environment of the event to the second neighbour atoms, or even third, however that just pushes away the same basic problem. That is, the atom B does not *see* the state it should diffuse into before it diffuses.

From the Test 2 in Sec. 5.1.2 we can see that the problem of uncontrolled clustering of atoms B can be solved by shifting the central atom of the event to the atom A, which can be thought of as the final state of the diffusion of atom B, however at the expense of losing information on the local environment of atom B prior to the diffusion.

Therefore, we combine the local environments around atom B from the initial and final states, as described in Chapter 4, Sec. 4.3.3, called the extended envi-

ronment. Like this we gain more control over the events, meaning we can control all the different scenarios of clustering events. Some examples of events with the extended environments are shown on Fig. 5.6. With the ability to control the energy barrier - and with it the probability - of each event comes also the fact that the number of possible different configurations increases. With the catalogue of

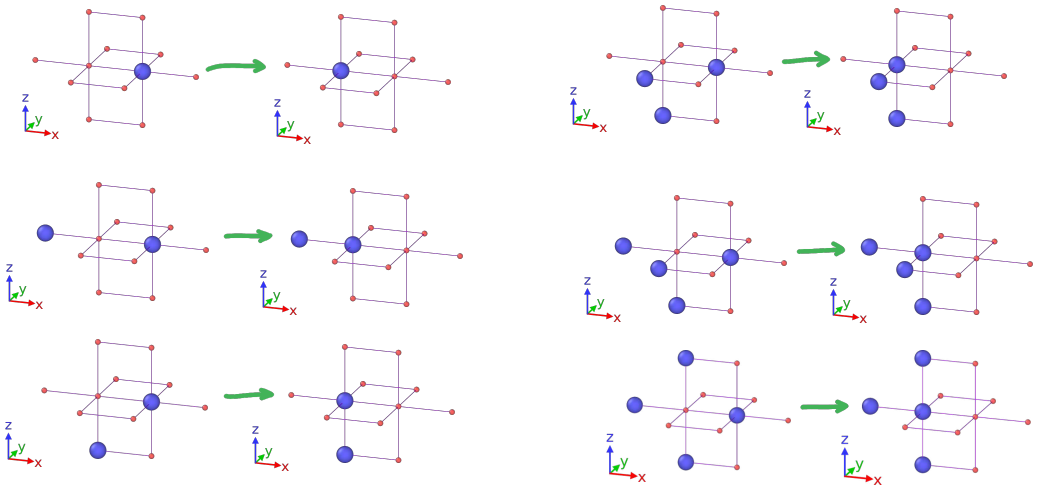


Figure 5.6: Some examples of events within the extended environment, which is made of combined atoms from initial and final neighbourhood of the central atom.

events depicted on Fig. 5.6, the kMC simulation ends up in similar situations as final states of Fig. 5.3 in Test 1, however with the major difference that in the present case, the clustering is under control. For example, by imposing a low barrier (high probability) to the single blue atom diffusion event, the simulation makes many steps to cluster any blue atoms. We can also play with the barriers of head-on clustering, and the diagonal-on clustering events (middle left, and bottom left on Fig. 5.6), which seems to slightly alter the shape of resulting clusters.

From the point of view of the event symmetries, the following happens. The diffusion of single blue atom, the event depicted at the top left of Fig. 5.6, has 6 associated symmetries, which are consistently checked. Whenever there is another blue atom at the second neighbour position, which is the left-most red atom in the initial state (see Fig. 5.6, top left), that particular symmetry (direction) becomes impossible to execute, while the other 5 remain possible. This capability is achieved due to the combination of consistent symmetry checking, and the use of extended local environments within the kMC.

5.2 O and O_2 diffusion in Si

To move away from toy models, we look into the O and O_2 diffusion in Si. The problem is in a way similar to the toy model from Sec. 5.1, however there are now only two atoms capable of diffusion - there are two O atoms in the simulation box, in a crystalline Si system. All the data used comes from concise, devoted DFT calculations, such that elastic distortions are present in the system at the initial stage of the simulation, and within all events.

The isolated O atom is situated in the bond between two Si atoms, and can diffuse into any of the six neighbouring bonds of the same type, three per Si atom on each side. Fig. 5.7 shows the Si atoms in red, and O atoms in blue, there are three possibilities of O atom diffusing over the central Si atom, into a neighbouring Si-Si bond. Since our shape matching algorithm in the kMC will

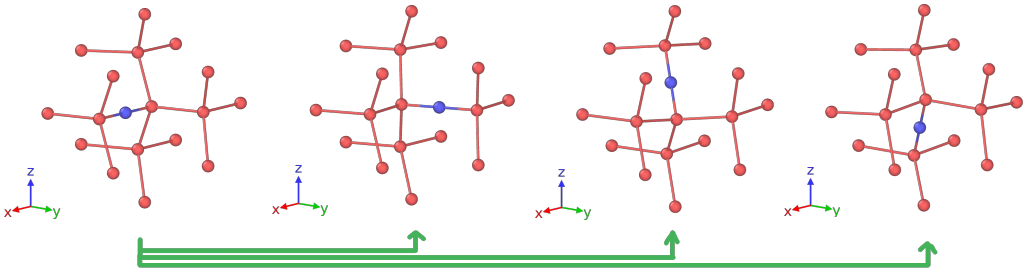


Figure 5.7: Single O atom (in blue) can diffuse into any neghbouring Si-Si bond (in red). On this figure the event is centered on the Si atom, thus generating the three unique events around that atom suffices to cover all six states into which the O atom can move.

recognize identical environment also from the Si atom on the opposite side of the Si-O-Si bond than the events shown in Fig. 5.7, all six possibilities for O diffusion are covered. The different directions of O diffusion depicted on Fig. 5.7 are all generated by our symmetry check procedure described in Sec. 4.3.5, from a single DFT calculation (NEB).

Two O atoms that are close together can rotate their configuration around a common Si atom, such that one O atom diffuses, this is depicted on Fig. 5.8(a), or reorient themselves without diffusing, Fig. 5.8(b). They can also diffuse together to a neighbouring Si atom, as in Fig. 5.9. And with some probability, they can also dissociate into two isolated O atoms, or conversly, associate from two isolated O atoms, shown on Fig. 5.10.

We launch a simulation with the event catalog containing these events, on an initial configuration of the box that contains two O close together, forming O_2, Fig. 5.11 (left). The O_2 successfully dissociates into two individual O atoms,

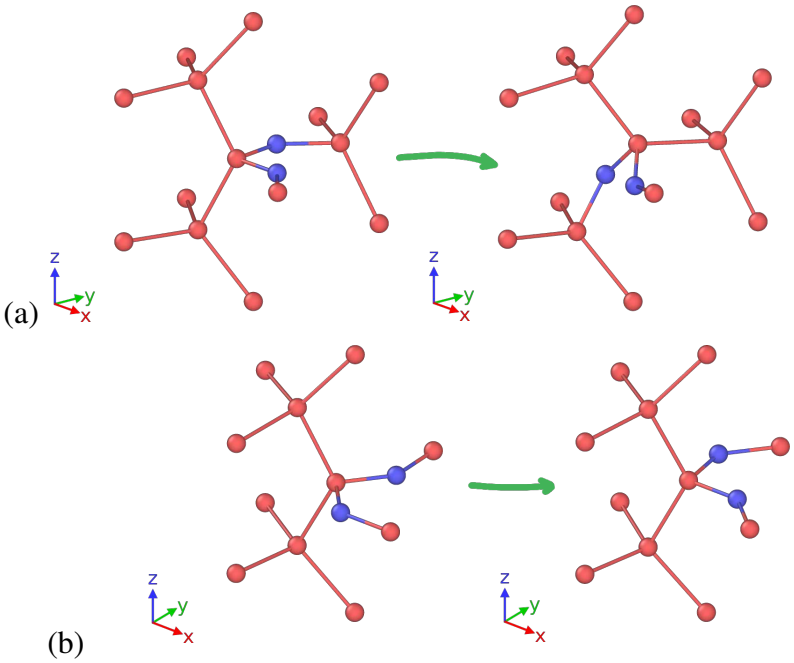


Figure 5.8: O_2 movements around common Si atom: (a) rotation, (b) reorientation.

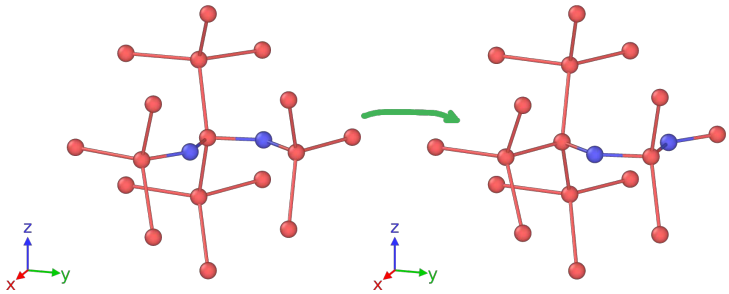


Figure 5.9: O_2 diffusion to another Si atom.

which diffuse independently, Fig. 5.11 (middle), and sometimes come close together to momentarily form O_2, Fig. 5.11 (right).

5.2.1 Effect of elastic distortions

As observed in Fig. 5.11, the structure of the Si crystal gets unphysically distorted by the O propagation. The reason for this artifact and an attempt at its quantification are discussed in this section.

In order to focus on the effect of elastic distortions in the system, and/or the

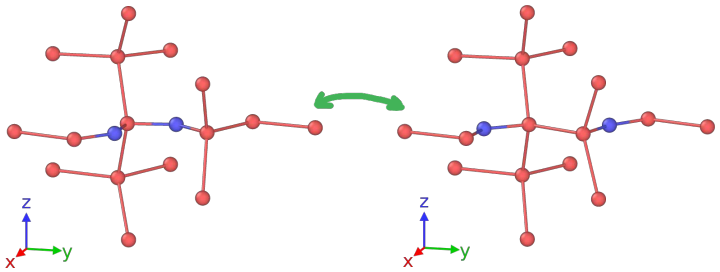


Figure 5.10: O_2 dissociates into two individial O, or two individual O associate into O_2.

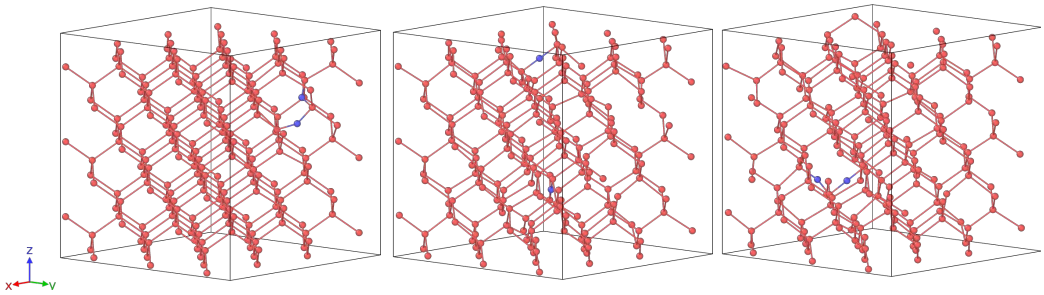


Figure 5.11: The initial state of the simulation (left) is with an O_2 formation, which dissociates into (middle) two individually diffusing O atoms, which can also (right) associate into O_2. Comparing the images on the left and right, notice the distortions of the Si crystal structure. They are an unphysical artifact related to the elastic distortions included in the event, described further in Sec. 5.2.1.

event configurations, we take the simulation box with a single O atom in the Si crystal network, with 217 atoms in total. The simulation box has been relaxed with DFT. For the event we take just the single O atom diffusion. The event data comes also from a DFT calculation, therefore all elastic distortions are included both in the kMC simulation box, and in original event data.

As we generate the event configurations that will be written into the event catalogue for kMC, we choose the `rcut_mode=neig` option (see Sec. 4.3.3). This allows us to choose the atoms that will get included in the event through the neighbour-cutoff parameter `rcut` in form of an integer, specifying the number of neighbour shells to be included. Another option for controlling which atoms get included in an event is the minimal-move threshold `small_move_thr`. We set `small_move_thr` relatively high, such that no atoms join the event due to this parameter, and we vary the neighbour-cutoff parameter `rcut`. Like this the number of atoms included in the event will depend only on the choice of the cutoff. We choose the values of cutoff to be 1, 2, 3, and 4. The event configurations S^E_{ini}

corresponding to these choices are shown on Fig. 5.12.

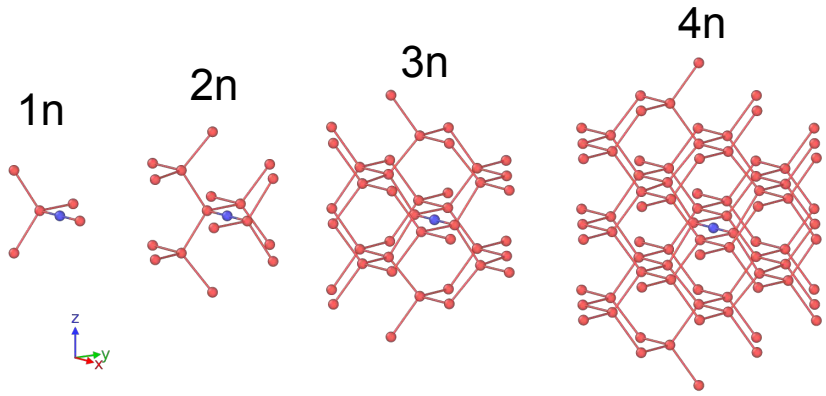


Figure 5.12: Initial state event configurations S^E_{ini} for the single O diffusion, with respective choice of the neighbour-cutoff parameter.

The simulation box R^{sys} at its initial state includes all elastic distortions around the O atom. As the kMC applies an event, only a portion of all atoms get moved. The portion depends on the cutoff of the event. Therefore if the event cutoff is too small, not all the atoms that originally contribute with elastic distortions will get moved, which introduces distortions into the system. Contrary, if the event cutoff is large enough, all elastic distortions should get propagated with the application of an event, and the distortions should not apprear

For this experiment, we make four different simulations of the same initial box. Each simulation includes one event, one of the four from Fig. 5.12, which is propagated for 500 kMC steps. Then we attempt to quantify the distortions present at the final configuration. The expectation is that the smallest event (1n) will introduce lots of distortion since it does not propagate many atoms around, and the largest event (4n) will introduce only a small amount of distortion.

The initial state of the simulation, and the four final states corresponding to the four simulations are shown on Fig. 5.13. From the point of view of the O diffusion, the four simulations are comparable, since all 6 possible directions of O diffusion are possible at each step. The diffusion of O atom is plotted in terms of the distance travelled from its initial position versus the kMC steps in Fig. 5.14, the simulation time here is not relevant, since all moves have identical probability.

The distortions present in the initial configuration, and in each of the four final configurations can be seen from the radial distribution functions, shown on Fig. 5.15. A radial distribution function is expected to show sharp delta-function shaped peaks for a crystalline solid, and more broad peaks for more disorganized structures.

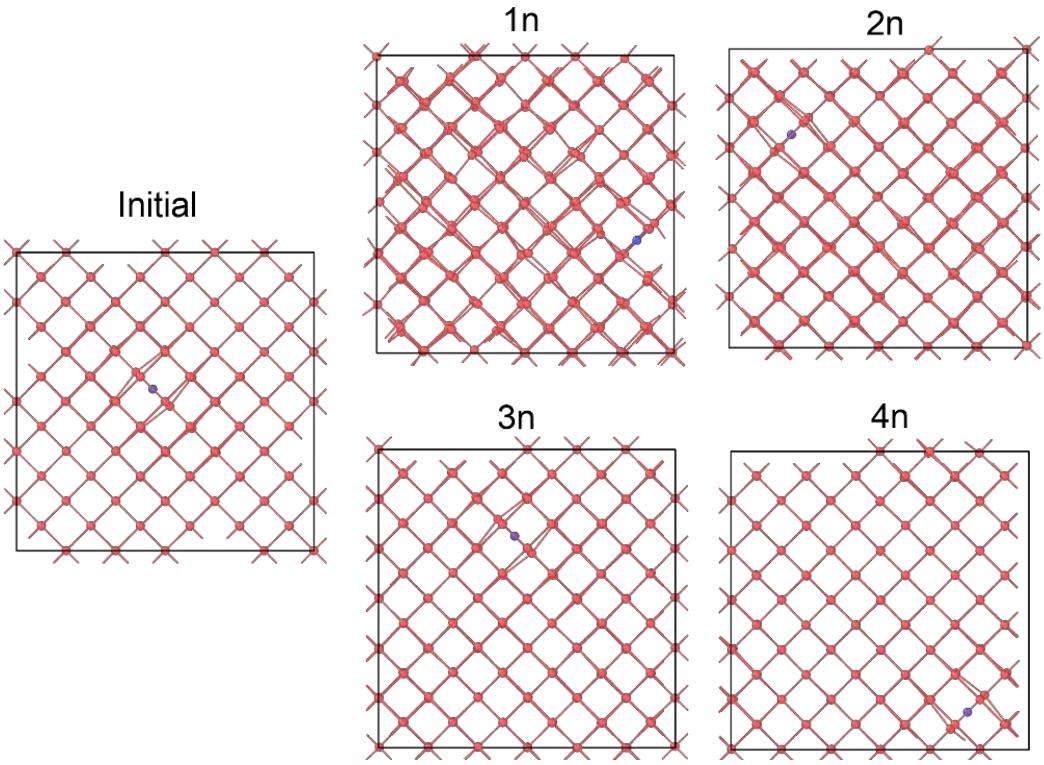


Figure 5.13: The initial state of kMC simulation, and the final states of four simulations with different cutoffs for O diffusion. A side view is chosen for simpler visualization of the distortions.

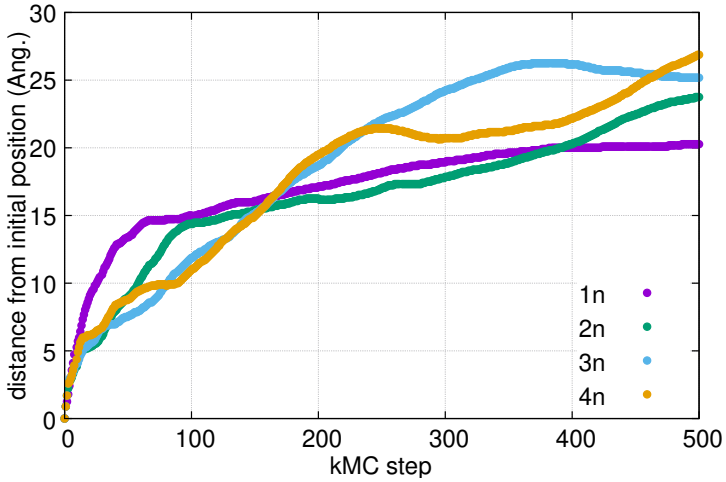


Figure 5.14: Diffusion of O atom in terms of the displacement of O atom relative to its initial position, for the four simulations.

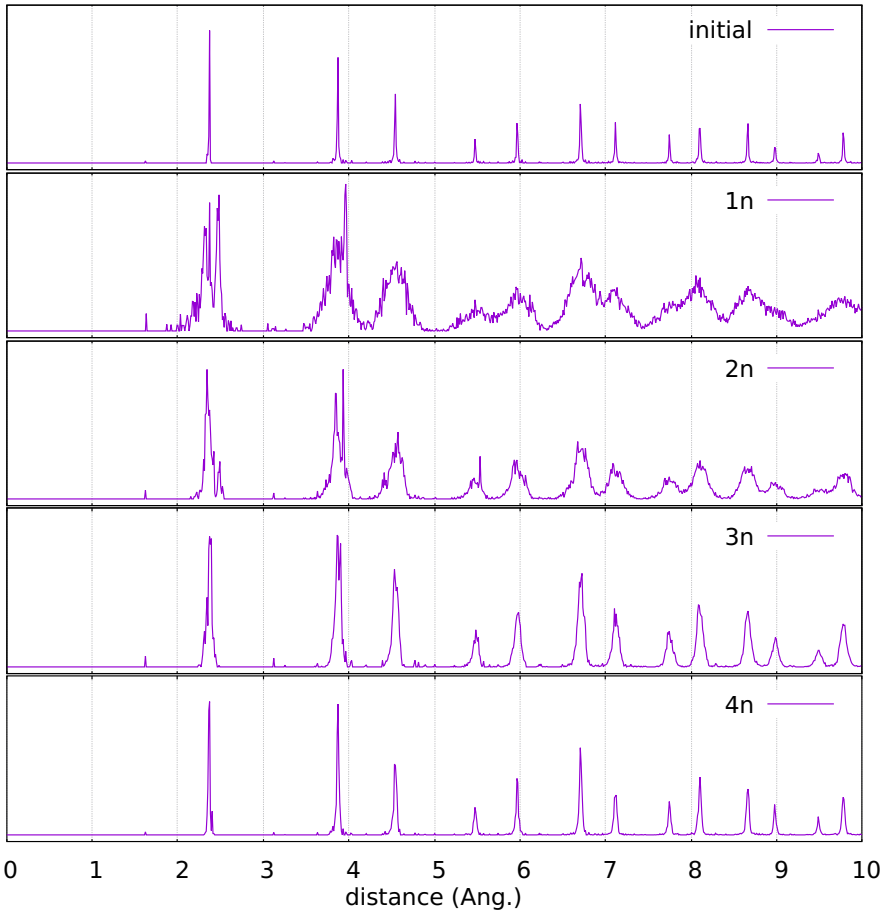


Figure 5.15: Radial distribution functions for the initial configuration and the four final configuration of each simulation. The distortions appearing during the kMC simulation can best be seen for the 1n case, as expected. The units of the y-axis are not important in this image, and have been rescaled for each plot.

In order to actually quantify the distortions, we calculate the forces present in the system with DFT, with identical parameters for each of the boxes. For this calculation, Quantum Espresso [124] has been used, with Projector-Augmented-Wave (PAW) pseudopotentials with 50 Ry cutoff for the energy, and Gaussian smearing of 0.005. The resulting total forces are reported in Table 5.1.

If the total force calculated is a good characterization of the distortions present in each system, then they follow the expected behaviour. They are largest in the 1n simulation, and smallest in 4n system, with respect to total force in the initial system.

From this subsection we conclude that the size of the event will have an effect

Simulation	Total force (Ry/au)
initial	0.003
1n	1.667
2n	0.680
3n	0.366
4n	0.147

Table 5.1: Resulting forces for each simulation.

on the distortions of the system during the kMC evolution. They arise due to disparity between the distortions initially present in the system, and the portion of atoms that actually get propagated with each event. As the elastic distortions can be very long-range ($1/r$) the most general way to fix this problem seems to be to compute the forces after each kMC step and let the system relax. However, that computation can be computationally prohibitive for large systems, particularly when using first-principles energy and force engines.

To mitigate, we use a modified IDPP (as described in Sec. 4.5), to check after every step of the simulation that the configuration of atoms involved in the event is distorted within a threshold from the final state configration of the event. In fact, without using this technique the 1n simulation from this subsection is generally not even possible to reach 500 kMC steps, the distortions get too large for the shape matching to identify the event structures.

The second conclusion we can make is that we would like to include as many elastic distortions as possible into the event, which implies generally a relatively big number of atoms and extent of the event local environment. This also implies that in the case of a second O atom present in the system, all symmetrically unique configurations that include the two atoms within that local environment need to be known, otherwise the event catalog is obviously incomplete.

5.3 Si interstitial diffusion in silicon

In this example we use the kMC to simulate the diffusion of an interstitial atom in silicon. Within this simulation, the interstitial configuration is possible in two distinct states, called the dumbbell, and the hexagonal configurations. The possible transitions that are diffusive are transitions between these two configurations. There are four symmetry-non-equivalent *canonical* events, each of them is generated from a single DFT calculation (NEB) data. They are described in more detail below.

The hexagonal configuration can diffuse into a dumbbell configuration, this transition is depicted on Fig. 5.16. The hexagonal configuration (in the center of

Fig. 5.16) is characterized by a single atom in the non-crystalline position, placed at the center of one of the four hexagon formations in the diamond crystal unit cell. The dumbbell configuration (in the six corners of Fig. 5.16) is characterized by two atoms on non-crystalline positions, the interstitial atom from hexagonal configuration moves slightly towards one of the atoms in the hexagon, which pushes this atom into the neighbouring hexagon, so the dumbbell configuration is characterized by interstitial atoms on similar off-center positions of two neighbouring hexagons. This event can occur in six different directions which were all found by the symmetry checks described in Sec. 4.3.5.

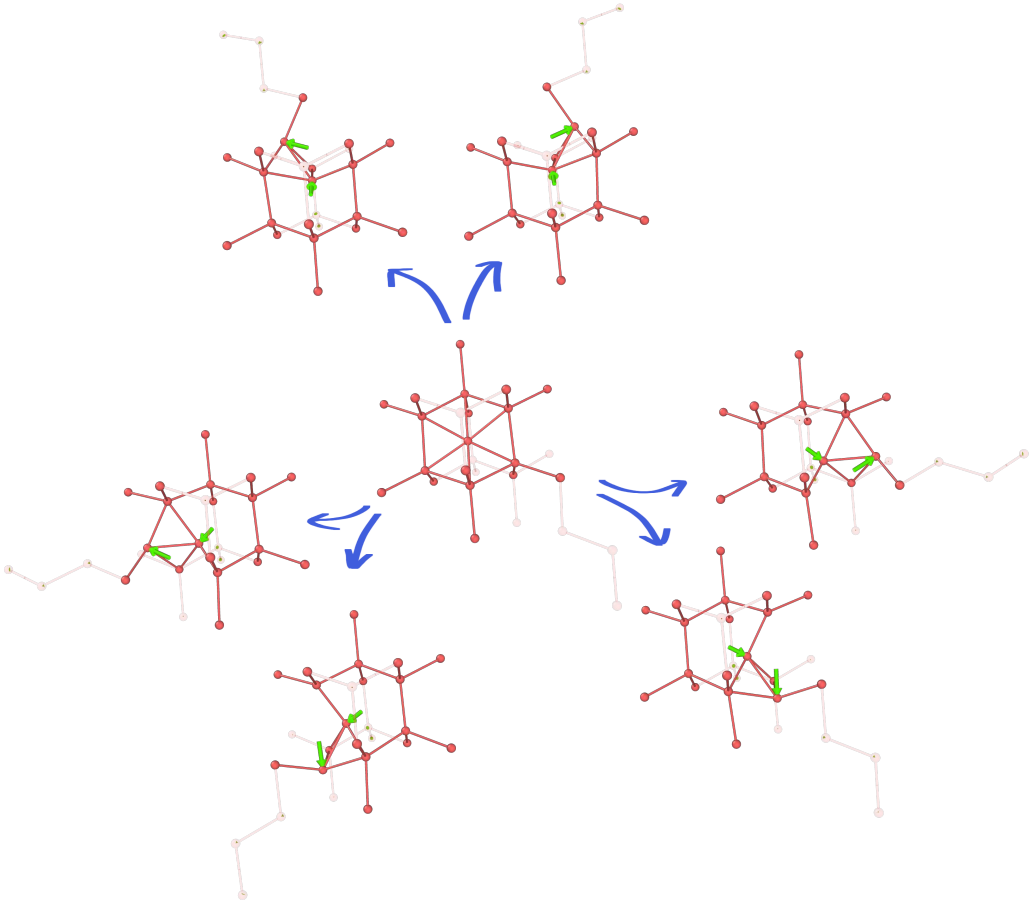


Figure 5.16: Hexagonal to dumbbell transition. Hexagonal configuration in the center, and dumbbell configurations in the six corners. The atoms of the primary environment are shown in solid colour, the atoms of extended environment are transparent. The green arrows indicate the movement of each atom with respect to the initial state. The six directions of this event are generated by the algorithm for finding the event symmetries θ (Alg. 8), from single event data.

The dumbbell configuration can transform back into hexagonal. This is depicted on Fig. 5.17. The atom at the left side of the dumbbell moves into a crystalline position, and the atom at the right of the dumbbell moves into interstitial hexagonal position. Notice that this event can occur on both atoms in the dumbbell configuration. The local environments as seen from these two atoms are symmetric over the rotation around the axis perpendicular to the plane of the image in Fig. 5.17. In our kMC simulation, both of these atomic sites in the system will be assigned the event, in the two possible directions respectively. Thus making four possible escape paths from the dumbbell configuration into the hexagonal.

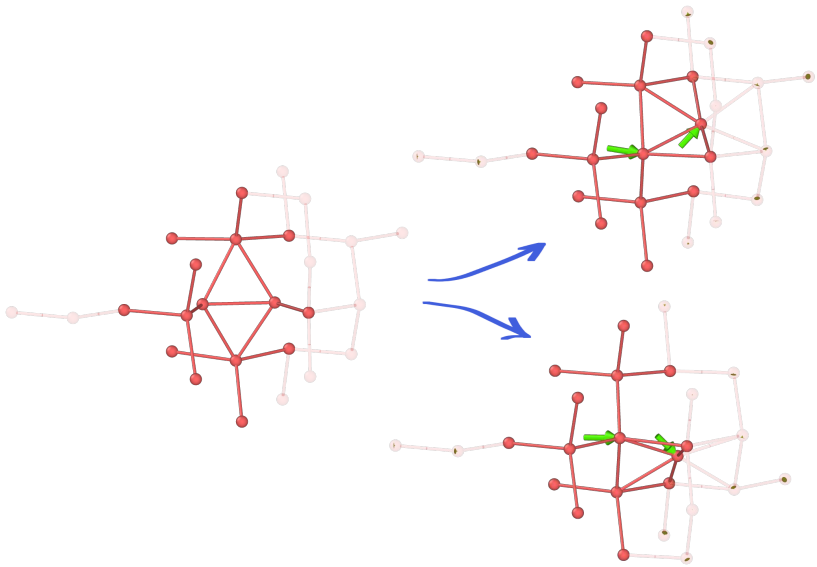


Figure 5.17: Dumbbell to hexagonal transition. The atom at the left side of the dumbbell moves into a crystalline position, the atom at the right of the dumbbell moves into a hexagonal interstitial configuration. The atoms of the primary environment are shown in solid color, the atoms of extended environment are transparent.

The hexagonal configuration can diffuse into another hexagonal configuration in two ways. The first one is called *inside*, depicted on Fig. 5.18. The interstitial atom moves from one hexagon ring to another, within the same diamond structure of the unit cell. This event has three possible directions.

The second way of hexagonal-hexagonal diffusion is called *outside*. Depicted on Fig. 5.19, the interstitial atom diffuses to the hexagonal structure of the neighboring diamond structure of the unit cell.

Taking all of the events described above, the event catalog has twelve events. These events have been generated from four DFT calculations (NEB), one for

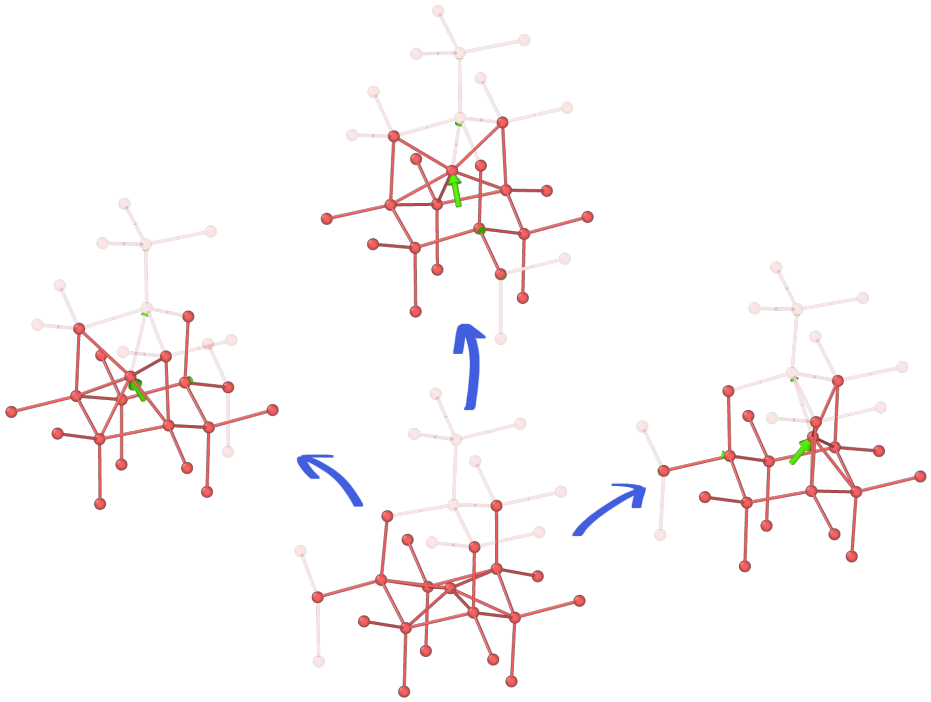


Figure 5.18: Hexagonal to hexagonal *inside*. The interstitial atom moves from one hexagon to another, within the same diamond unit cell structure.

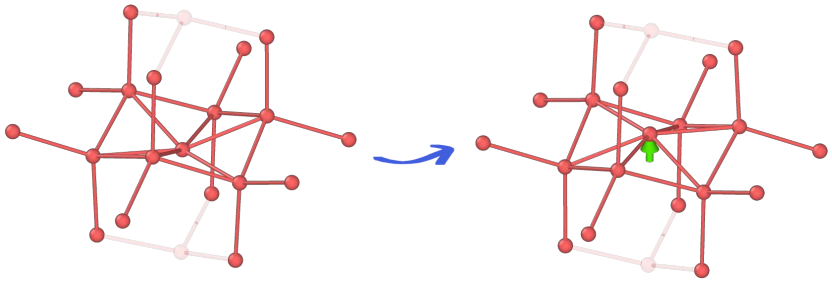


Figure 5.19: Hexagonal to hexagonal *outside*. The interstitial atom diffuses to neighbouring hexagon of the diamond structure.

each event. The energy barriers associated to each event are given in Table 5.2. These energy barriers are given as input with each event for our kMC.

The simulation box in this example is a cube with the side length 64.8 Å, and has 13825 atoms of Si, which are for the most part crystalline. In the initial state of the simulation, there is one interstitial defect in the dumbbell configuration. The initial configuration is shown on Fig. 5.20 from two viewpoints, and a close-up of the dumbbell interstitial defect in blue circle.

Event	E_{ac} (eV)
hex-dum	0.261
dum-hex	0.278
hex-hex in	0.156
hex-hex out	0.015

Table 5.2: Activation (barrier) energies for each event.

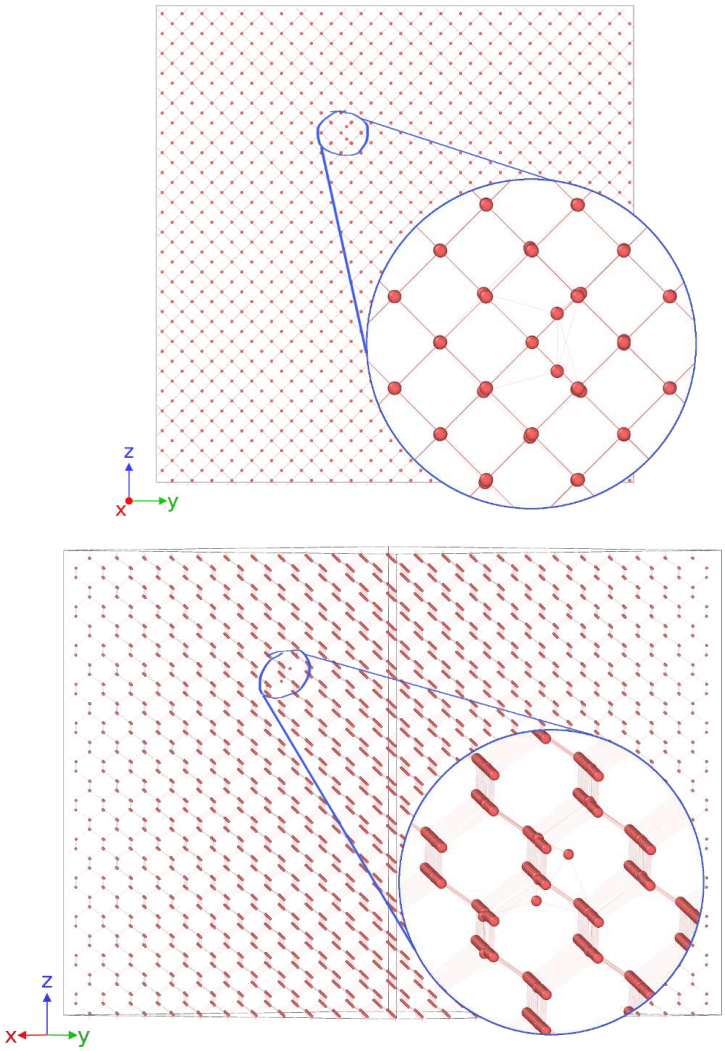


Figure 5.20: Initial state of the Si interstitial simulation, with 13825 Si atoms, one interstitial in the dumbbell configuration from two viewpoints. The dumbbell interstitial is shown in the blue zoom-in.

We characterize the distortions accumulated due to not accounting for all elastic distortions in the events exactly (see Sec. 5.2.1) by comparing the radial distribution functions at the initial and final state of the simulation. After 5000 kMC steps, the radial distribution functions are shown on Fig. 5.21, we see that the dispersion of the width of peaks is very small, which implies that there is some small amount of distortion in the atomic positions at the final state, but is largely negligible, and does not affect the shape matching process used for recognition and execution of events. The threshold for equivalence of structures for the IRA algorithm used is `dist_thr=0.3` Å.

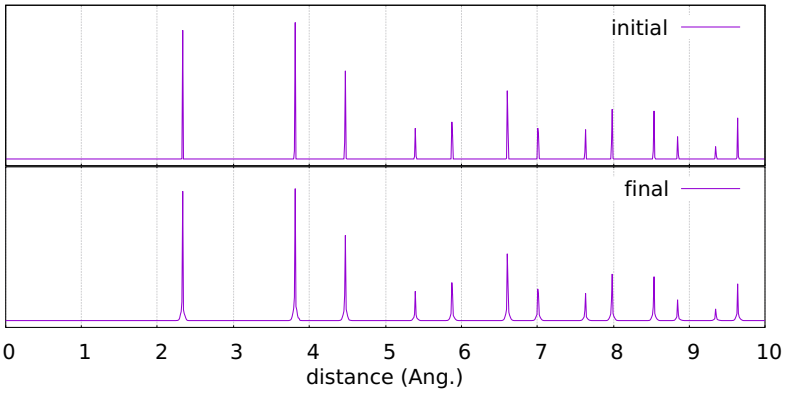


Figure 5.21: Radial distribution functions for the initial and final states of the simulation. The dispersion of the width of each peak is associated to the atomic distortions that arise due to the error buildup of not exactly accounting for all elastic distortions in the events.

We can calculate the mean displacement of the trajectory, by computing the running average of the displacement of the current simulation step from a reference. That is usually done by taking the initial state of the simulation as the reference structure A, and the current simulation step as structure B. Then we calculate the displacement of the current simulation step, with respect to the reference, by computing $RMSD(A, B)$ as defined by Eq. (2.6). Doing this for each simulation step in a running average way, we obtain the mean displacement. Equivalently, we can take just the last step of the simulation, and compute the $RMSD$ with reference to the initial state. The value computed is often thought of as a radius of a sphere, representing the mean distance an atom can diffuse in a given time unit.

There is however a problem with computing the $RMSD$ in this way, since it is not invariant over the permutations. The problem arises when the index of the diffusing atom changes. In that case, the sum in $RMSD$ is made up of a number of vectors, whose norms all contribute to the final value. In fact, every time the index of diffusing atom changes, a new vector will appear in that sum.

To illustrate the problem, we use the present Si interstitial diffusion example. The main diffusion happens through the diffusion of a hexagonal interstitial into another hexagonal. Here there is no problem, since the interstitial atom remains the same after the diffusion process. However, the diffusion into a dumbbell configuration produces two atoms that are not in the crystalline positions (or very close to their reference crystalline positions). And when the dumbbell diffuses back into a hexagonal, one of these two atoms becomes crystalline (or very close to its reference crystalline position, it shall be called crystalline in the following), while the other becomes the hexagonal interstitial. Thus, it can happen that an atom that has originally been a hexagonal interstitial, becomes crystalline after the diffusion through dumbbell, and an atom that has originally been crystalline, to become a hexagonal interstitial. This effectively means that the index of the diffusing atom has changed. This process is illustrated in Fig. 5.22.

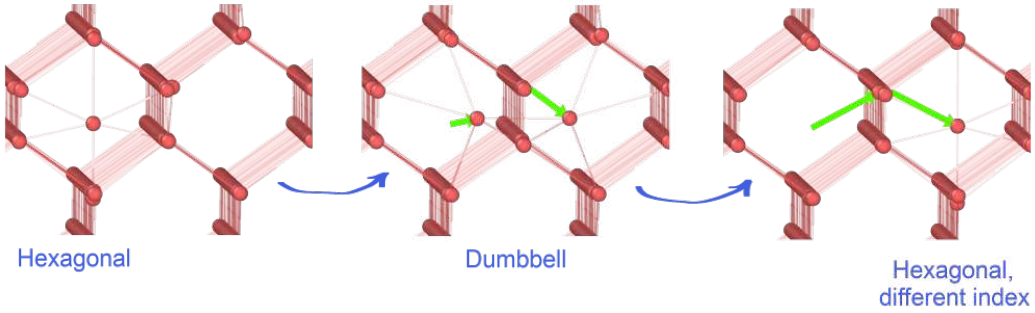


Figure 5.22: The process of change of the index of the diffusing atom in the hexagonal configuration, through the dumbbell configuration. Green arrows show the atomic displacements with respect to the left image.

A question naturally arises. How can we compare values related to the diffusion, obtained by computational methods, to those obtained by experimental techniques? Since the experimental techniques do not have an insight into each atomic move, they cannot resolve the history of the states between the initial measurement, and the final measurement, since the atoms of equivalent species are indistinguishable.

If we apply this reasoning to our simulation, more specifically the computation of the $RMSD$, we see that instead of what is computed by directly applying the $RMSD$ formula (all the displacement vectors in the sum are shown on Fig. 5.23 left), one should take into account that the atoms are indistinguishable, and that there is no resolution on the intermediate history (single displacement vector shown on Fig. 5.23 right). We shall call the $RMSD$ computed as Fig. 5.23 (left) "direct" $RMSD_{dir}$, and the $RMSD$ computed with a single vector displacement from Fig. 5.23 (right) "invariant" $RMSD_{inv}$.

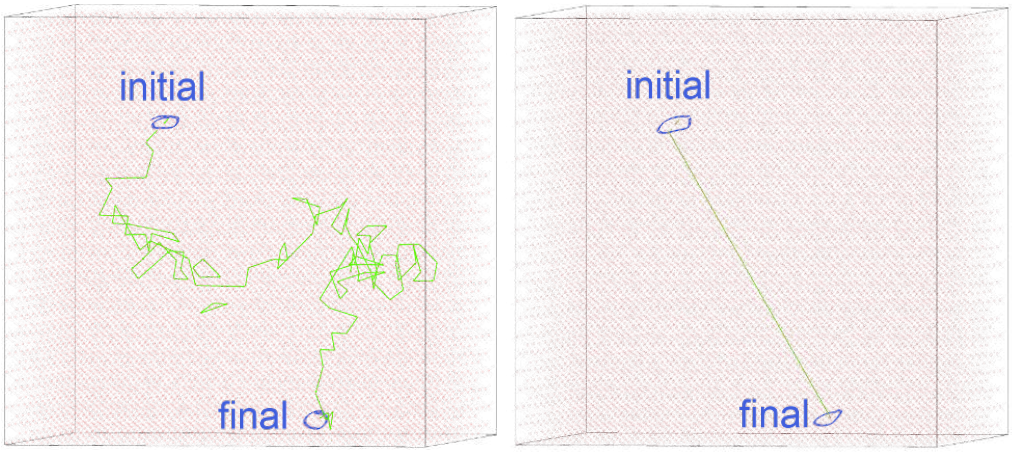


Figure 5.23: Displacement vectors taken into the $RMSD$ calculation in green. On the left: due to exchange of the index of diffusing atom, simply aplying the $RMSD$ formula from Eq. (2.6) includes a large number of vectors in the sum. On the right: the intermediate history of the simulation is generally not known, so it should not be taken into account when computing the diffusion, as the diffusing atom is not distinguishable from the others.

In order to compute the single vector displacement from the right side of-Fig. 5.23, we can take the final state of the simulation as structure B, and compute the permutations P_B, with reference to the initial state of simulation A, by the CShDA algorithm (Sec. 3.4). For large simulation boxes, this operation is quite slow, so doing it for every simulation step is not viable. Alternatively, we can follow a specific local environment through its graph hash value. So we calculate the distance between an atom with a specific hash value in the current simulation state, and an atom with identical hash value in the initial simulation state. This distance is equivalent to the invariant $RMSD_{inv}$.

The Fig. 5.24 shows the computation of $RMSD(A, B)$ done for each simulation step. The value of the direct $RMSD_{dir}$ by applying the definition from Eq. (2.6), and by following a specific graph hash value of the hexagonal interstitial atom, to compute the invariant $RMSD_{inv}$ of Fig. 5.23 (right). Both computations are done for the same simulation run.

The frequency of the change of index of the diffusing atom can be seen by looking at the direct displacement vectors during the simulation, as in Fig. 5.23 (left), from simulations at different temperatures. We take three different temperatures, $T = 300$, $T = 600$, and $T = 900$ K, and plot the direct displacement vectors, shown in green on Fig. 5.25 for the three temperatues. Each of the images comes from a single simulation of 5000 kMC steps at given temperature.

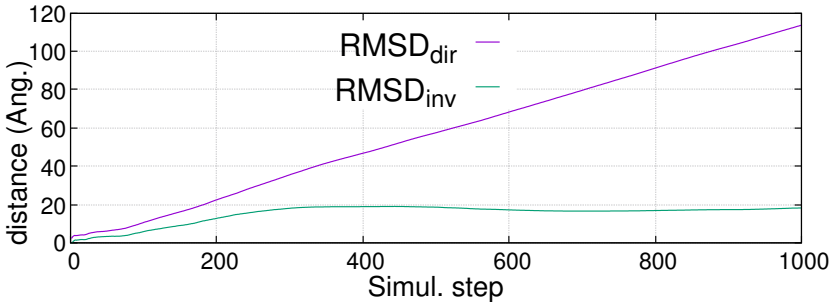


Figure 5.24: Computations of direct $RMSD_{dir}$, and invariant $RMSD_{inv}$, during 1000 steps of the kMC simulation.

For better representation of the event statistics, the number of steps should be of a much higher number, and the simulation box should be made bigger. This example can however give a sketch of what is happening. The vectors shown on Fig. 5.25 are the displacement vectors of each atom during the simulation. If the diffusion happens on the same atomic index for several steps, it is represented by a longer straight vector, and if the index of the diffusing atom changes, a new vector is started at the current location.

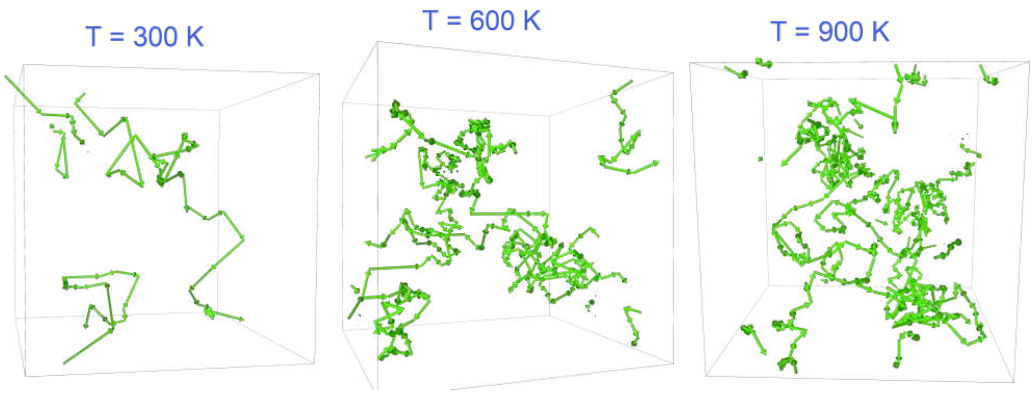


Figure 5.25: Direct displacement vectors of the atoms. Each new vector represents a change in the index of diffusing atom. The atomic positions are made invisible for clarity.

We see that the dumbbell event, which is responsible for the change of the index of diffusing atom happens much less often in the $T = 300$ K simulation, as expected judging from the energy barrier of that event in Table 5.2.

5.4 Gas sensor

The main purpose of the example in this subsection is to show the ability of our kMC software to deal with events with non-constant number of atoms. Such events are adsorption and desorption events.

The system of simulation is a slab of SnO_2 crystal, which comes in contact with a CO gas. In this example we include the following events:

- adsorption event A, where a CO molecule adsorbs on the SnO_2 surface;
- a diffusive event B where the CO molecule reacts with the surface, with the possibility to be executed also in backward fashion as $-B$;
- and desorption event $-A$ where the CO molecule desorbs from the surface, which is the adsorption event executed in backwards.

All the events are shown in Fig. 5.26, where the Sn atoms are colored red, the O atoms blue, and C atoms yellow

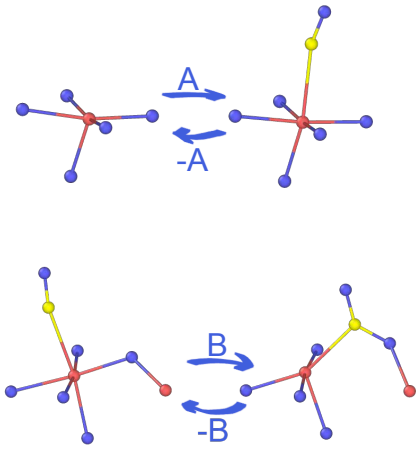


Figure 5.26: Events included in the simulation.

The chain of occuring events can be written as: $A \rightarrow (B \rightarrow -B)^n \rightarrow -A$, such that the succession $(B \rightarrow -B)$ can happen n times. After the desorption event, the surface is identical to the surface before the adsorption event. For this reason, the presentation of this simulation via animation/video would be slightly more convenient than a pictorial before/after showcase. Nevertheless I show some snapshots of a single simulation at different timesteps in Fig. 5.27.

The rate of adsorption is related to the partial pressure of the impinging gas, following Eq. (1.11) and Eq. (1.12). We test the adsorptions/desorptions with the following experiment. We set the probabilities for all events to the same value,

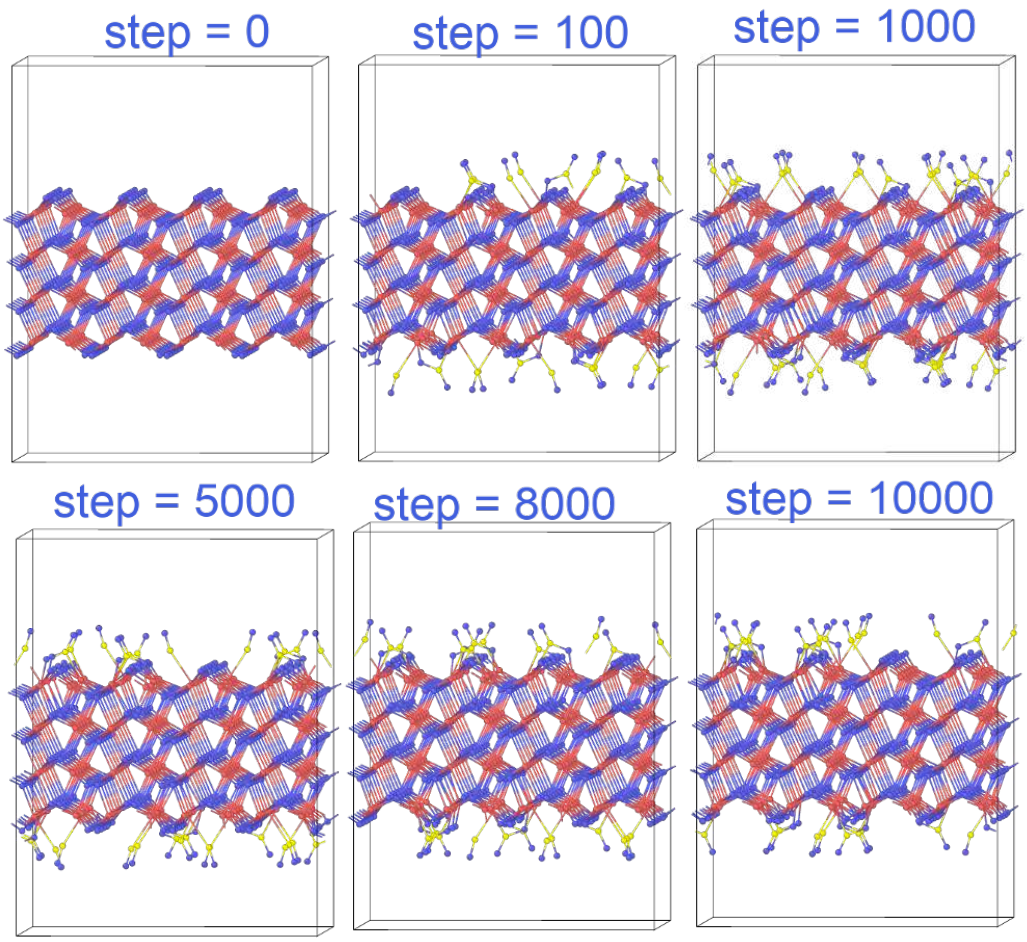


Figure 5.27: Some snapshots of the adsorption/desorption simulation.

and run the simulation at several different values of partial pressure, at equal temperature. There are in total 96 Sn sites where the CO molecule can adsorb with event A from Fig. 5.26. The number of CO molecules present at the surface at each simulation step is shown on Fig. 5.28. The values of pressure are given relative to each other, the first simulation is run at $P_1 = P_0$, the second simulation at $P_2 = 2P_0$, and third at $P_3 = 0.5P_0$. As expected, the number of CO molecules is higher at higher partial pressure of the CO gas.

There are other physically relevant events for this simulation, which together form the working principle of a CO gas sensor [125]. Those will however not be further discussed here, the main reason being that the interactions among events have not been calculated, which inevitably means an incomplete event catalogue for the kMC approach used here. And since the surface of simulation is relatively small, the interactions among events cannot be neglected.

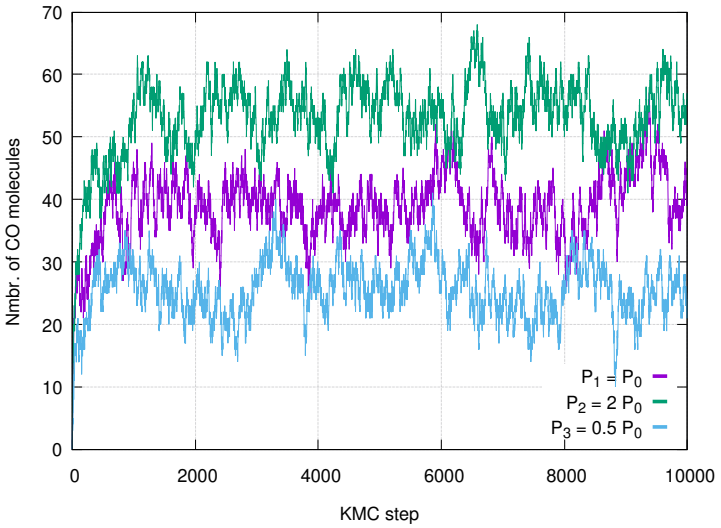


Figure 5.28: Number of CO molecules on the surface as function of the kMC step, for three simulations at different pressures.

5.5 O_2 adsorption on Si(100) surface

The Si(100) surface is known to reconstruct into a buckled structure, such that Si dimers are formed on the surface, which tilt in the direction of either of the two Si atoms in the dimer (see Fig. 5.29). The periodicity of these tilting dimers has long been a topic of investigations. Two specific periodicities, namely $c(4 \times 2)$ and $p(2 \times 2)$ have been shown to be energetically the most favourable [126]. In low temperature Scanning Tunneling Microscopy (STM) experiments, coexistence of these two reconstructions has been observed [127]. At room temperature, it is understood that the tilting of the dimers can change rapidly, causing the experimental methods to detect an "average" image, meaning a symmetric dimer with no tilting.

The mechanics of O_2 molecule adsorption and diffusion on Si(100) $p(2 \times 2)$ surface have previously been reported [128, 129, 130], and simulated with the OXCAD Lattice-kMC software [131]. In the OXCAD framework, the distinction of surface dimer tilting was disregarded, and the simulated surface was symmetric with no tilting. Another difficulty with OXCAD software is that each Si atom has an idealized on-lattice local environment, and the events are hard-coded in the software. Idealized on-lattice environment means that the growth of SiO_2 layer cannot happen in the predicted amorphous way, and that any strain-related effects cannot be accounted. The hard-coding of events in the software is a potentially major inconvenience for a user wishing to extend the catalogue of possible events.

The resolution of these problems has actually been the main driving force of this PhD work since the beginning.

Note that the first-neighbour Si-Si distances in this example are between 2.4 Å and 2.6 Å, and the Si-O distances are between 1.0 Å and 1.8 Å. The threshold for equivalence `dist_thr` in the rest of this section is in units of Å.

5.5.1 Buckling/tilting of surface Si-Si dimers

With the IRA shape matching algorithm implemented in our kMC software, there are a number of local atomic environments that can be distinguished at the Si(100) surface, depending on the `dist_thr` used for the equality of structures. Some of them are shown on Fig. 5.29, where the different colors indicate atoms with an equal local environemnts as characterized in four cases: graph hash only on top left, and three different thresholds with IRA matching algorithm. On the bottom right panel, the yellow and green environments are actually mirrored reflections of each other. We note that with using only the graph hash as descriptor, the tilting (buckling) of surface dimers cannot be distinguished, while with IRA we can distinguish the up- and down-Si atoms in the dimer (purple and pink, respectively, on Fig. 5.29). The thresholds used on Fig. 5.29 are quite high, for a real simulation we would like to use a value around `dist_thr=0.3`, in order to properly distinguish different local environments.

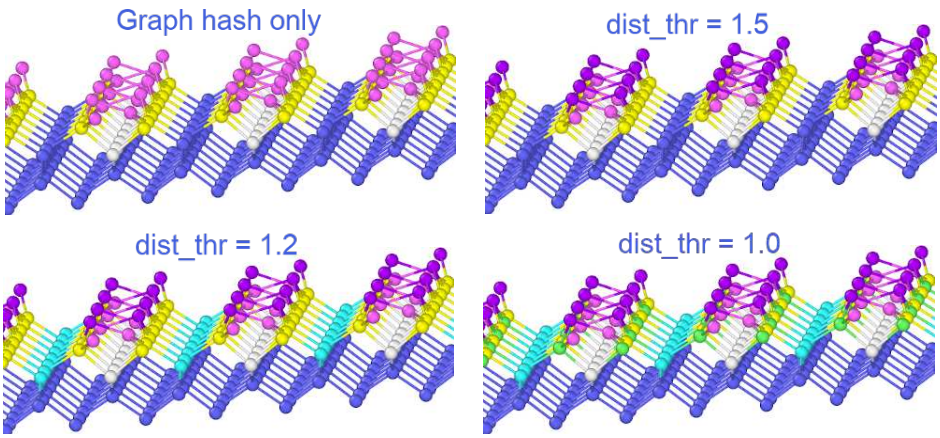


Figure 5.29: Local environments of Si(100) surface, represented with different colors, at different thresholds for equality.

5.5.2 Atomic distortions

To run the kMC simulation of O_2 adsorption on Si(100) surface, we have taken the DFT data of events published in Refs. [128, 130, 129, 131], Ref. [132], and some other unpublished DFT calculations. Already from looking at a small number of this dataset, we notice that the surface is quite responsive to any movement, in some cases an event on one site can induce distortions in Si atoms up to 3 neighbours away, see Fig. 5.30 and Fig. 5.31 for two examples. In Fig. 5.31, the displacement vectors are shown in green, and a particular Si atom which is 2 neighbours from the event site has marked the magnitude of displacement, which can be considered "small", but is very close to the order of threshold `dist_thr=0.3` that we would like to use. Any such atom that appears near the border of several events simultaneously, can be displaced several times due to events in its vicinity. By doing so, these kinds of "small" unphysical displacements tend to build up very fast. In fact, they soon build up into a value which is beyond the given `dist_thr` value, meaning that specific atomic site is not anymore deemed possible for any event. Even if it had never adsorbed any atom, in this particular case.

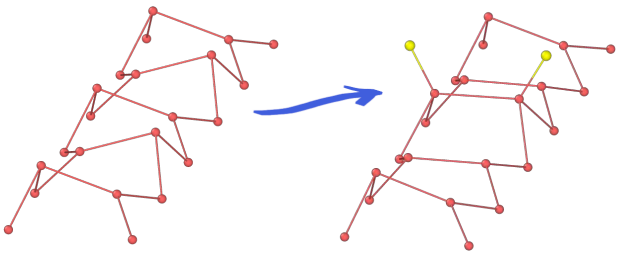


Figure 5.30: Adsorption event, where the Si dimers in red, parallel to the event site are heavily affected - they become flat. The adsorbed O atoms are in yellow.

Very often among the events in the dataset, an event on one dimer affects the Si atoms in the two neighboring dimers in the same dimer row. For example it might change the direction of their tilt, as in Fig. 5.30. This characteristic could probably be ascribed to the same physical process as the one that causes spontaneous flipping of the dimer tilts at room temperature. If we want to exclude such distortions in our kMC simulation, it means that each such event needs lots of curation by the user. For instance, we have attempted to ignore the atomic movements beyond the second neighbour, or beyond the first neighbour, or preventing specific atoms from moving at all, or even explicitly including or exluding particular atoms from the event, but the basic problem is that when two events happen close to each other, the execution of one can alter the environment of the other, rendering it impossible to detect and execute. It might be that event curation is the

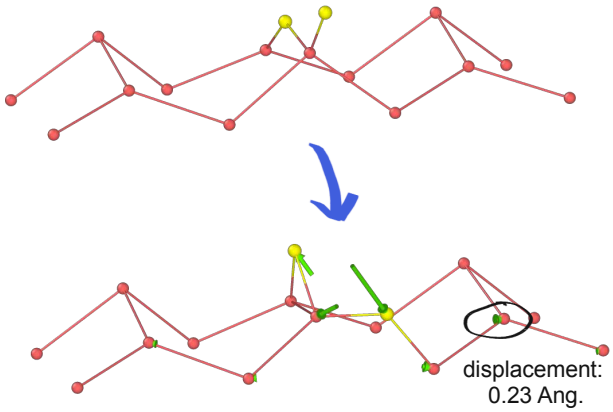


Figure 5.31: Diffusion event, with marked displacement vectors in green, and order of magnitude of distortion in a Si atom which is 2 neighbours away from event site.

fundamental mistake we are making, in the sense that all events should be able to modify the surface as they want. From this point of view, the simulation of O_2 adsorption on Si(100) surface by means of off-lattice kMC, is the prime example of the need for the relaxation of the forces, and an on-the-fly approach, able to correct the mistakes and explore the configuration space on its own.

5.5.3 Simulation with a subset of all events

In the following, I show a smaller subset of curated events from the O_2 adsorption event data, that lead to a curious problem at the stage of event identification, involving primary and extended local environments S and S^E, structural fragment detection, and graph isomorphism. The subset of events included is shown on Fig. 5.32 and Fig. 5.33, the data and schematics are taken from Ref. [131], and modified for present needs. The numbering of structures follows the numbering in the mentioned article. The events taken include configurations labelled in the original article from 1 to 8, excluding 6. Events added extra for the present simulation are adsorptions into structures 1 and 5. In Fig. 5.32 and Fig. 5.33, the atoms shown in a configuration are all atoms included in each event, and they do not strictly correspond to the atoms drawn in the schematic of the event.

Notice that the final state configurations of event $1 \rightarrow 3$ in Fig. 5.32 are different from the initial configuration of event $3 \rightarrow 1$, which should in principle be identical. In particular, there is an additional bond between two Si atoms in final structures of event $1 \rightarrow 3$. This is a consequence of attempting to curate the event by freezing some atoms, to prevent some structural disorder buildup.

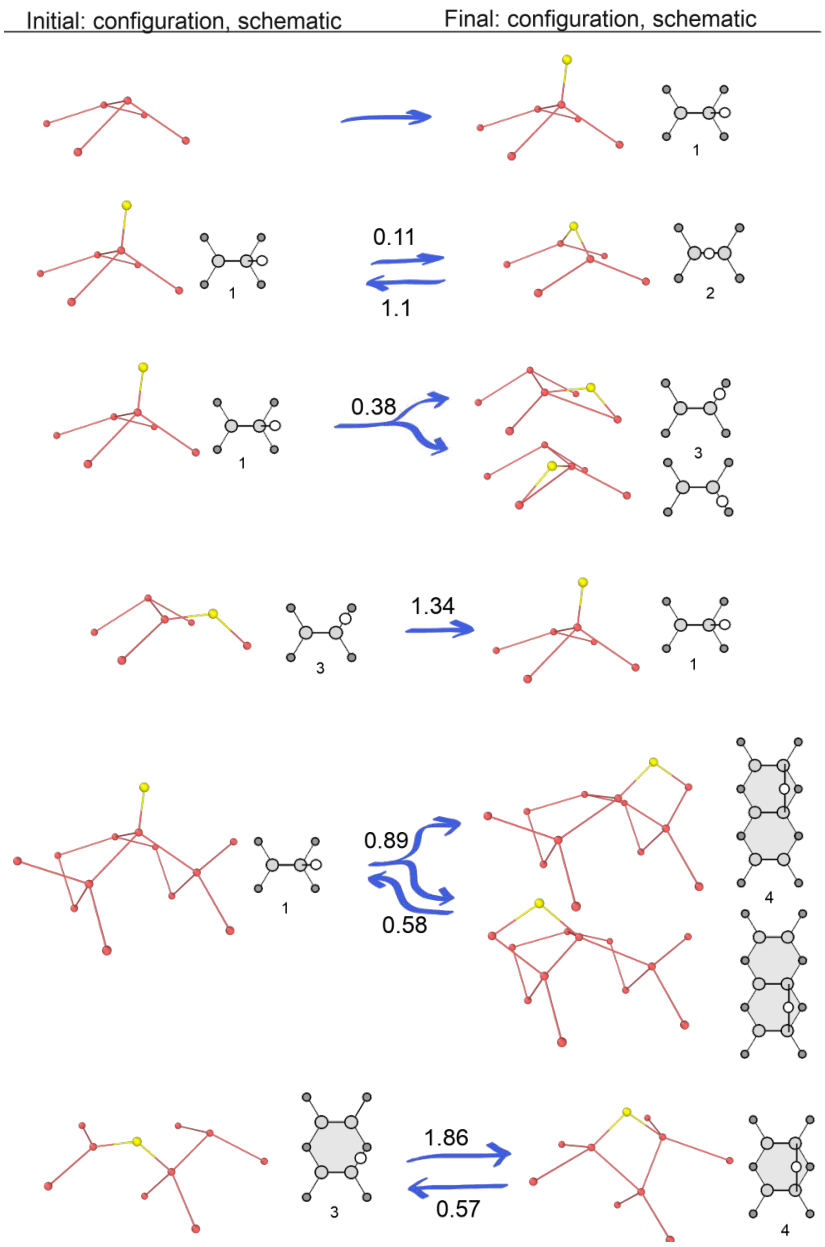


Figure 5.32: Events including structures from 1 to 4, showing the atomic configurations, schematic representations, and values of the energy barrier E_{ac} in eV above or under the arrows.

The direct consequence being that the graph hash value of final state of $1 \rightarrow 3$ is different than the graph hash value of initial state of event $3 \rightarrow 1$, which is obviously undesirable. Similar problem occurs also at events $7 \rightarrow 8$ and $8 \rightarrow 7$

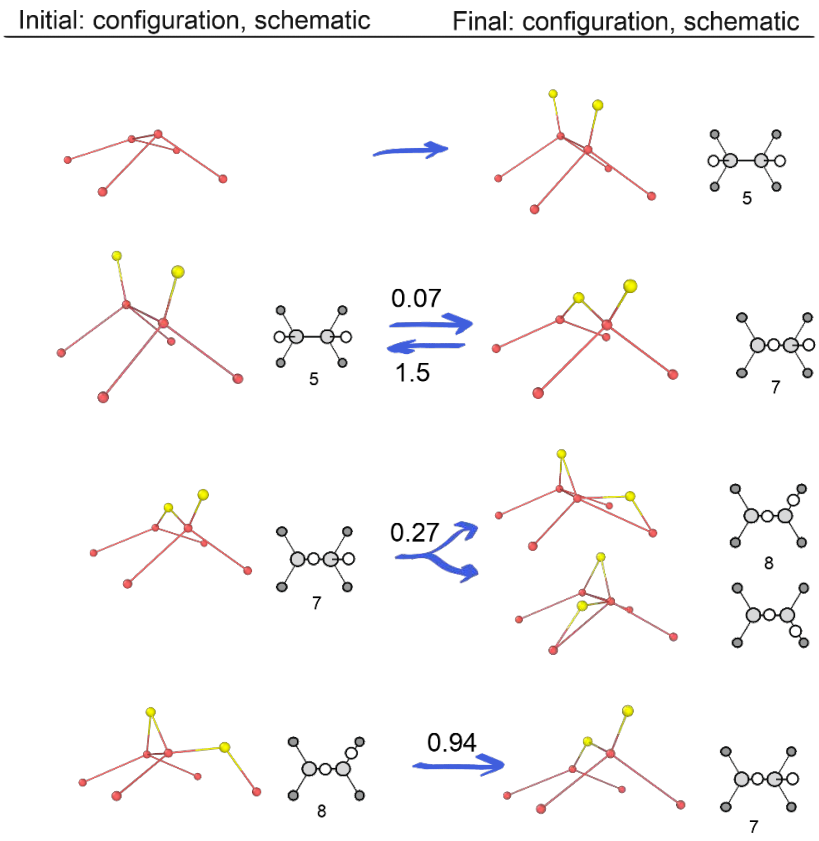


Figure 5.33: Events including structures from 5 to 8, excluding 6, showing the atomic configurations, schematic representations, and values of the energy barrier E_{ac} in eV above or below the arrows.

on Fig. 5.33. From the point of view of structural distortions, this is actually not a problem, since each event is applied as in Eq. (4.18), thus any non-moving atom contributes 0 distortions. For the actual simulation done here, the cutoff distances used for graph constructions have been altered such that this discrepancy does not cause problems, but I wanted to highlight it as one of the many difficulties of curating certain events, and the inconsistencies that may arise from doing so.

We run a kMC simulation with this event catalogue on a system of 2304 atoms, with 144 Si-Si dimers at the surface. At first glance, the simulation is quite sucecssful. At the end of 1000 kMC steps, there are still more than 200 possible events, indicating low overall distortions of the structure. However upon closer inspection of the atomic configuration, we notice structures that are not present in the event catalogue, such as shown on Fig. 5.34. The full chain of events that lead to this structure is shown on Fig. 5.35, with the dimer site circled in blue. We can

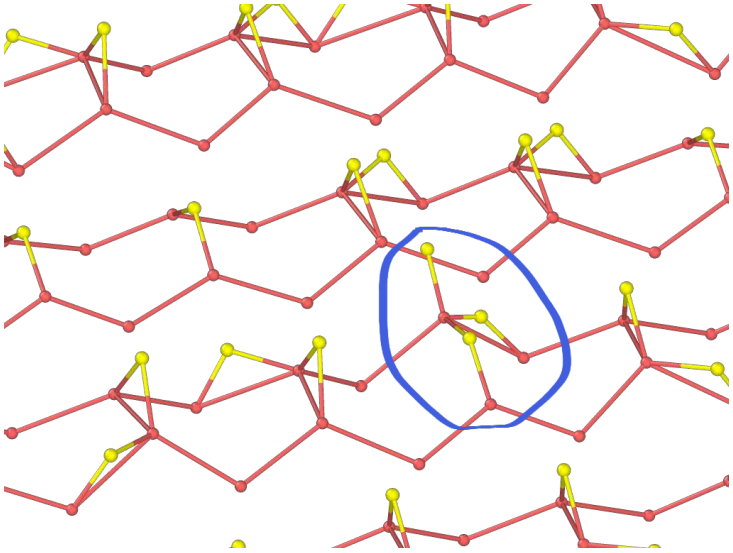


Figure 5.34: A structure not present in the event catalogue, that appears in the kMC simulation.

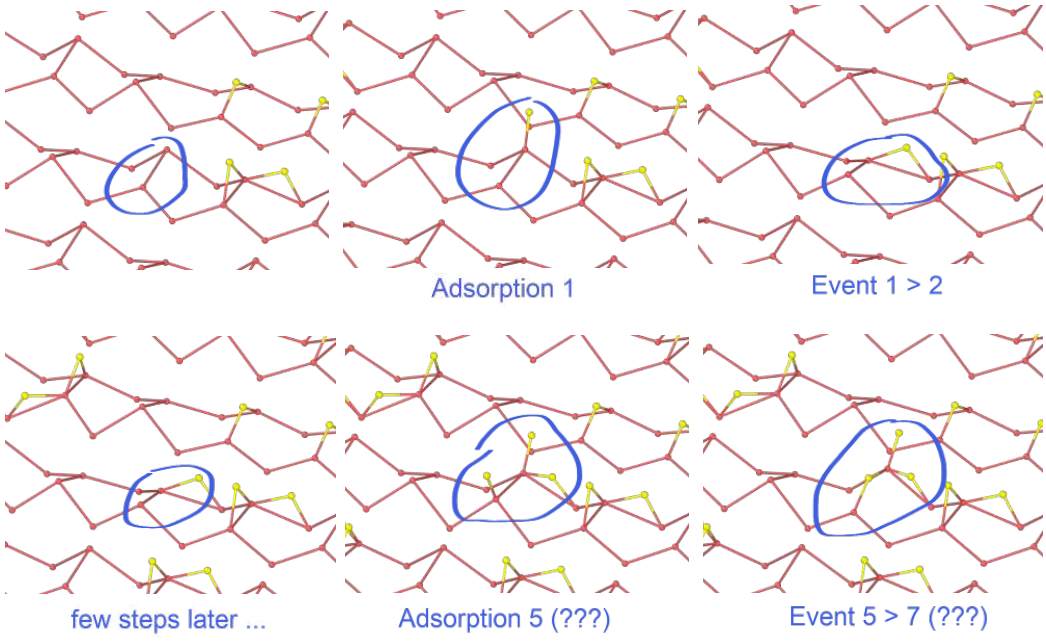


Figure 5.35: The chain of events leading up to an unknown structure from Fig. 5.34. The configurations marked with question marks (???) highlight the fact that these configurations are not present in the event catalogue.

pinpoint the cause of formation of this structure to the event, henceforth called

"adsorption 5", which is the first event listed in Fig. 5.33. In order to understand why this event happens, we follow the steps of the algorithm which identifies possible event sites.

The first stage of this algorithm is to filter atomic sites in R^{sys} by graph isomorphism through their hash values using NAUTY (see Sec. 4.4.1). The central atom of the "adsorption 5" event configuration S^E_{ini} is the Si atom at the "up" site of the dimer. The primary part S_{ini}, and the corresponding graph is composed of the atoms marked in green on Fig. 5.36 (left). The corresponding central atom in the system R^{sys}, with its four first-neighbour atoms composing the graph are shown in green on Fig. 5.36 (middle), label this configuration S_{sys}. The graphs of S_{ini} and S_{sys} are isomorphic, thus the algorithm passes it onto the second part, which is the more rigorous geometry check (see Sec. 4.4.2).

In the second part of the algorithm, the correct transformation is found, such that the event structure S^E_{ini} is matched to the structure of the system R^{sys}, utilising the capability of IRA to match structural fragments. During the matching process, atoms of R^{sys} belonging to the environment extension of S^E_{ini} are searched. The atoms of extension of S^E_{ini} are the remaining two red atoms on Fig. 5.36 (left). The criterion for the search on those atoms is the distance from a reference position, thus the distance between an atom in S^E_{ini} which is the reference, and a corresponding atom in R^{sys}, see Fig. 5.36 (right). Thus, all atoms present in the system R^{sys} are searched in order to identify a subset which consists of atoms that are closest to the reference atoms, where the reference atoms are the atoms of the extension part of S^E_{ini} configuration of the event. The atoms of this subset are then combined with S_{sys}, to form the configuration which is matching the S^E_{ini} configuration. In the present example, the subset of atoms of R^{sys} matching the extension of S^E_{ini} are the two atoms circled in red on Fig. 5.36 (right), where the event environment S^E_{ini} is colored blue and written in the reference frame matching the system environment R^{sys}. The two found atoms are combined with S_{sys} into S^E_{sys}, which is such that it matches S^E_{ini}, with equal number of atoms. The atoms of S^E_{sys} are all within a satisfactory distance from their referece positions in S^E_{ini}, the distance $h(S^E_{ini}, S^E_{sys}) < \texttt{dist\_thr}$, and the event "adsorption 5" is deemed possible to occur.

During this whole process, the O atom in yellow that is present in one of the Si-Si bonds is never seen by the algorithm. It is impossible to include that O atom into any kind of graph, or structural check, because the present algorithm simply does not know about it. Thus an incorrcet event is marked as possible, and also gets applied.

The most obvious way of avoiding the showcased problem, is to increase the cutoff of the events. In this way, the primary part of the event configurations is larger, and since the primary part is sensitive to this kind of situations, the error would be avoided. However, increasing the event cutoff does not really solve this

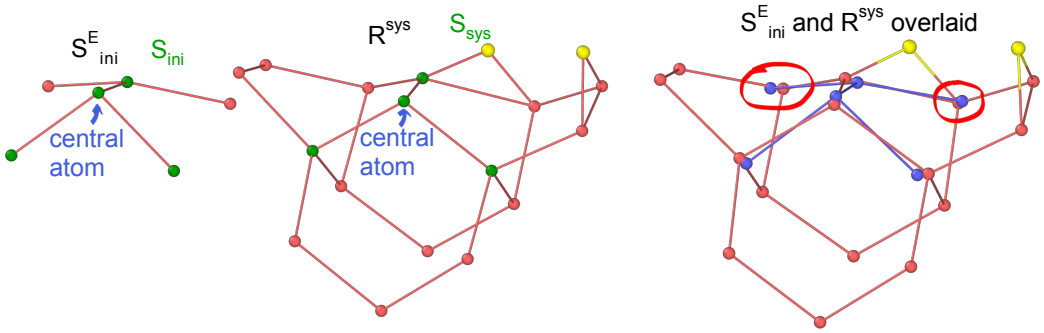


Figure 5.36: Left: initial state S^E_{ini} of the event, first neighbour atoms from the central atom are colored in green, they compose the primary part S_{ini} and the graph. Middle: local environment found in the system R^{sys}, first neighbour atoms from central atom are colored in green, they compose the primary part of the system configuration S_{sys} and their graph is isomorphic to the graph of event initial state S_{ini} of the left. Right: the two structures written in corresponding frame of reference, the two blue atoms of environment extension are circled in red, each of them has an atom from the system in red, within a satisfactory distance. The yellow O atom is never seen by the algorithm.

problem in general. A different cutoff means that the same situation can still happen at the border of whatever distance the event cutoff is, i.e. in the extension part of the environment. The problem is thus only pushed further away from the central atom. If the cutoff is large enough, then the problem might be negligible from the structural point of view (not speaking about the energetics). In this example, increasing the cutoff of events inevitably means that a much larger catalogue of events is needed, to take into account all the combinatiorial possibilities. On top of that, since there are long range elastic distortions, and events happening close to each other, or even interfering with one another, the number of events needed in the catalogue becomes enormous, and impossible to handle manually. Thus the need for the possibility of relaxation of the forces, and on-the-fly exploration of the events is crucial.

Chapter 6

Perspectives

This chapter presents some perspectives for the work done during this PhD project. Some of them are already a subject of on-going work, namely the efforts toward the on-the-fly exploration of events in the kMC (see Sec. 6.1). An observed side-effect of our kMC with an event catalogue, is that it can be exploited in other ways, not only for simulating the real evolution of a system (see Sec. 6.2). As an algorithm independent of its implementation in the kMC, IRA alone also has some interesting perspectives (see Sec. 6.3).

6.1 Towards the on-the-fly exploration of events in kMC

As has been shown throughout the Examples chapter, the main limitation of not computing the forces is that there are almost always some errors that either build up from smaller errors, or errors that arise on the border of an event, or errors due to interaction of multiple events. Or most often they are due to the event catalogue being incomplete. These errors are manifested through atomic positions, which can for some time continue to evolve due to the distortions being below the equivalence threshold, then at some point they surpass that threshold, and the evolution stops.

It is therefore fundamental in perspective to add relaxation of the forces with a realistic potential, and on-the-fly capabilities. Currently, the idea for implementing the on-the-fly learning capabilities heavily realies on the Activation-Relaxation Technique nouveau (ARTn) [133, 134, 135]. This section is meant as proof-of-concept. The importance of correct and exact physics is thus pushed a bit to the side.

6.1.1 Including relaxation of the forces

To include the relaxation of the forces, the off lattice kMC kernel with IRA shape matching implemented is interfaced with the LAMMPS molecular dynamics code [136], with the official website at `www.lammps.org`. This interface allows the usage of all capabilities of LAMMPS within our kMC, the main idea being the calculation of forces, to be used for structural relaxation.

The interfacing is done through the official fortran interface provided with the LAMMPS package. The calculation of forces requires the definition of a potential, which also come with LAMMPS.

In the future, coupling with a first-principles code such as Quantum Espresso [124] is not excluded.

The decision of when a relaxation of the system should be launched can be made by several criteria. The most simple one being to launch system relaxation every m number of steps. Other possibilities include launching relaxation once values of distances from the IRA shape matching algorithm start going beyond some threshold value, which usually indicates that there is an error buildup of whatever origin. Or, in situation where after event application some atoms in the system are distorted from their reference positions, given by the event final state. Or to replace the call to IDPP (Sec. 4.5) with a regular force minimization. The different criteria would most probably be chosen based on what kind of force computation we are doing, is it a fast empirical potential-based computation, or a slow but precise first-principles computation.

As of the moment of writing this, we have implemented the most simple force relaxation criterion, which is to launch it every m number of kMC steps.

6.1.2 Tracing the unknown configurations

An important aspect of the on-the-fly exploration scheme is to know when a configuration is unknown, as it has not been encountered before. The most straightforward tool available in our kMC for this task is the graph hash value.

Unknown structure by graph hash

This section is a kMC simulation of migration of vacancies in Al crystal. There is one event in the catalogue, which is a single vacancy migration, with environment cutoff of first neighbour. In the system of simulation there are two vacancies, which are initially three neighbours away. Thus initially, the migration event of one vacancy does not interfere with the migration event of the other vacancy. The situation is shown on Fig. 6.1, where the first-neighbour atoms around each vacancy are shown in white, and other atoms in transparent green.

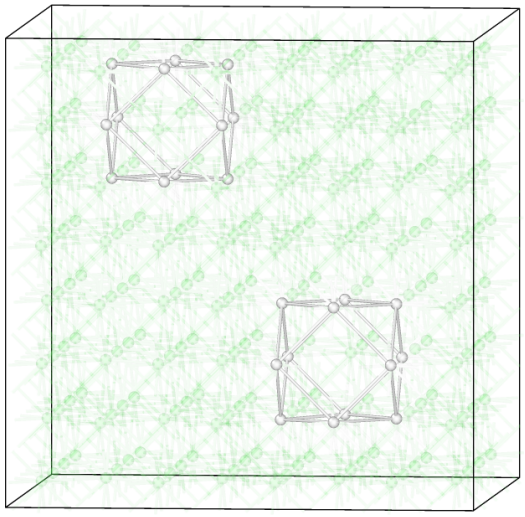


Figure 6.1: Initial state with two vacancies in Al. They are sufficiently far away such that the migration events do not interfere.

Within a number of migration events of each vacancy separately, we see some atomic environments which were not present at the initial state, nor in the events catalogue, recognised by a graph hash that has never appeared before. They represent environments where the two vacancies come close together, within the first neighbour distance. They are shown on Fig. 6.2, represented by black atoms.

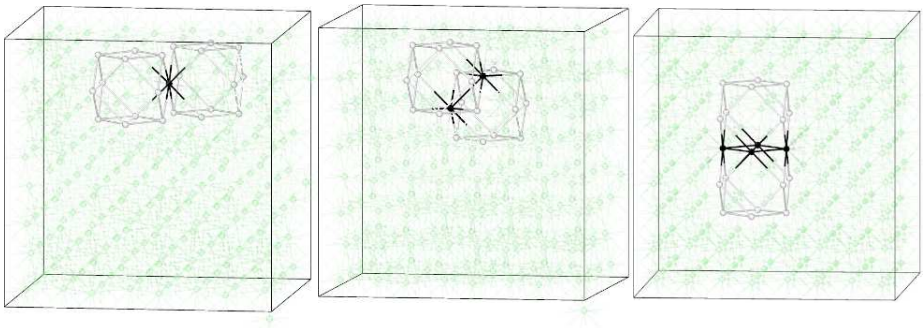


Figure 6.2: Unknown local environments

At the simulation step where a previously unknown graph hash is encountered, we have the data on atomic positions, and the index of the atom which had the new graph hash value. The configuration, along with the atomic index is sufficient information for ARTn exploration to succeed.

Thus the atomic positions and atomic index were written to a file whenever a new hash was found. At the end of one kMC run, the information from that

file was used to launch ARTn explorations on the system, using the embedded atom method (EAM) [137] potential provided in LAMMPS. The events found are shown on Fig. 6.3.

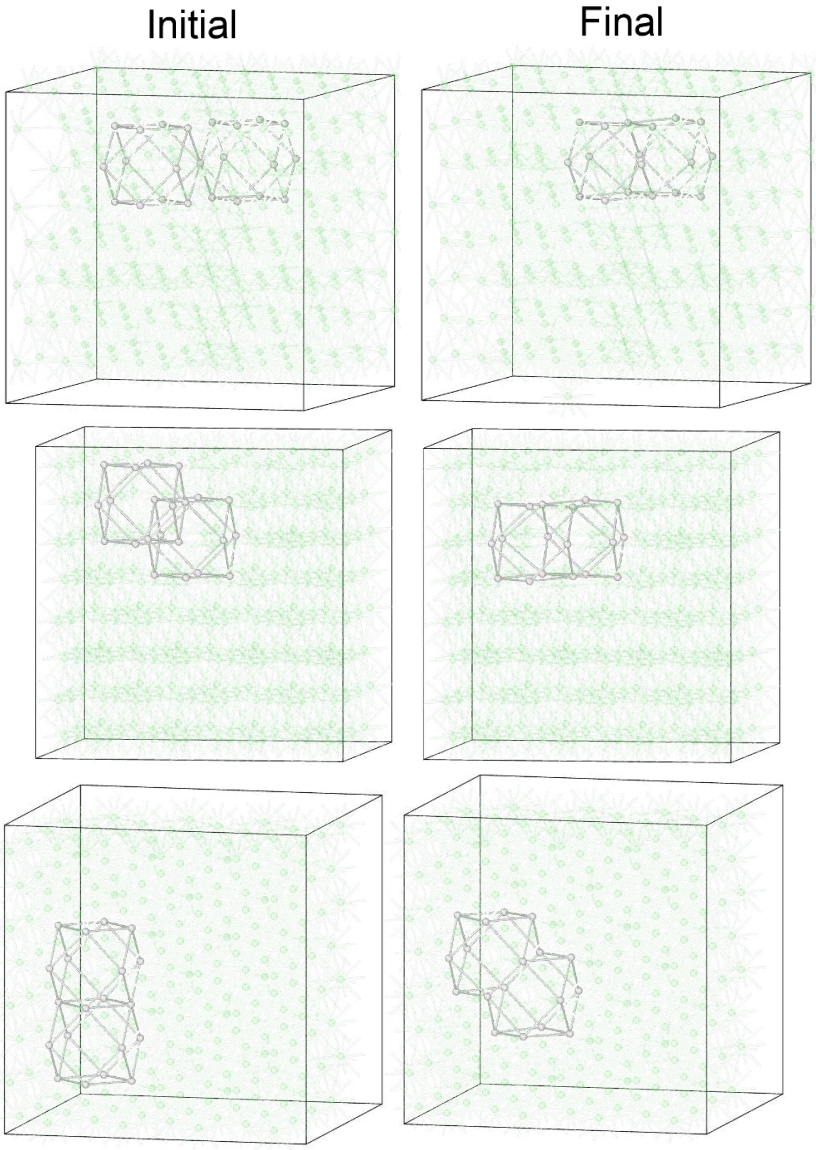


Figure 6.3: Found events by ARTn from the configurations of previously unknown graph hashes, from Fig. 6.2.

These events can then be included in the next kMC run, through the event catalogue. This process is not yet automated, but this example shows a possible workflow to follow, in which the information on unknown local environments is deduced from the graph hash values. A slightly more general approach could be

to recognize unknown structures with the IRA shape matching algorithm, which could provide information on specific atoms that differ the most from their reference positions. The ARTn exploration could then be launched on these specific atoms.

Potentially interesting event direction

If there is an event in the catalogue which has several possible directions, all the directions are tested during the identification of possible events at the current kMC step. If some direction is not found to be possible, due to the local environment in that particular direction having some distortions, that particular direction could be used as target direction for event exploration.

From my experience, typically the events with several possible directions are events on some kind of interstitial atoms. Thus if we have a diffusion of interstitial atom that is normally possible to occur in a number of independent directions, but then at some step is not possible in a specific direction - due to possibly another defect present in the environment in that direction - that particular direction might be an interesting area for event exploration. Since the knowledge of that particular direction is readily availabe in our kMC, it could directly be passed to the event exploration module, and used as guide for the exploration.

6.2 Exploiting the kMC and its event catalogue in different ways

The generation of event data described in Sec. 4.3 can also be used to transform some initial structure into any other structure. Such procedure could be useful for the preparation of structures used as initial state of a simulation.

For example, transforming an initially crystalline structure into a vacancy, or interstitial, or any combination of them. This kind of an event can then be used to create defects, along with the elastic distortions around them, as desired, in specific locations within the initially crystalline simulation box.

Taking for example the event which generates a vacancy, the event will be marked as desorption event in the events catalogue, since an atom disappears. The initial and final states of this event are shown on Fig. 6.4 for a single vacancy creation from a perfect Si crystal diamond structure.

Running a kMC simulation with this event for 20 steps produces 20 vacancies, as shown on Fig. 6.5, where on the bottom the green atoms are atoms around a vacancy.

The same trick can be done for interstitial atom generation. In that case the event would be treated as adsorption.

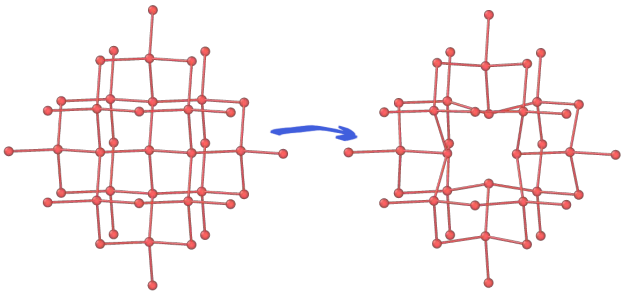


Figure 6.4: Vacancy generation event, transforming a crystalline structure into a vacancy defect structure, including the elastic distortions around it.

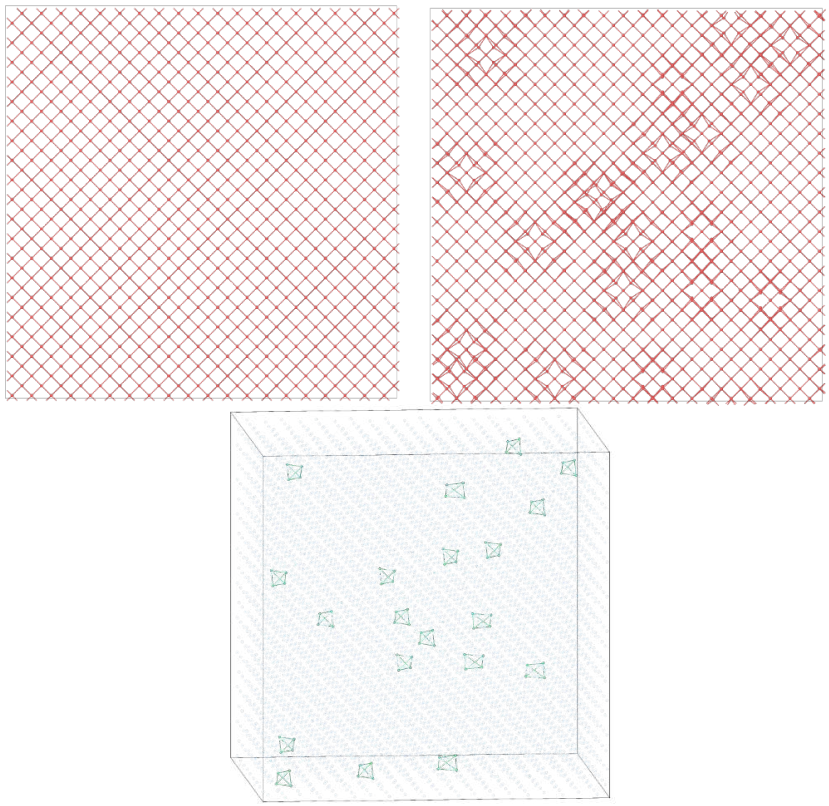


Figure 6.5: Top left: initial state of the simulation - Si in perfect crystalline structure; top right: structure after 20 vacancy-creation events, side view; bottom: structure after 20 vacancy-creation events, atoms in green are atoms around a vacancy.

With some minor modifications to the kMC procedure, such generation of defects could be done in a relatively fast and controlled way, where the number of

defects, or their density, could be controlled, and also their spatial distribution.

6.3 Beyond kMC

Other than just the ideas of heuristic techniques for further reducing the speed of IRA from the software point of view, some ideas for real applications of the IRA algorithm outside of the kMC setting are collected below.

6.3.1 IRA alone

Since IRA is independent of the shape of atomic structures, and does not specifically require any parameter of distance cutoff, it can be used to match and compare any kind of structures, and/or structural fragments.

For the example, in the MD trajectory of the cyanine molecule from Sec. 3.8.3, we can measure the diversity in the structure of a specific fragment. Take for example the fragment shown on Fig. 3.15 as reference structure A, and each instance of the cyanine molecule throughout the MD trajectory as structure B (the MD trajectory contains 80 thousand instances). Then make a histogram of all distances $RMSD(A, B)$ obtained after the matching procedure, on Fig. 6.6. Note the four peaks in the histogram, corresponding to the clustering of structures into four clusters. Fig. 6.6 also shows the typical member structure of each cluster on the right, where the reference fragment is shown in darker colors, and the viewing angle is such that the fragment is kept fixed. From the images we can immediately notice two main differences among the four: the rotation of the blue atom at the top left corner of the fragment, and the rotation of the rest of the molecule at the bottom of the fragment. From the point of view of a molecule-fixed reference frame, the latter actually corresponds to the rotation of the fragment around the bond connecting it to the rest of the molecule. This rotation is reported in the original work of cyanine, Ref. [121], as one of the two main structural changes of the cyanine molecule throughout the trajcetory.

More complicated examples could be devised, such as fragments containing parts of non-connected structures, or even just endpoints of a fragment. Like that, we could think of the matching process as a kind of "triangulation" of the structures. This could be used in collecting data about rotations of structures in a simulation, or even across multiple simulations. Like this one could also generate correlation functions over multiple structures, or fragments within the same structure, taking also into account their orientations. All this invariantly of the permutations of atoms. Another idea could also be to disregard the atomic species, and focus only on the overall shapes of structures.

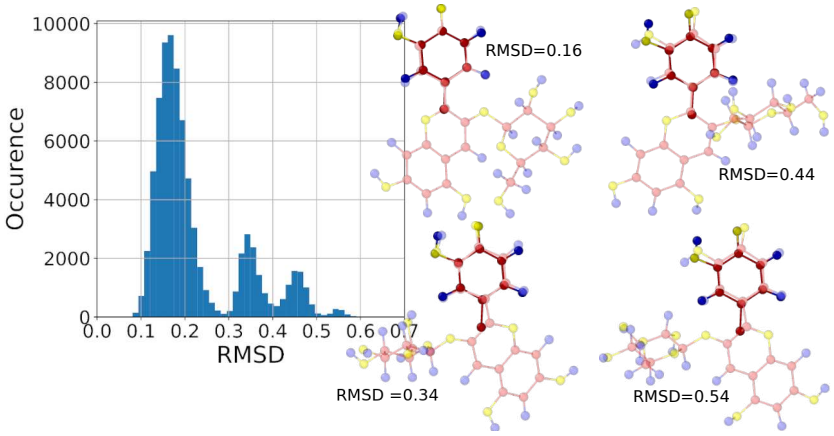


Figure 6.6: Histogram of the $RMSD(A, B)$ values obtained after the matching of structural fragment A, and the whole molecule B, for all steps of the MD trajectory. The four peaks correspond to grouping of the structures into four clusters, the representative structure of each cluster is shown on the right, with the reference fragment structure A in darker colors.

6.3.2 IRA with a catalogue of structures

The combination of a catalogue of structures, and the IRA algorithm could be used in a similar way as in the kMC, for other vectorial properties. For example as follows. Have not only the atomic structures in the catalogue, but also some other vectorial per-atom quantity, like the velocity, force, eigenvector, etc, which depend on the general reference frame of the structure. Then the matching transformation found by IRA, which consists of the rigid rotation, translation, and permutation, could be applied to this vectorial quantity, and thus be correctly mapped onto the atomic structure that is being compared. Moreover, the knowledge of all atom-to-reference-atom distances could be seen as a distortion score, and used to adjust, or interpolate, these vectorial quantities accordingly.

Summary and conclusions

As it has been highlighted throughout this thesis, the key point of an off-lattice kMC is the mapping between structures visited in the evolution, and the real representation of the PES. We have developed an algorithm Iterative Rotations and Assignments (IRA), which allows a precise description and a parameterless comparison of atomic structures. A workflow has also been developed, which inserts this algorithm into our in-house off-lattice kMC. This workflow enables efficient automatization and reuse of structural information throughout the kMC evolution.

The IRA algorithm implements a reliable, and accurate way of solving the shape matching problem, in congruent, and near-congruent cases. It operates directly with the raw state of structures, i.e. the atomic positions. There are two key components of our IRA shape matching algorithm, one is the way of solving for the rotations, and the other is solving for the permutations. To solve for the rotations, the main idea is the partitioning of the rotation space into specific points, given by the atoms of the structure itself (see Sec. 3.3.2). To solve for the permutations, the assignment algorithm Constrained Shortest Distance Assignment (CShDA) has been developed, which is based on the idea of minimizing the cost of assignment for each atom locally (see Sec. 3.4.1). CShDA resolves atomic assignments under the one-to-one assignment constraint, and provides the evaluation of the Hausdorff distance h (Eq. (2.10)), which is used as the distance D in the shape matching problem statement (Eq. (3.2)). CShDA works also for structures containing different number of atoms (see Sec. 3.4.4), which gives the possibility of generalizing the IRA algorithm to match generic structural fragments (see Sec. 3.5). The benchmark tests of IRA show a 100% efficiency in the case of exact congruence and equal number of atoms (see Sec. 3.8.1), and a slightly lower efficiency in the case of a fragments belonging to highly distorted structures (99.7% in Sec. 3.8.3). The loss of efficiency is related to larger distortions that appear among the structures, which open the possibility for the algorithm to find a numerically more satisfactory matching than the matching that is expected (correct). To mitigate these mismatches, prior knowledge of the structures needs to be provided to the algorithm. In Sec. 3.8.4, we suggest this knowledge in the form of a known common point, or a central atom. This information limits the search

space of IRA algorithm to a more relevant subspace where the correct matching is expected to be found.

In order to exploit IRA in our off-lattice kinetic Monte Carlo (kMC) kernel, a workflow has been developed, which enables efficient automatization and reuse of structural information. In particular, the workflow separates the catalogue of events from the internal logic of the kMC (see Sec. 4.1). This separation allows greater user control over the events included in the kMC, and keeps the internal logic of the kMC independent of the events, thus making it general. This is achieved by splitting the IRA algorithm into two parts. The first part is done already in the catalogue of events, where the basis β of Eq. (3.4) is found, along with possible symmetries θ of the event (see Sec. 4.3.5). In order to accelerate the identification of possible events in the system of simulation, a pre-screening based on isomorphism of simple graphs is used. This is done through the NAUTY [73] software, which generates a canonical hash value for each graph, such that identical hash values indicate isomorphic graphs. Each event in the catalogue contains information on this hash value (see Sec. 4.3.4). Each atomic site in the kMC simulation which passes the pre-screening of graph isomorphism (see Sec. 4.4.1), is passed to the second part of the IRA algorithm. At this stage, IRA searches for the γ basis in the configuration of the kMC system (see Eq. (3.4), Sec. 4.4.2). The information from the two pieces of the IRA algorithm are combined into the solution of the shape matching problem, such that the similarity of structures can be evaluated, and the existence of possible symmetries confirmed (see Sec. 4.4.3). Each confirmed symmetry of an event is deemed as separate event. An event is deemed possible if the similarity (see Eq. (4.14)) is below a designated threshold. After the selection of a certain event as the next event to be applied, the information from IRA is retrieved, and the event is applied (see Sec. 4.4.4).

In order to make the workflow with the event catalogue robust and portable to many systems, while operating at a high degree of automatization, we had to face several problems. In order to enable the use of graph isomorphism as a pre-screening technique, a cutoff for the local environment had to be introduced. Since the events can be of different sizes among each other, we have introduced a method which allows different events to have their own specific cutoff. This method defines a cutoff value that is common to all events, and first screens the graph hash values within this cutoff. The subset of atomic sites whose graphs are isomorphic with the graph of some event in the common cutoff, are further checked with the specific cutoff of the event. In the case of diffusing atoms, we noticed that in order to have real control over the diffusion process, the final state of a diffusion event has to be included in the local configuration of its initial state (see Sec. 5.1). For this reason we have introduced the concept of an extended local configuration (see Sec. 4.2.4), which by default includes atoms comprising the initial and final states of an event. Other criteria can be used for the inclusion of

additional atoms in the extended environment, as for instance including atoms that move a significant distance during an event. In the case when there is a disparity between the extent of elastic distortions present in the system of simulation, and the size of configurations included in an event, we noticed unphysical and non-negligible accumulation of distortions in the atomic positions (see Sec. 5.2 and Sec. 5.5.2). We mitigate the situation by using an ad-hoc variation of the Image-Dependent Pair Potential (IDPP) interpolation scheme (see Sec. 4.5), which brings the atoms after an event application closer to their reference positions, which are given by the final state configuration of an event. This mitigation was successful in the case when there is only one event possible to happen in the same region (see Sec. 5.2.1), while it had only limited effect when a higher number of events are happening in the same region (see Sec. 5.5.2). Combining all these considerations, successful simulation of a non-trivial diffusion process has been shown, namely the Si interstitial diffusion in silicon, where there is a possibility of exchange of the diffusing atom (see Sec. 5.3), as well as simulation of processes which change the total number of atoms, such as adsorption and desorption of CO molecule on the SnO_2 surface (see Sec. 5.4), and the oxidation of Si(100) surface (Sec. 5.5). The off-lattice nature of our approach has been proven useful for the recognition of geometrically different atomic sites in the buckled Si(100) surface (see Sec. 5.5.1), which is not possible with lattice-based kMC approaches. Moreover, the oxidation of this surface (see Sec. 5.5.2 and Sec. 5.5.3) is the paradigm of a very complex system, since the oxide grows in the amorphous phase, and the elastic distortions are extremely long-ranged. As such, the local environment encompassing each event needs to be quite large to meaningfully map the diversity of the local structures. This inevitably implies that the catalogue needs to include a huge number of events. Moreover, along the evolution the adsorption of additional oxygen atoms occurs in parallel with diffusion-like events, thus becoming an even larger combinatory problem. For this reason, the generation of input event data by hand becomes unaffordable, and the need for structural relaxations and an on-the-fly approach to the exploration of events is crucial.

Beyond the use of IRA within our off-lattice kMC, the developed workflow could be useful for finding correlations and identifying collective behaviours, in the context of any simulation method working with atomic positions (Sec. 6.3). It could also aid in constructing correlation functions of generic atomic structures, or any other vectorial quantity (velocities, forces, etc.), disentangled of the simulated time. The structural similarity expressed in the distance D can be interpreted as distortion score, and used as surrogate model for scalar properties (definition of a "local energy").

The natural evolution of the present work is to extend our in-house general off-lattice kMC with the capabilities of structural relaxation, and an on-the-fly approach to the exploration of events, and this is already in progress (see Sec. 6.1).

Appendix A

A breadth-first graph traversal algorithm

A breadth-first graph traversal algorithm that is used within the scope of our kMC software is described.

Any known node (or vertex) of a graph can be called a "parent" node, then all nodes connected to the parent node can be called "child" nodes. In the context of atomic structures inside our kMC software, the nodes are mapped from atomic positions. As described in Sec. 2.2.1, two nodes are connected when the distance between the atoms representing the two nodes are within a radial distance given by the cutoff.

The phrase "graph traversal" refers to finding some path over the nodes and connections of a graph, starting at the input parent node, and traversing the graph by visiting the child nodes. Such path might have a given definite endpoint, for example a specific node index, or it might have a given size, or depth. There are in general two ways a graph can be traversed, the first is called depth-first, and the second is called breadth-first. The former is not discussed here, but the basic description of the latter is the following. In a single step of the traversal path, all child nodes of given parent node(s) are visited, and memorized. In the next step, these nodes become parent nodes, and all their child nodes are visited. This basic operation is illustrated on Fig. A.1, where the input node is colored in green, all child nodes of the input node are colored in blue, and will be visited in the first step of a breadth-first traversal algorithm. On the second step, the blue nodes become parent nodes, and all red nodes will be visited.

We write the connectivity of an atomic structure in a $(N \times N)$ matrix C, where N is the number of atoms. The matrix elements c_{ij} are given as Eq. (2.2).

$$c_{ij} = \begin{cases} 1, & \text{if } d(i,j) \leq R_{cut} \\ 0, & \text{otherwise} \end{cases} \qquad \text{(2.2 revisited)}$$

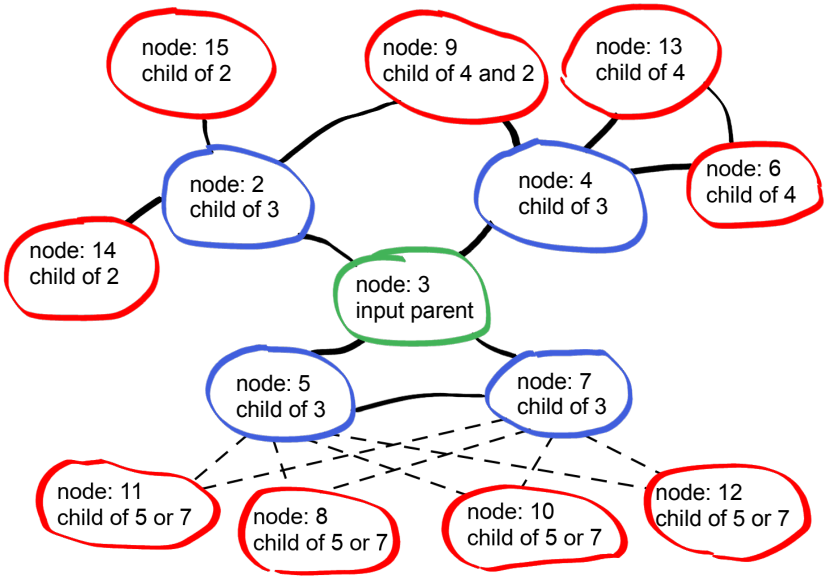


Figure A.1: Breadth-first traversal of a given graph. The input node is colored in green, it is the start of a traversal path. In the first step, all blue nodes are visited, in the second step all red nodes are visited. When two nodes at the same level are connected to each other, such as nodes 5 and 7, keeping the correct history of the traversal path can be tricky, but is irrelevant for our purpose.

Assume an N-dimensional $(0, 1)$ integer vector v, call it the input vector, with values 1 on specific indexes m. Multiplying the connectivity matrix C with v gives a resulting vector u, as in Eq. A.1.

$$Cv = u \tag{A.1}$$

The vector u contains nonzero elements on indexes which are endpoints of connections from all indexes m of the vector v. The values at nonzero indexes of u are the number of paths leading to that index, from all indexes m. Stated simply, vector u is nonzero at indexes of all child nodes of the parent nodes with indexes m.

We can reuse the ouput vector u as input vector v for the second step: $v \leftarrow u$ and apply Eq. A.1. The new vector u has nonzero values at indexes of all second-level child nodes of the parent node(s).

If the input node m in the first step is the green node from Fig. A.1, then the vector u at first step contains all blue nodes, and at second step all blue and red nodes. Repeating the processs once more would return the third-level child nodes, etc. In the context of atomic structures, each n-th level of child nodes represents the n-th neighbours from the input parent atom, with respect to the connectivity matrix C.

In our kMC software, we use this algorithm to search for atoms belonging to a local environment of a specific atom (see Sec. 4.3.3). In this context, the connectivity matrix contains all atoms in the simulation box. The $(N \times N)$ representation of the connectivity matrix can become extremely large, so we transform it to a neighbour list structure. The breadth-first tarversal algorithm described above is then transformed accordingly, but the idea remains the same.

A problem with the described algorithm is that the exact history of each step of the traversal path is lost when the input vector v contains several (interconnected) nodes, such as nodes 5 and 7 on Fig. A.1. This is however irrelevant for our needs.

Bibliography

[1] A. Laio and M. Parrinello, "Escaping free-energy minima," *Proceedings of the National Academy of Sciences*, vol. 99, no. 20, pp. 12 562–12 566, 2002. URL: https://www.pnas.org/content/99/20/12562

[2] A. Laio and F. L. Gervasio, "Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science," *Reports on Progress in Physics*, vol. 71, no. 12, p. 126601, nov 2008. URL: https://doi.org/10.1088/0034-4885/71/12/126601

[3] G. Henkelman and H. Jónsson, "Long time scale kinetic Monte Carlo simulations without lattice approximation and predefined event table," *The Journal of Chemical Physics*, vol. 115, no. 21, pp. 9657–9666, 2001. URL: https://doi.org/10.1063/1.1415500

[4] L. K. Béland, P. Brommer, F. El-Mellouhi, J.-F. m. c. Joly, and N. Mousseau, "Kinetic activation-relaxation technique," *Phys. Rev. E*, vol. 84, p. 046704, Oct 2011. URL: https://link.aps.org/doi/10.1103/PhysRevE.84.046704

[5] I. Martin-Bragado, R. Borges, J. P. Balbuena, and M. Jaraiz, "Kinetic Monte Carlo simulation for semiconductor processing: A review," *Progress in Materials Science*, vol. 92, pp. 1–32, 2018. URL: https://www.sciencedirect.com/science/article/pii/S0079642517301135

[6] B. P. Haley, K. M. Beardmore, and N. Grønbech-Jensen, "Vacancy clustering and diffusion in silicon: Kinetic lattice Monte Carlo simulations," *Phys. Rev. B*, vol. 74, p. 045217, Jul 2006. URL: https://link.aps.org/doi/10.1103/PhysRevB.74.045217

[7] I. Martin-Bragado and V. Moroz, "Facet formation during solid phase epitaxy regrowth: A lattice kinetic Monte Carlo model," *Applied Physics Letters*, vol. 95, no. 12, p. 123123, 2009. URL: https://doi.org/10.1063/1.3236535

[8] A. Hémeryck, A. Estève, N. Richard, M. D. Rouhani, and G. Landa, "A kinetic Monte Carlo study of the initial stage of silicon oxidation: Basic mechanisms-induced partial ordering of the oxide interfacial layer," *Surface Science*, vol. 603, no. 13, pp. 2132–2137, 2009. URL: https://www.sciencedirect.com/science/article/pii/S0039602809002957

[9] M. Stamatakis and D. G. Vlachos, "A graph-theoretical kinetic Monte Carlo framework for on-lattice chemical kinetics," *J. Chem. Phys.*, vol. 134, no. 21, p. 214115, Jun 2011. URL: https://doi.org/10.1063/1.3596751

[10] M. J. Hoffmann, S. Matera, and K. Reuter, "kmos: A lattice kinetic Monte Carlo framework," *Computer Physics Communications*, vol. 185, no. 7, pp. 2138–2150, 2014. URL: https://www.sciencedirect.com/science/article/pii/S001046551400126X

[11] S. Piccinin and M. Stamatakis, "Steady-State CO Oxidation on Pd(111): First-Principles Kinetic Monte Carlo Simulations and Microkinetic Analysis," *Top. Catal.*, vol. 60, no. 1, pp. 141–151, Feb 2017. URL: https://doi.org/10.1007/s11244-016-0725-5

[12] S. Plimpton, C. Battaile, M. Chandross, A. Holm, L.and Thompson, V. Tikare, G. Wagner, E. Webb, X. Zhou, C. Garcia Cardona, and A. Slepoy, "Crossing the Mesoscale No-Man's Land via Parallel Kinetic Monte Carlo," *Sandia report SAND2009-6226*, October 2009. URL: https://spparks.github.io/index.html

[13] J. Li, G. Liu, B. Ren, E. Croiset, Y. Zhang, and L. Ricardez-Sandoval, "Mechanistic study of site blocking catalytic deactivation through accelerated kinetic Monte Carlo," *Journal of Catalysis*, vol. 378, pp. 176–183, 2019. URL: https://www.sciencedirect.com/science/article/pii/S0021951719304154

[14] Z. Wu, K. Tan, R. Zhang, Q. Wei, and Y. Lin, "Atomistic kinetic Monte Carlo Embedded atom method simulation on growth and morphology of Cu–Zn–Sn precursor of Cu2ZnSnS4 solar cells," *J. Mater. Res.*, vol. 35, no. 3, pp. 252–262, Feb 2020. URL: https://doi.org/10.1557/jmr.2019.413

[15] O. Trushin, A. Karim, A. Kara, and T. S. Rahman, "Self-learning kinetic Monte Carlo method: Application to Cu(111)," *Phys. Rev. B*, vol. 72, p. 115401, Sep 2005. URL: https://link.aps.org/doi/10.1103/PhysRevB.72.115401

[16] H. Xu, Y. N. Osetsky, and R. E. Stoller, "Simulating complex atomistic processes: On-the-fly kinetic Monte Carlo scheme with selective

active volumes," *Phys. Rev. B*, vol. 84, p. 132103, Oct 2011. URL: https://link.aps.org/doi/10.1103/PhysRevB.84.132103

[17] D. Konwar, V. J. Bhute, and A. Chatterjee, "An off-lattice, self-learning kinetic Monte Carlo method using local environments," *The Journal of Chemical Physics*, vol. 135, no. 17, p. 174103, 2011. URL: https://doi.org/10.1063/1.3657834

[18] G. H. Vineyard, "Frequency factors and isotope effects in solid state rate processes," *Journal of Physics and Chemistry of Solids*, vol. 3, no. 1, pp. 121–127, 1957. URL: https://www.sciencedirect.com/science/article/pii/0022369757900598

[19] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *Journal of Computational Physics*, vol. 22, no. 4, pp. 403–434, 1976. URL: https://www.sciencedirect.com/science/article/pii/0021999176900413

[20] A. B. Bortz, M. H. Kalos, J. L. Lebowitz, and M. A. Zendejas, "Time evolution of a quenched binary alloy: Computer simulation of a two-dimensional model system," *Phys. Rev. B*, vol. 10, pp. 535–541, Jul 1974. URL: https://link.aps.org/doi/10.1103/PhysRevB.10.535

[21] A. Slepoy, A. P. Thompson, and S. J. Plimpton, "A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks," *The Journal of Chemical Physics*, vol. 128, no. 20, p. 205101, 2008. URL: https://doi.org/10.1063/1.2919546

[22] A. F. Voter, "Introduction to the kinetic Monte Carlo method," in *Radiation Effects in Solids*, K. E. Sickafus, E. A. Kotomin, and B. P. Uberuaga, Eds. Dordrecht: Springer Netherlands, 2007, pp. 1–23. URL: https://doi.org/10.1007/978-1-4020-5295-8_1

[23] K. A. Fichthorn and W. H. Weinberg, "Theoretical foundations of dynamical Monte Carlo simulations," *The Journal of Chemical Physics*, vol. 95, no. 2, pp. 1090–1096, 1991. URL: https://doi.org/10.1063/1.461138

[24] M. A. Novotny, *A Tutorial on Advanced Dynamic Monte Carlo Methods for Systems with Discrete State Spaces*. Worl Scientific, 2001, pp. 153–210. URL: https://www.worldscientific.com/doi/abs/10.1142/9789812811578_0003

[25] B. Puchala, M. L. Falk, and K. Garikipati, "An energy basin finding algorithm for kinetic Monte Carlo acceleration," *The Journal of Chemical Physics*, vol. 132, no. 13, p. 134104, 2010. URL: https://doi.org/10.1063/1.3369627

[26] K. A. Fichthorn and Y. Lin, "A local superbasin kinetic Monte Carlo method," *The Journal of Chemical Physics*, vol. 138, no. 16, p. 164104, 2013. URL: https://doi.org/10.1063/1.4801869

[27] K. Ferasat, Y. N. Osetsky, A. V. Barashev, Y. Zhang, Z. Yao, and L. K. Béland, "Accelerated kinetic Monte Carlo: A case study; vacancy and dumbbell interstitial diffusion traps in concentrated solid solution alloys," *The Journal of Chemical Physics*, vol. 153, no. 7, p. 074109, 2020. URL: https://doi.org/10.1063/5.0015039

[28] W. Kaiser, M. Gößwein, and A. Gagliardi, "Acceleration scheme for particle transport in kinetic Monte Carlo methods," *The Journal of Chemical Physics*, vol. 152, no. 17, p. 174106, 2020. URL: https://doi.org/10.1063/5.0002289

[29] F. Soisson, C. Becquart, N. Castin, C. Domain, L. Malerba, and E. Vincent, "Atomistic Kinetic Monte Carlo studies of microchemical evolutions driven by diffusion processes under irradiation," *Journal of Nuclear Materials*, vol. 406, no. 1, pp. 55–67, 2010, fP6 IP PERFECT Project: Prediction of Irradiation Damage Effects in Reactor Components. URL: https://www.sciencedirect.com/science/article/pii/S0022311510002308

[30] N. Castin, L. Messina, C. Domain, R. C. Pasianot, and P. Olsson, "Improved atomistic Monte Carlo models based on ab-initio-trained neural networks: Application to FeCu and FeCr alloys," *Phys. Rev. B*, vol. 95, p. 214117, Jun 2017. URL: https://link.aps.org/doi/10.1103/PhysRevB.95.214117

[31] T. Garnier and M. Nastar, "Coarse-grained kinetic Monte Carlo simulation of diffusion in alloys," *Phys. Rev. B*, vol. 88, p. 134207, Oct 2013. URL: https://link.aps.org/doi/10.1103/PhysRevB.88.134207

[32] N. Castin, M. I. Pascuet, and L. Malerba, "Modeling the first stages of Cu precipitation in a-Fe using a hybrid atomistic kinetic Monte Carlo approach," *The Journal of Chemical Physics*, vol. 135, no. 6, p. 064502, 2011. URL: https://doi.org/10.1063/1.3622045

[33] D. R. Mason, A. E. Sand, and S. L. Dudarev, "Atomistic-object kinetic Monte Carlo simulations of irradiation damage in tungsten," *Modelling*

and Simulation in Materials Science and Engineering, vol. 27, no. 5, p. 055003, may 2019. URL: https://doi.org/10.1088/1361-651x/ab1a1e

[34] A. Ali-Messaoud, A. Chikouche, A. Estève, A. Hemeryck, C. Lanthony, C. Mastail, and M. Djafari Rouhani, "Defect generation during silicon oxidation: A Kinetic Monte Carlo study," *Thin Solid Films*, vol. 520, no. 14, pp. 4734–4740, 2012, proceedings of the EMRS 2011 Spring Meeting Symposium D: Synthesis, Processing and Characterization of Nanoscale Multi Functional Oxide Films III. URL: https://www.sciencedirect.com/science/article/pii/S0040609011019481

[35] S. Alas and L. Vicente, "TPD study of NO decomposition on Rh(111) by dynamic Monte Carlo simulation," *Surface Science*, vol. 604, no. 11, pp. 957–964, 2010. URL: https://www.sciencedirect.com/science/article/pii/S0039602810000828

[36] M. Andersen, C. Panosetti, and K. Reuter, "A Practical Guide to Surface Kinetic Monte Carlo Simulations," *Frontiers in Chemistry*, vol. 7, p. 202, 2019. URL: https://www.frontiersin.org/article/10.3389/fchem.2019.00202

[37] H. Prats, L. Álvarez, F. Illas, and R. Sayós, "Kinetic Monte Carlo simulations of the water gas shift reaction on Cu(111) from density functional theory based calculations," *Journal of Catalysis*, vol. 333, pp. 217–226, 2016. URL: https://www.sciencedirect.com/science/article/pii/S0021951715003619

[38] H. Prats, F. Illas, and R. Sayós, "General concepts, assumptions, drawbacks, and misuses in kinetic Monte Carlo and microkinetic modeling simulations applied to computational heterogeneous catalysis," *International Journal of Quantum Chemistry*, vol. 118, no. 9, p. e25518, 2018. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.25518

[39] J. P. Balbuena and I. Martin-Bragado, "Lattice kinetic Monte Carlo simulation of epitaxial growth of silicon thin films in H2/SiH4 chemical vapor deposition systems," *Thin Solid Films*, vol. 634, pp. 121–133, 2017. URL: https://www.sciencedirect.com/science/article/pii/S0040609017303462

[40] X. Chen and Y. Li, "Stepped morphology on vicinal 3C- and 4H-SiC (0001) faces: A kinetic Monte Carlo study," *Surface Science*, vol. 681, pp. 18–23, 2019. URL: https://www.sciencedirect.com/science/article/pii/S0039602818304618

[41] S. Ravipati, M. d'Avezac, J. Nielsen, J. Hetherington, and M. Stamatakis, "A Caching Scheme To Accelerate Kinetic Monte Carlo Simulations of Catalytic Reactions," *The Journal of Physical Chemistry A*, vol. 124, no. 35, pp. 7140–7154, 2020, pMID: 32786994. URL: https://doi.org/10.1021/acs.jpca.0c03571

[42] J. Nielsen, M. d'Avezac, J. Hetherington, and M. Stamatakis, "Parallel kinetic Monte Carlo simulation framework incorporating accurate models of adsorbate lateral interactions," *The Journal of Chemical Physics*, vol. 139, no. 22, p. 224706, 2013. URL: https://doi.org/10.1063/1.4840395

[43] M. Panshenskov, I. A. Solov'yov, and A. V. Solov'yov, "Efficient 3D kinetic Monte Carlo method for modeling of molecular structure and dynamics," *J. Comput. Chem.*, vol. 35, no. 17, pp. 1317–1329, Jun 2014. URL: https://doi.org/10.1002/jcc.23613

[44] J. Dai, J. M. Kanter, S. S. Kapur, W. D. Seider, and T. Sinno, "On-lattice kinetic Monte Carlo simulations of point defect aggregation in entropically influenced crystalline systems," *Phys. Rev. B*, vol. 72, p. 134102, Oct 2005. URL: https://link.aps.org/doi/10.1103/PhysRevB.72.134102

[45] C. A. Miermans and C. P. Broedersz, "A lattice kinetic Monte-Carlo method for simulating chromosomal dynamics and other (non-)equilibrium bio-assemblies," *Soft Matter*, vol. 16, no. 2, pp. 544–556, Jan 2020. URL: https://doi.org/10.1039/C9SM01835B

[46] Y. K. Lee and T. Sinno, "Analysis of the lattice kinetic Monte Carlo method in systems with external fields," *The Journal of Chemical Physics*, vol. 145, no. 23, p. 234104, 2016. URL: https://doi.org/10.1063/1.4972052

[47] C. Domain, C. Becquart, and L. Malerba, "Simulation of radiation damage in Fe alloys: an object kinetic Monte Carlo approach," *Journal of Nuclear Materials*, vol. 335, no. 1, pp. 121–145, 2004. URL: https://www.sciencedirect.com/science/article/pii/S0022311504006385

[48] C. Meng, L. Wang, K. Xu, J. Hao, H. bo Zhou, X. Shu, S. Jin, L. Liang, G.-H. Lu, and C. Becquart, "Object Kinetic Monte Carlo simulation of hydrogen clustering behaviour with vacancies in tungsten," *Journal of Nuclear Materials*, vol. 526, p. 151768, 2019. URL: https://www.sciencedirect.com/science/article/pii/S0022311519303757

[49] C. Domain, C. S. Becquart, Y. Chen, E. Homer, and C. A. Schuh, "Object Kinetic Monte Carlo (OKMC): A Coarse-Grained

Approach to Radiation Damage," in *Handbook of Materials Modeling*. Cham, Switzerland: Springer, Mar 2020, pp. 1287–1312. URL: https://doi.org/10.1007/978-3-319-44677-6_101

[50] L. Xu and G. Henkelman, "Adaptive kinetic Monte Carlo for first-principles accelerated dynamics," *The Journal of Chemical Physics*, vol. 129, no. 11, p. 114104, 2008. URL: https://doi.org/10.1063/1.2976010

[51] G. Henkelman and H. Jónsson, "A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives," *The Journal of Chemical Physics*, vol. 111, no. 15, pp. 7010–7022, 1999. URL: https://doi.org/10.1063/1.480097

[52] S. T. Chill and G. Henkelman, "Molecular dynamics saddle search adaptive kinetic Monte Carlo," *The Journal of Chemical Physics*, vol. 140, no. 21, p. 214110, 2014. URL: https://doi.org/10.1063/1.4880721

[53] S. T. Chill, M. Welborn, R. Terrell, L. Zhang, J.-C. Berthet, A. Pedersen, H. Jónsson, and G. Henkelman, "EON: software for long time simulations of atomic scale systems," *Modelling and Simulation in Materials Science and Engineering*, vol. 22, no. 5, p. 055002, may 2014. URL: https://doi.org/10.1088/0965-0393/22/5/055002

[54] A. Kara, O. Trushin, H. Yildirim, and T. S. Rahman, "Off-lattice self-learning kinetic Monte Carlo: application to 2D cluster diffusion on the fcc(111) surface," *Journal of Physics: Condensed Matter*, vol. 21, no. 8, p. 084213, jan 2009. URL: https://doi.org/10.1088/0953-8984/21/8/084213

[55] H. Xu, Y. N. Osetsky, and R. E. Stoller, "Self-evolving atomistic kinetic Monte Carlo: fundamentals and applications," *Journal of Physics: Condensed Matter*, vol. 24, no. 37, p. 375402, aug 2012. URL: https://doi.org/10.1088/0953-8984/24/37/375402

[56] S. Hayakawa, J. Isaacs, H. R. Medal, and H. Xu, "Atomistic modeling of meso-timescale processes with SEAKMC: A perspective and recent developments," *Computational Materials Science*, vol. 194, p. 110390, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0927025621001154

[57] R. Terrell, M. Welborn, S. T. Chill, and G. Henkelman, "Database of atomistic reaction mechanisms with application to kinetic Monte Carlo," *The Journal of Chemical Physics*, vol. 137, no. 1, p. 014105, 2012. URL: https://doi.org/10.1063/1.4730746

[58] R. C. Veltkamp, "Shape matching: similarity measures and algorithms," in *Proceedings International Conference on Shape Modeling and Applications*, May 2001, pp. 188–197. URL: https://doi.org/10.1109/SMA.2001.923389

[59] D. Eppstein, M. T. Goodrich, J. Jorgensen, and M. R. Torres, "Geometric Fingerprint Recognition via Oriented Point-Set Pattern Matching," *CoRR*, vol. abs/1808.00561, 2018. URL: https://arxiv.org/abs/1808.00561

[60] J. Sreevalsan-Nair, A. Jindal, and B. Kumari, "Contour Extraction in Buildings in Airborne LiDAR Point Clouds Using Multiscale Local Geometric Descriptors and Visual Analytics," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 7, pp. 2320–2335, July 2018. URL: https://ieeexplore.ieee.org/document/8369345

[61] J. Vymětal and J. Vondrášek, "Gyration- and Inertia-Tensor-Based Collective Coordinates for Metadynamics. Application on the Conformational Behavior of Polyalanine Peptides and Trp-Cage Folding," *The Journal of Physical Chemistry A*, vol. 115, no. 41, pp. 11 455–11 465, 2011, pMID: 21961799. URL: https://doi.org/10.1021/jp2065612

[62] H. Arkın and W. Janke, "Gyration tensor based analysis of the shapes of polymer chains in an attractive spherical cage," *The Journal of Chemical Physics*, vol. 138, no. 5, p. 054904, 2013. URL: https://doi.org/10.1063/1.4788616

[63] J. Behler and M. Parrinello, "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces," *Phys. Rev. Lett.*, vol. 98, p. 146401, Apr 2007. URL: https://link.aps.org/doi/10.1103/PhysRevLett.98.146401

[64] A. P. Bartók, R. Kondor, and G. Csányi, "On representing chemical environments," *Phys. Rev. B*, vol. 87, p. 184115, May 2013. URL: https://link.aps.org/doi/10.1103/PhysRevB.87.184115

[65] R. Drautz, "Atomic cluster expansion for accurate and transferable interatomic potentials," *Phys. Rev. B*, vol. 99, p. 014104, Jan 2019. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.014104

[66] M. Valle and A. R. Oganov, "Crystal fingerprint space – a novel paradigm for studying crystal-structure sets," *Acta Crystallographica Section A*, vol. 66, no. 5, pp. 507–517, Sep 2010. URL: https://doi.org/10.1107/S0108767310026395

[67] J. Jenke, A. P. A. Subramanyam, M. Densow, T. Hammerschmidt, D. G. Pettifor, and R. Drautz, "Electronic structure based descriptor for characterizing local atomic environments," *Phys. Rev. B*, vol. 98, p. 144102, Oct 2018. URL: https://link.aps.org/doi/10.1103/PhysRevB.98.144102

[68] G. A. Gallet and F. Pietrucci, "Structural cluster analysis of chemical reactions in solution," *The Journal of Chemical Physics*, vol. 139, no. 7, p. 074101, 2013. URL: https://doi.org/10.1063/1.4818005

[69] E. Kocer, J. K. Mason, and H. Erturk, "A novel approach to describe chemical environments in high-dimensional neural network potentials," *The Journal of Chemical Physics*, vol. 150, no. 15, p. 154102, 2019. URL: https://doi.org/10.1063/1.5086167

[70] C. Zeni, K. Rossi, A. Glielmo, and S. de Gironcoli, "Compact atomic descriptors enable accurate predictions via linear models," *The Journal of Chemical Physics*, vol. 154, no. 22, p. 224112, 2021. URL: https://doi.org/10.1063/5.0052961

[71] A. Glielmo, P. Sollich, and A. De Vita, "Accurate interatomic force fields via machine learning with covariant kernels," *Phys. Rev. B*, vol. 95, p. 214302, Jun 2017. URL: https://link.aps.org/doi/10.1103/PhysRevB.95.214302

[72] A. Samanta, "Representing local atomic environment using descriptors based on local correlations," *The Journal of Chemical Physics*, vol. 149, no. 24, p. 244102, 2018. URL: https://doi.org/10.1063/1.5055772

[73] B. D. McKay and A. Piperno, "Practical graph isomorphism, {II}," *Journal of Symbolic Computation*, vol. 60, no. 0, pp. 94 – 112, 2014. URL: https://www.sciencedirect.com/science/article/pii/S0747717113001193

[74] O. Isayev, C. Oses, C. Toher, E. Gossett, S. Curtarolo, and A. Tropsha, "Universal fragment descriptors for predicting properties of inorganic crystals," *Nature Communications*, vol. 8, no. 1, p. 15679, Jun 2017. URL: https://doi.org/10.1038/ncomms15679

[75] F. Pietrucci and W. Andreoni, "Graph Theory Meets Ab Initio Molecular Dynamics: Atomic Structures and Transformations at the Nanoscale," *Phys. Rev. Lett.*, vol. 107, p. 085504, Aug 2011. URL: https://link.aps.org/doi/10.1103/PhysRevLett.107.085504

[76] S. Pipolo, M. Salanne, G. Ferlat, S. Klotz, A. M. Saitta, and F. Pietrucci, "Navigating at Will on the Water Phase Diagram,"

Phys. Rev. Lett., vol. 119, p. 245701, Dec 2017. URL: https://link.aps.org/doi/10.1103/PhysRevLett.119.245701

[77] E. V. Konstantinova and V. A. Skorobogatov, "Application of hypergraph theory in chemistry," *Discrete Mathematics*, vol. 235, no. 1, pp. 365 – 383, 2001, chech and Slovak 3. URL: https://www.sciencedirect.com/science/article/pii/S0012365X00002909

[78] D. Weininger, "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules," *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, Feb 1988. URL: https://pubs.acs.org/doi/abs/10.1021/ci00057a005

[79] M.-M. Deza and E. Deza, *Dictionary of Distances*. Walthm, MA, USA: Elsevier Science, Sep 2006. URL: https://www.elsevier.com/books/dictionary-of-distances/deza/978-0-444-52087-6

[80] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, Dec 1987. URL: https://doi.org/10.1007/BF02278710

[81] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109

[82] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. URL: https://doi.org/10.1137/0105003

[83] I. A. Blatov, E. V. Kitaeva, A. P. Shevchenko, and V. A. Blatov, "A universal algorithm for finding the shortest distance between systems of points," *Acta Crystallographica Section A*, vol. 75, no. 6, pp. 827–832, Nov 2019. URL: https://doi.org/10.1107/S2053273319011628

[84] B. Helmich and M. Sierka, "Similarity recognition of molecular structures by optimal atomic matching and rotational superposition," *Journal of Computational Chemistry*, vol. 33, no. 2, pp. 134–140, 2012. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21925

[85] A. Sadeghi, S. A. Ghasemi, B. Schaefer, S. Mohr, M. A. Lill, and S. Goedecker, "Metrics for measuring distances in configuration spaces," *The Journal of Chemical Physics*, vol. 139, no. 18, p. 184118, 2013. URL: https://doi.org/10.1063/1.4828704

[86] O. Trott and A. J. Olson, "AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *Journal of Computational Chemistry*, vol. 31, no. 2, pp. 455–461, 2010. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21334

[87] M. Griffiths, S. P. Niblett, and D. J. Wales, "Optimal Alignment of Structures for Finite and Periodic Systems," *Journal of Chemical Theory and Computation*, vol. 13, no. 10, pp. 4914–4931, 2017, pMID: 28841314. URL: https://doi.org/10.1021/acs.jctc.7b00543

[88] T. Eiter and H. Mannila, "Distance measures for point sets and their computation," *Acta Informatica*, vol. 34, no. 2, pp. 109–133, Feb 1997. URL: https://doi.org/10.1007/s002360050075

[89] C. Eckart, "Some Studies Concerning Rotating Axes and Polyatomic Molecules," *Phys. Rev.*, vol. 47, pp. 552–558, Apr 1935. URL: https://link.aps.org/doi/10.1103/PhysRev.47.552

[90] J. D. Louck and H. W. Galbraith, "Eckart vectors, Eckart frames, and polyatomic molecules," *Rev. Mod. Phys.*, vol. 48, pp. 69–106, Jan 1976. URL: https://link.aps.org/doi/10.1103/RevModPhys.48.69

[91] D. R. Flower, "Rotational superposition: a review of methods," *J. Mol. Graph. Model.*, vol. 17, no. 3-4, pp. 238–244, 1999. URL: https://europepmc.org/abstract/MED/10736782

[92] E. A. Coutsias and M. J. Wester, "RMSD and Symmetry," *Journal of Computational Chemistry*, vol. 40, no. 15, pp. 1496–1508, 2019. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.25802

[93] A. J. Hanson, "The quaternion-based spatial-coordinate and orientation-frame alignment problems," *Acta Crystallographica Section A*, vol. 76, no. 4, pp. 432–457, Jul 2020. URL: https://doi.org/10.1107/S2053273320002648

[94] B. F. Green, "The orthogonal approximation of an oblique structure in factor analysis," *Psychometrika*, vol. 17, no. 4, pp. 429–440, Dec 1952. URL: https://doi.org/10.1007/BF02288918

[95] H. L. Strauss and H. M. Pickett, "Conformational structure, energy, and inversion rates of cyclohexane and some related oxanes," *Journal of the American Chemical Society*, vol. 92, no. 25, pp. 7281–7290, 1970. URL: https://doi.org/10.1021/ja00728a009

[96] C. Fábri, E. Mátyus, and A. G. Császár, "Numerically constructed internal-coordinate Hamiltonian with Eckart embedding and its application for the inversion tunneling of ammonia," *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, vol. 119, pp. 84 – 89, 2014, frontiers in molecular vibrational calculations and computational spectroscopy. URL: https://www.sciencedirect.com/science/article/pii/S1386142513003223

[97] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. Opt. Soc. Am. A*, vol. 5, no. 7, pp. 1127–1135, Jul 1988. URL: https://josaa.osa.org/abstract.cfm?URI=josaa-5-7-1127

[98] N. Cliff, "Orthogonal rotation to congruence," *Psychometrika*, vol. 31, pp. 33–42, 1966. URL: https://doi.org/10.1007/BF02289455

[99] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, Sep 1976. URL: https://doi.org/10.1107/S0567739476001873

[100] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987. URL: https://doi.org/10.1109/TPAMI.1987.4767965

[101] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr 1987. URL: https://josaa.osa.org/abstract.cfm?URI=josaa-4-4-629

[102] S. K. Kearsley, "On the orthogonal transformation used for structural comparisons," *Acta Crystallographica Section A*, vol. 45, no. 2, pp. 208–210, Feb 1989. URL: https://doi.org/10.1107/S0108767388010128

[103] G. R. Kneller, "Superposition of Molecular Structures using Quaternions," *Molecular Simulation*, vol. 7, no. 1-2, pp. 113–119, 1991. URL: https://doi.org/10.1080/08927029108022453

[104] S. V. Krasnoshchekov, E. V. Isayeva, and N. F. Stepanov, "Determination of the Eckart molecule-fixed frame by use of the apparatus of quaternion algebra," *The Journal of Chemical Physics*, vol. 140, no. 15, p. 154104, 2014. URL: https://doi.org/10.1063/1.4870936

[105] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, Mar 1966. URL: https://doi.org/10.1007/BF02289451

[106] B. Maiseli, Y. Gu, and H. Gao, "Recent developments and trends in point set registration methods," *Journal of Visual Communication and Image Representation*, vol. 46, pp. 95 – 106, 2017. URL: https://www.sciencedirect.com/science/article/pii/S1047320317300743

[107] H. Zhu, B. Guo, K. Zou, Y. Li, K.-V. Yuen, L. Mihaylova, and H. Leung, "A Review of Point Set Registration: From Pairwise Registration to Groupwise Registration," *Sensors*, vol. 19, no. 5, p. 1191, Mar 2019. URL: https://dx.doi.org/10.3390/s19051191

[108] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992. URL: https://ieeexplore.ieee.org/document/121791

[109] H. Pottmann, Q.-X. Huang, Y.-L. Yang, and S.-M. Hu, "Geometry and Convergence Analysis of Algorithms for Registration of 3D Shapes," *International Journal of Computer Vision*, vol. 67, no. 3, pp. 277–296, May 2006. URL: https://doi.org/10.1007/s11263-006-5167-2

[110] N. J. Richmond, P. Willett, and R. D. Clark, "Alignment of three-dimensional molecules using an image recognition algorithm," *Journal of Molecular Graphics and Modelling*, vol. 23, no. 2, pp. 199–209, 2004. URL: https://www.sciencedirect.com/science/article/pii/S1093326304000592

[111] W. J. Allen and R. C. Rizzo, "Implementation of the Hungarian Algorithm to Account for Ligand Symmetry and Similarity in Structure-Based Design," *Journal of Chemical Information and Modeling*, vol. 54, no. 2, pp. 518–529, 2014, pMID: 24410429. URL: https://doi.org/10.1021/ci400534h

[112] A. Wagner and H.-J. Himmel, "aRMSD: A Comprehensive Tool for Structural Analysis," *Journal of Chemical Information and Modeling*, vol. 57, no. 3, pp. 428–438, 2017, pMID: 28191844. URL: https://doi.org/10.1021/acs.jcim.6b00516

[113] B. Temelso, J. M. Mabey, T. Kubota, N. Appiah-Padi, and G. C. Shields, "ArbAlign: A Tool for Optimal Alignment of Arbitrarily Ordered Isomers Using the Kuhn–Munkres Algorithm," *Journal of Chemical Information and Modeling*, vol. 57, no. 5, pp. 1045–1054, 2017, pMID: 28398732. URL: https://doi.org/10.1021/acs.jcim.6b00546

[114] M. Gunde, N. Salles, A. Hémeryck, and L. Martin-Samos, "IRA: A Shape Matching Approach for Recognition and Comparison of Generic Atomic Patterns," *Journal of Chemical Information and Modeling*, vol. 61, no. 11, pp. 5446–5457, 2021, pMID: 34704748. URL: https://doi.org/10.1021/acs.jcim.1c00567

[115] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*, ser. SIAM e-books. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009. URL: https://doi.org/10.1137/1.9781611972238

[116] A. Y. Dymarsky and K. N. Kudin, "Computation of the pseudorotation matrix to satisfy the Eckart axis conditions," *The Journal of Chemical Physics*, vol. 122, no. 12, p. 124103, 2005. URL: https://doi.org/10.1063/1.1864872

[117] D. J. Wales, J. P. K. Doye, A. Dullweber, M. P. Hodges, F. Y. N. F. Calvo, J. Hernández-Rojas, and T. F. Middleton, "The Cambridge Cluster Database." URL: https://www-wales.ch.cam.ac.uk/CCD.html

[118] X. Shao, X. Wu, and W. Cai, "Growth Pattern of Truncated Octahedra in AlN ($N \leq 310$) Clusters," *The Journal of Physical Chemistry A*, vol. 114, no. 1, pp. 29–36, 2010, pMID: 20014801. URL: https://doi.org/10.1021/jp906922v

[119] B. Brena and L. Ojamäe, "Surface Effects and Quantum Confinement in Nanosized GaN Clusters: Theoretical Predictions," *The Journal of Physical Chemistry C*, vol. 112, no. 35, pp. 13 516–13 523, 2008. URL: https://doi.org/10.1021/jp8048179

[120] Q. Liu, C. Xu, X. Wu, and L. Cheng, "Electronic shells of a tubular Au26 cluster: a cage–cage superatomic molecule based on spherical aromaticity," *Nanoscale*, vol. 11, pp. 13 227–13 232, 2019. URL: https://dx.doi.org/10.1039/C9NR02617G

[121] M. Rusishvili, L. Grisanti, S. Laporte, M. Micciarelli, M. Rosa, R. J. Robbins, T. Collins, A. Magistrato, and S. Baroni, "Unraveling the molecular mechanisms of color expression in anthocyanins," *Phys. Chem. Chem. Phys.*, vol. 21, pp. 8757–8766, 2019. URL: https://dx.doi.org/10.1039/C9CP00747D

[122] S. Smidstrup, A. Pedersen, K. Stokbro, and H. Jónsson, "Improved initial guess for minimum energy path calculations," *The Journal*

of Chemical Physics, vol. 140, no. 21, p. 214106, 2014. URL: https://doi.org/10.1063/1.4878664

[123] H. Jónsson, G. Mills, and K. W. Jacobsen, "Nudged elastic band method for finding minimum energy paths of transitions," in *Classical and Quantum Dynamics in Condensed Phase Simulations*. Singapore: WORLD SCIENTIFIC, Jun 1998, pp. 385–404. URL: https://www.worldscientific.com/doi/abs/10.1142/9789812839664_0016

[124] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, and R. M. Wentzcovitch, "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials," *Journal of Physics: Condensed Matter*, vol. 21, no. 39, p. 395502 (19pp), 2009. URL: https://www.quantum-espresso.org

[125] J.-M. Ducéré, A. Hemeryck, A. Estève, M. D. Rouhani, G. Landa, P. Ménini, C. Tropis, A. Maisonnat, P. Fau, and B. Chaudret, "A computational chemist approach to gas sensors: Modeling the response of SnO2 to CO, O2, and H2O Gases," *Journal of Computational Chemistry*, vol. 33, no. 3, pp. 247–258, 2012. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21959

[126] A. Ramstad, G. Brocks, and P. J. Kelly, "Theoretical study of the Si(100) surface reconstruction," *Phys. Rev. B*, vol. 51, pp. 14 504–14 523, May 1995. URL: https://link.aps.org/doi/10.1103/PhysRevB.51.14504

[127] H. Shigekawa, K. Miyake, M. Ishida, K. Hata, H. Oigawa, Y. Nannichi, R. Yoshizaki, A. Kawazu, T. Abe, T. Ozawa, and T. Nagamura, "Phase Transition between $c(4 \times 2)$ and $p(2 \times 2)$ Structures of the Si(100) Surface at 6 K Caused by the Fluctuation of Phase Defects on Dimer Rows due to Dimer Flip-Flop Motion," *Japanese Journal of Applied Physics*, vol. 35, no. Part 2, No. 8B, pp. L1081–L1084, aug 1996. URL: https://doi.org/10.1143/jjap.35.l1081

[128] A. Hemeryck, A. J. Mayne, N. Richard, A. Estève, Y. J. Chabal, M. Djafari Rouhani, G. Dujardin, and G. Comtet, "Difficulty for oxygen to incorporate into the silicon network during initial O2 oxidation of

Si(100)-(2×1)," *The Journal of Chemical Physics*, vol. 126, no. 11, p. 114707, 2007. URL: https://doi.org/10.1063/1.2566299

[129] A. Hemeryck, N. Richard, A. Estève, and M. D. Rouhani, "Multi-scale modeling of oxygen molecule adsorption on a Si(100)-p(2×2) surface," *Journal of Non-Crystalline Solids*, vol. 353, no. 5, pp. 594–598, 2007, siO2, Advanced Dielectrics and Related Devices 6. URL: https://www.sciencedirect.com/science/article/pii/S0022309306013937

[130] A. Hemeryck, N. Richard, A. Estève, and M. Djafari Rouhani, "Diffusion of oxygen atom in the topmost layer of the Si(100) surface: Structures and oxidation kinetics," *Surface Science*, vol. 601, no. 11, pp. 2339–2343, 2007. URL: https://www.sciencedirect.com/science/article/pii/S003960280700249X

[131] A. Hémeryck, A. Estève, N. Richard, M. D. Rouhani, and G. Landa, "A kinetic Monte Carlo study of the initial stage of silicon oxidation: Basic mechanisms-induced partial ordering of the oxide interfacial layer," *Surface Science*, vol. 603, no. 13, pp. 2132–2137, 2009. URL: https://www.sciencedirect.com/science/article/pii/S0039602809002957

[132] N. Salles, N. Richard, N. Mousseau, and A. Hemeryck, "Strain-driven diffusion process during silicon oxidation investigated by coupling density functional theory and activation relaxation technique," *The Journal of Chemical Physics*, vol. 147, no. 5, p. 054701, 2017. URL: https://doi.org/10.1063/1.4996206

[133] G. T. Barkema and N. Mousseau, "Event-Based Relaxation of Continuous Disordered Systems," *Phys. Rev. Lett.*, vol. 77, pp. 4358–4361, Nov 1996. URL: https://link.aps.org/doi/10.1103/PhysRevLett.77.4358

[134] R. Malek and N. Mousseau, "Dynamics of Lennard-Jones clusters: A characterization of the activation-relaxation technique," *Phys. Rev. E*, vol. 62, pp. 7723–7728, Dec 2000. URL: https://link.aps.org/doi/10.1103/PhysRevE.62.7723

[135] A. Jay, C. Huet, N. Salles, M. Gunde, L. Martin-Samos, N. Richard, G. Landa, V. Goiffon, S. De Gironcoli, A. Hémeryck, and N. Mousseau, "Finding Reaction Pathways and Transition States: r-ARTn and d-ARTn as an Efficient and Versatile Alternative to String Approaches," *Journal of Chemical Theory and Computation*, vol. 16, no. 10, pp. 6726–6734, 2020, pMID: 32794748. URL: https://doi.org/10.1021/acs.jctc.0c00541

[136] S. Plimpton, “Fast Parallel Algorithms for Short-Range Molecular Dynamics,” *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, 1995. URL: https://www.sciencedirect.com/science/article/pii/S002199918571039X

[137] M. S. Daw and M. I. Baskes, “Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals,” *Phys. Rev. B*, vol. 29, pp. 6443–6453, Jun 1984. URL: https://link.aps.org/doi/10.1103/PhysRevB.29.6443

Abstract

The long-term evolution of a large-scale atomic system can be simulated by approximating it as a series of events, also called jumps, with a kinetic Monte Carlo (kMC) algorithm. Particular problems arise when the system to be simulated cannot be assigned to a rigid, periodic lattice. Off-lattice kMC approaches can be used to overcome this difficulty. For off-lattice kMC software, desirable characteristics are the ability to efficiently reuse information from its event catalogue and to be accurate throughout the simulation. To enable these characteristics, a structural comparison technique is needed at two stages of each kMC simulation step: when identifying the possible events, and when executing the events in the simulation. This thesis presents the development of the necessary structural comparison technique, the so-called Iterative Rotations and Assignments (IRA) shape matching algorithm, and details of its implementation and use within a general off-lattice kinetic Monte Carlo kernel.

As an independent algorithm, the IRA algorithm is able to solve the shape matching problem for any two arbitrarily-rotated and/or distorted atomic structures. The IRA algorithm is based on the idea of reducing the phase space of possible rotations to a set of points, given by the atomic vectors of the structure itself. The algorithm iterates through all of the rotation points thus generated and selects the rotation for which a particular distance function (the Hausdorff distance function) gives the minimum value. To address and solve the problem of atomic assignment between two atomic structures, generally called the Linear Assignment Problem (LAP), within the shape matching problem, the IRA algorithm uses the Constrained Shortest Distance Assignments (CShDA) algorithm also developed and presented in this thesis. Due to the ability of CShDA to solve assignments for structures containing different numbers of atoms, the IRA algorithm can also be applied to structural fragments. When inserted into the specific situation of off-lattice kMC software, we establish that the IRA algorithm is an efficient structural comparison technique, at both critical stages of kMC simulation. In addition, IRA is able to efficiently and accurately identify all symmetries of kMC events, thus granting a statistically correct execution of the move directions. The off-lattice kMC approach using the shape matching algorithm developed here (IRA) also allows the simulation of processes which change the total number of atoms in a system, namely adsorption and desorption processes. Several examples of simulations using the off-lattice kMC software that incorporates IRA and CShDA algorithms are discussed, along with the novelties of our approach. We also discuss the successful and unsuccessful resolutions of the difficulties encountered in the examples. The thesis concludes with the possible future directions for the work, including an exciting fully independent learning-on-the-fly approach to kMC.

Résumé

L'évolution à long terme d'un système atomique à grande échelle peut être simulée en l'approchant comme une série d'événements, également appelés sauts, à l'aide d'un algorithme de Monte Carlo cinétique (kMC). Des problèmes particuliers se posent lorsque le système à simuler ne peut être assigné à un réseau rigide et périodique. Les approches kMC hors réseau peuvent être utilisées pour surmonter cette difficulté. Pour un logiciel kMC hors réseau, les caractéristiques souhaitables sont la capacité de réutiliser efficacement les informations de son catalogue d'événements et d'être précis tout au long de la simulation. Pour permettre ces caractéristiques, une technique de comparaison structurelle est nécessaire à deux étapes de chaque simulation kMC : lors de l'identification des événements possibles, et lors de l'exécution des événements dans la simulation. Cette thèse présente le développement de la technique de comparaison structurelle nécessaire, l'algorithme IRA (Iterative Rotations and Assignments) de mise en correspondance des formes, ainsi que les détails de sa mise en œuvre et de son utilisation dans un noyau général de Monte Carlo cinétique hors réseau.

En tant qu'algorithme indépendant, l'algorithme IRA est capable de résoudre le problème de correspondance de forme pour deux structures atomiques arbitrairement tournées et/ou déformées. L'algorithme IRA est basé sur l'idée de réduire l'espace de phase des rotations possibles à un ensemble de points, donnés par les vecteurs atomiques de la structure elle-même. L'algorithme itère à travers tous les points de rotation ainsi générés et sélectionne la rotation pour laquelle une fonction de distance particulière (la fonction de distance de Hausdorff) donne la valeur minimale. Pour aborder et résoudre le problème de l'affectation atomique entre deux structures atomiques, généralement appelé le Linear Assignment Problem (LAP), dans le cadre du problème de correspondance des formes, l'algorithme IRA utilise l'algorithme Constrained Shortest Distance Assignments (CShDA) également développé et présenté dans cette thèse. En raison de la capacité de CShDA à résoudre les assignations pour des structures contenant différents nombres d'atomes, l'algorithme IRA peut également être appliqué à des fragments structurels. Lorsqu'il est inséré dans la situation spécifique du logiciel kMC hors réseau, nous établissons que l'algorithme IRA est une technique de comparaison structurelle efficace, aux deux étapes critiques de la simulation kMC. En outre, l'IRA est capable d'identifier efficacement et précisément toutes les symétries des événements kMC, garantissant ainsi une exécution statistiquement correcte des directions de déplacement. L'approche kMC hors réseau utilisant l'algorithme de correspondance de forme développé ici (IRA) permet également la simulation de processus qui modifient le nombre total d'atomes dans un système, à savoir les processus d'adsorption et de désorption. Plusieurs exemples de simulations utilisant le logiciel kMC hors réseau qui incorpore les algorithmes IRA et CShDA

sont discutés, ainsi que les nouveautés de notre approche. Nous discutons également des résolutions réussies et non réussies des difficultés rencontrées dans les exemples. La thèse se termine par les directions futures possibles pour le travail, y compris une approche passionnante d'apprentissage à la volée totalement indépendante pour kMC.