Highlights

kMCpy: A Python Package to Simulate Transport Properties in Solids with Kinetic Monte Carlo

Zeyu Deng,Tara P. Mishra,Weihang Xie,Daanyal Ahmed Saeed,Gopalakrishnan Sai Gautam,Pieremanuele Canepa

- Kinetic Monte Carlo

kMCpy: A Python Package to Simulate Transport Properties in Solids with Kinetic Monte Carlo

Dr. Zeyu Deng[a,*], Dr. Tara P. Mishra[a,b], Mr. Weihang Xie[a], Mr. Daanyal Ahmed Saeed[e], Prof. Gopalakrishnan Sai Gautam[c] and Prof. Pieremanuele Canepa[a,b,d,*]

[a]National University of Singapore, Department of Materials Science and Engineering, 9 Engineering Drive 1, Singapore 117575, Singapore
[b]Singapore-MIT Alliance for Research and Technology, 1 CREATE Way, 10-01 CREATE Tower, Singapore 138602, Singapore
[c]Indian Institute of Science, Department of Materials Engineering, Indian Institute of Science, Bengaluru 560012, India
[d]National University of Singapore, Department of Chemical and Biomolecular Engineering, 4 Engineering Drive 4, Singapore 117585, Singapore
[e]University of California, Berkeley, 2437 Piedmont Avenue, Berkeley, California 94704, United States of America

ARTICLE INFO

Keywords:
Kinetic Monte Carlo
Transport Property
Kinetics
Cluster Expansion
Ion Transport

ABSTRACT

Understanding ion transport in functional materials is crucial to unravel complex chemical reactions, improve rate performance of materials for energy storage and conversion, and optimize catalysts. To model ionic transport, atomistic simulations, including molecular dynamics (MD) and kinetic Monte Carlo (kMC) have been developed and applied to shed light on intricate materials science and chemistry problems. Typically, kMC simulations are utilized to a lower extent compared to MD due to a lack of systematic workflows to construct a model for predicting transition rates. Here, we propose kMCpy, a light-weight, customizable, and modular python package to compute the ionic transport properties in crystalline materials using kMC that can be combined with a (local) cluster expansion Hamiltonian derived from first-principles calculations. kMCpy is versatile with respect to any type of crystalline material, bearing any dimensionality, such as 1D, 2D and 3D. kMCpy provides: i) a comprehensive workflow to enumerate all possible migration events in crystalline systems, ii) to derive transition rates efficiently and at the accuracy of first-principles calculations, and iii) a robust kMC solver to study kinetic phenomena in materials. The workflow implemented in kMCpy provides a systematic way to compute highly-accurate kinetic properties, which can be used in high-throughput simulations for the discovery and optimization of novel functional materials.

1. Introduction

Quantifying ionic transport properties in materials is crucial in a wide variety of applications, such as molecular & protein biology[1, 2, 3], energy[4, 5, 6], chemical reactions[7, 8], and solid mechanics.[9, 10, 11, 12] The advancement of computer hardware, theoretical models, and suitable software that scale and parallelize with available computing resources, have enabled the evaluation of ionic transport in solid-state materials.[13, 14] A widely used atomistic simulation technique to probe kinetic properties is molecular dynamics (MD),[13, 15] which propagates the state of a given system as a function of time, where individual particles (atoms) interact via Newton's laws of motion. MD has been implemented in a wide variety of software packages[16, 17, 18, 19, 20, 21], where the accuracy of MD is dependent on the accuracy of force evaluations. Forces acting on atoms in a MD simulation is accessed from accurate (but expensive) first principles calculations, or inexpensive (and less accurate) interatomic potentials (i.e., force fields).

An alternative to MD is kinetic Monte Carlo (kMC, also known as dynamic Monte Carlo)[22, 23, 24], which has been extensively applied to study materials kinetics including, rechargeable batteries[25, 26, 27, 28], solid-oxide fuel cells[29, 30], catalysis[31, 32], crystal growth[33], vacancy diffusion in alloys[34, 35], thin film growth[36], and fluid flow[37]. kMC is particularly useful in quantifying ionic transport in battery materials, as demonstrated by van der Ven *et al.* in electrode materials, such as Li_xCoO_2[25], Li_xTiS_2[38], $Li_{1+x}Ti_2O_4$[39]. Notably, Deng *et al.*[28] used kMC to estimate the conductivity of Na in solid electrolytes: $Na_{1+x}Zr_2P_xSi_{3-x}O_{12}$, as a function of Na content and temperature, eventually sampling a vast compositional, spatial, and temporal scale. kMC can also be used to examine the structural evolution of nano-particles as well, as demonstrated by Li *et al.*[35].

kMC is based on a stochastic algorithm which randomly samples various microstates of a given system, utilizing the ergodic principle to arrive at statistically-averaged transport properties. Thus, the chief advantage of kMC over MD is the ability of kMC to access "long" timescales (∼ms) and "large" lengthscales (∼ μm) compared to what is usually possible in MD (∼ μs, ∼nm)[40].

Two main kMC algorithms have been proposed: i) kMC with rejection (r-kMC)[41] and ii) rejection-free kMC (rf-kMC)[22]. The former algorithm is similar to the Metropolis algorithm[41] which can select or reject a transition event using a probability estimate. In rf-kMC, a transition event is always executed based on a "list" of probabilities. Thus, rf-kMC is computationally efficient compared to r-kMC,

*Corresponding author.
msedz@nus.edu.sg (Z. Deng); pcanepa@nus.edu.sg (P. Canepa)
ORCID(s): 0000-0003-0109-9367 (Z. Deng); 0000-0002-3000-2555 (T.P. Mishra); 0000-0002-6498-2328 (W. Xie); 0000-0003-4620-4829 (D.A. Saeed); 0000-0002-1303-0976 (G.S. Gautam); 0000-0002-5168-9253 (P. Canepa)

especially when transition rates are low (i.e., event rejection rates are high). There are several software packages to perform kMC simulations [42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53], including codes that target higher efficiency kMC algorithms[54, 55, 56, 57, 58, 59, 60], and those that construct novel models to compute accurate transition rates[61, 62, 63, 64, 65].

Compared to MD, kMC is a fairly general simulation algorithm which can be applied to coarse grain material properties and contribute to multi scale modelling efforts[66, 67, 68]. However, in kMC all transition events should be known *a priori*, and a model is typically needed to compute transition rates between different microstates swiftly. Therefore, using kMC to study ionic transport usually needs a workflow that is typically "tailored" to a given system. Such a workflow should include modules to generate all possible transition events, a comprehensive model to compute transition rate for each event (swiftly and accurately), and a robust kMC solver.

Here, we present our `python`-based code `kMCpy`[1] to simulate the kinetic properties of materials, with inputs from first principles calculations. Specifically, we implement a local cluster expansion (LCE) model[25, 26] to compute migration barriers in crystalline materials (within the transition state theory framework), where the model is fitted to calculated barriers from accurate first principles calculations. `kMCpy` contains a rf-kMC solver and related `python` classes to extract ion transport properties, such as diffusivities, conductivities, etc. In addition, `kMCpy` includes the following features:

- `kMCpy` is fully developed using `python`[69, 70].
- Cross Platform: `kMCpy` supports most "mainstream" operating systems, such as Windows, macOS, and Linux, in both x86/64 and ARM architectures.
- Modular Code Structure: `kMCpy` is written as modulus, which can be easily modified and ported to any specific application.
- Ease of Use: All input and output data are supplied using human-readable JSON format, which is easily parsed and generated by computers.
- Performance: The computationally-intensive routines of `kMCpy` are translated into optimized machine code at runtime using `Numba` [71], which is a just-in-time (JIT) compiler desgined to increase computational performance of `python` codes.

The paper is structured as follows: Sec. 2 deals with the theoretical background to compute transport properties in crystalline materials, Sec. 3 provides an overview of the `kMCpy` code, Sec. 4 describes the performance of `kMCpy`, and Sec. 5 compiles our concluding remarks and possible future developments of `kMCpy`. All nomenclature used through the manuscript is listed in Sec. A.

[1] `kMCpy` is an open-source code developed under the MIT license and can be accessed at: https://github.com/caneparesearch/kMCpy

2. Theoretical Background

Ionic transport in solids is a stochastic process, occurring through a series of correlated/non-correlated migration events (or ionic 'hops'), which can be effectively modeled using the kMC formalism. The local energy landscape around the migrating ion determines the ease of migration within the solid. Quantifying macroscopic ionic transport of a given chemical species in a given material is usually done in terms of ionic diffusivities and/or ionic conductivities (see below), both of which can be evaluated using kMC.[25]

Before understanding how a typical kMC simulation progresses, we briefly overview some of the fundamentals of ion transport in solids. The macroscopic measure of mobility of a migrating species is determined by the chemical diffusivity (D_c), which relates to the flux and conductivity of the species through Fick's law[72, 73], as stated in Eq. 1.

$$J = -D_c \nabla C \tag{1}$$

where J is the flux of the migrating species, and C is the composition of the mobile species defined as the number of migrating ions per unit volume. The chemical diffusivity of the migrating ion relates to the jump diffusivity (D_J) through the thermodynamic factor Θ of Eq. 2.

$$D_c = D_J \Theta \tag{2}$$

Θ measures the deviation of the interaction between migrating ions from ideal behavior and is given in Eq. 3

$$\Theta = \frac{\partial \left(\frac{\mu}{k_B T} \right)}{\partial \ln x} \tag{3}$$

where μ is the chemical potential, k_B is the Boltzmann constant, and x is the molefraction of the migrating species.

D_J of Eq. 2 is proportional to the mean squared displacement of the center of mass of the mobile species, as mathematically described in Eq. 4.

$$D_J = \frac{\left(\sum_i \vec{r}_i \right)^2}{2dNt} \tag{4}$$

where d is the dimensionality of the diffusion process, N is the number of diffusing species, and t is the time taken for diffusion. Furthermore, from the square of displacements of the migrating ions, one can also calculate the tracer diffusivity (D^*, Eq. 5), which excludes cross-correlation effects between the migrating ion [26].

$$D^* = \frac{\sum_i \vec{r}_i^{\,2}}{2dNt} \tag{5}$$

The ionic conductivity σ can be then computed via the Nernst–Einstein relationship:

$$\sigma = \frac{e^2 C D_J}{k_B T} \tag{6}$$

where C is the number of migrating species per unit volume.

Therefore, the cross-correlation between migrating ions can be quantified from the ratio of D^* and D_J, which is called the Haven's ratio (H_R) [74]. Note that H_R does not measure the correlation between subsequent hops of a single ion that is migrating, i.e., the deviation of the trajectory of a single migrating ion from a fully random walk. This deviation from a fully random walk is measured by the correlation factor (f) of Eq. 7.

$$f = \frac{\sum_i \vec{r}_i^{\,2}}{Nna^2} \tag{7}$$

where $\vec{r}_i$ is the net displacement of a migrating ion after n hops, while a is the average distance for a single hop. Therefore, an accurate calculation of the ionic transport properties requires the sampling of a large-enough number of migration events, which requires that all mobile species are tracked during the simulation.

One of the important parameters required by a kMC simulation are the migration barriers (E_bs), which are the energy barrierers that the mobile ion must overcome to complete a successful hop. E_b ultimately determines the probability of occurrence a given ionic hop. Typically, E_bs are evaluated using the nudged elastic band (NEB) method in combination with density functional theory (DFT). [75, 76]. In a NEB calculation, one performs a constrained relaxation of a specific number of virtually connected "images", between the initial and final positions of a migration event, along a guessed minimum energy pathway (MEP). The relaxation is constrained to maintain a uniform spacing between the images (i.e., as uniform as possible), through the addition of fictitious spring forces. Other tools, such as force fields and machine-learned interatomic potentials, can also be used to determine E_b instead of DFT, and kMCpy is also compatible with such tools.

Note that E_b in solids not only depends on the local environment of the migrating ion but also the direction of the hop. Hence, to remove any direction-dependence of a hop, we resort to the so-called kinetically resolved activation barrier (E_{KRA}) of Eq. 8 proposed by van der Ven et al. [25].

$$E_{\mathrm{KRA}} = E_b[i \rightarrow j] - \frac{1}{2}\Delta E_{\mathrm{end}} \tag{8}$$

where $E_b[i \rightarrow j]$ is the calculated E_b (e.g., with NEB) for a site i to site j hop and ΔE_{end} is the absolute difference between the computed DFT total energies of the initial and final positions (i.e., the endpoints).

In principle, the E_{KRA} has to be calculated for all possible migration events that can occur in a solid (as the local bonding/coordination environment changes for example). However, calculating E_b for all possible hops via NEB calculations is computationally intensive and often impractical. One strategy to circumvent the computational obstacles of NEB calculations is that of the LCE approach. A LCE is normally used to construct a simplified lattice Hamiltonian, which can generate approximate E_b quickly (by estimating a E_{KRA}), based on the local configuration(s) of the moving and non moving species, which is defined in Eq. 9 [25, 77].

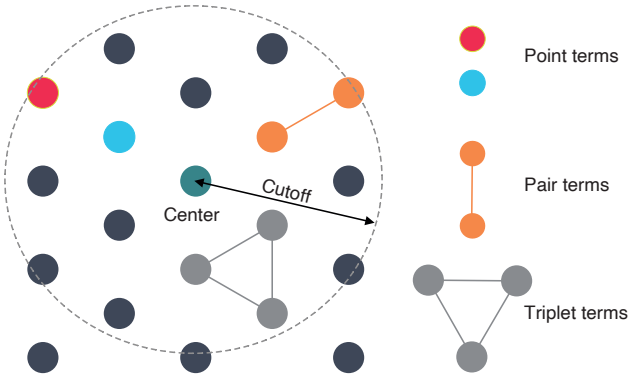

Figure 1: Example of clusters that are typically encountered in a cluster expansion overlaid over a representative lattice. Within a cutoff distance from the center of a given lattice site, local orbits are drawn which extract the different interactions, such as point (red and blue dots), pair (orange dots), and triplet (grey dots). For each lattice, only the symmetrically-unique clusters are used to construct the cluster expansion.

$$E_{\mathrm{KRA}} = V_0 + \sum_{\alpha} V_{\mathrm{orbit}} \phi_{\mathrm{orbit}} \tag{9}$$

where

$$\phi_{\mathrm{orbit}} = \prod_{i \in \mathrm{orbit}} \sigma_i \tag{10}$$

Here, an orbit implies a cluster of sites, which for example, can be a point, a pair, or a triplet, as depicted in Fig. 1. σ is the occupation variable of a given site within a cluster, whose value depends on the basis set used. For example, σ can take the value of -1 or $+1$ to indicate the presence or absence of an atom at a given site. To account for local interactions, orbits are usually truncated at finite distances from a given site. In Eq. 9, the terms V_0 and V_{orbit} are the kinetic effective cluster interactions (KECIs). The values of the KECIs are determined by fitting Eq. 9 a set of NEB-calculated E_{KRA}. Note that instead of a LCE, surface models, thin film models, or coarse grain models [66, 67] can also be used for estimating E_b.

After determining V_{orbit} and V_0 in Eq. 9, one proceeds with kMC simulations, whose workflow is shown schematically in Fig. 2(a). In kMCpy, we have implemented the rf-kMC method, also known as the Bortz-Kalos-Lebowitz (BKL) algorithm [22]. Specifically, we list the set of all possible migration events in a given solid and their corresponding probabilities, amongst which one migration event is selected. Once a hop is selected, the hop is always executed, and subsequently, the list of possible migration events is updated.

The typical procedure for the BKL method is summarized in the text below and in Fig. 2. Note that Fig. 2 does not include the equilibration process.

1. *Initialization*: In this step a representative structure is generated, which contains a fixed concentration

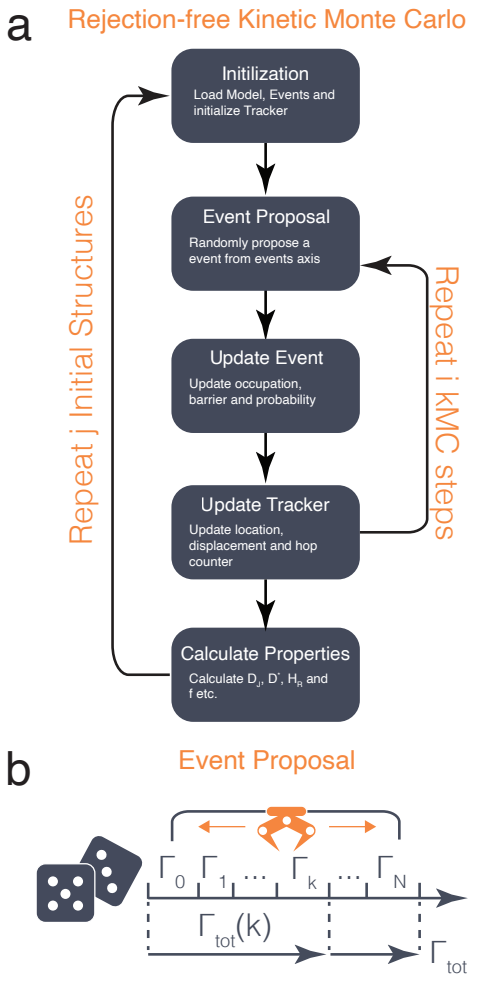

Figure 2: a Flow chart of the kMC process. i kMC steps are repeated with each kMC simulation started from a different initial structure (i.e., j initial structures in total). **b** All migration events are listed on a hypothetical axis, with the solid line representing their hopping probabilities (Γ). An event no. k is then randomly proposed based on a random number ρ. $\Gamma_{\text{tot}}(k)$ is a cumulative sum of events from no. 0 to no. k (i.e., $\Gamma_{tot}(k) = \sum_{m=0}^{k} \Gamma_m$). Γ_{tot} is the sum of hopping probabilities of all migration events.

of mobile ions and vacancies (assuming a vacancy-mediated migration mechanism). These structures can be obtained from canonical Monte Carlo (CMC), grand-canonical Monte Carlo (GCMC) [78] simulations, random structure generators [34], or other structural enumeration techniques. During the initialization, a tracker is also set, which keeps track of the migration observables, such as, the mean squared displacement (MSD) of the diffusing species, the location of the center of mass, D_J, and f.

2. *Event proposal*: A list of probabilities (Γ_m) is generated for all possible migrating paths (m) available for all mobile ions in the simulation box. This list also includes hops which may not be feasible. For example, if both the initial and final sites of a migration path are occupied by an atom (instead of one of the sites being vacant), the value of Γ_m is set to 0. The hopping probability (Γ) for each migration event is calculated using the transition state theory [79] via Eq. 11.

$$\Gamma = \nu^* \exp\left(\frac{-E_b}{k_B T}\right) \tag{11}$$

From Eqs. 8 and 9, it is possible to quickly generate E_b) for every possible hop. ν^* is the prefactor and is usually assumed to be of the order of 10^{11} to 10^{13} Hz [25, 80]. T is the simulation temperature.
Following the generation of the probability list, a migration event (k) is chosen based on a random number ($0 < \rho < 1$), such that it satisfies Eq. 12.

$$\frac{1}{\Gamma_{\text{tot}}} \sum_{m=1}^{k-1} \Gamma_m < \rho \leq \frac{1}{\Gamma_{\text{tot}}} \sum_{m=1}^{k} \Gamma_m \tag{12}$$

where Γ_{tot} is the sum of all the individual probabilities of all migration events. This step is shown schematically in Fig. 2(b).

3. *Update event and tracker*: After an event is chosen and executed, the time step (δt) is updated by drawing another random number ($0 < \zeta < 1$) as shown in Eq. 13.

$$\delta t = -\frac{1}{\Gamma_{\text{tot}}} \ln \zeta \tag{13}$$

Subsequently, the occupation vector, the new event list and their corresponding probabilities, the displacement vector(s), the location(s) of the mobile ions, the location of the center of mass, and the hop counter are updated.

A single kMC pass includes repeating the event proposal, update event, and update tracker steps the same number of times as the number of mobile ions in the initialized structure. Generally, a large number of kMC passes are required to accurately predict transport properties. For example, Deng *et al.*, undertook $\approx 10^6$ kMC passes to simulate Na-transport in superionic conductor over a millisecond scale [28]. After running a sufficiently large number of kMC passes, properties, such as, D_J, D^*, H_R, and f are estimated. Thus, a collection of kMC passes for a single initial structure is referred to as a kMC run. To get a better estimate of the transport properties at a given composition, kMCpy also calculates the properties as the initial structure is varied j times (i.e., j kMC runs). This ensures that the transport properties calculated represent the statistical estimate that is observed in experiments better.

3. Overview of kMCpy

3.1. Workflow

The workflow of kMCpy is shown in Fig. 3. The specific python classes for each action are shown as grey boxes on the right-hand side of Fig. 3. kMCpy contains functions to analyse

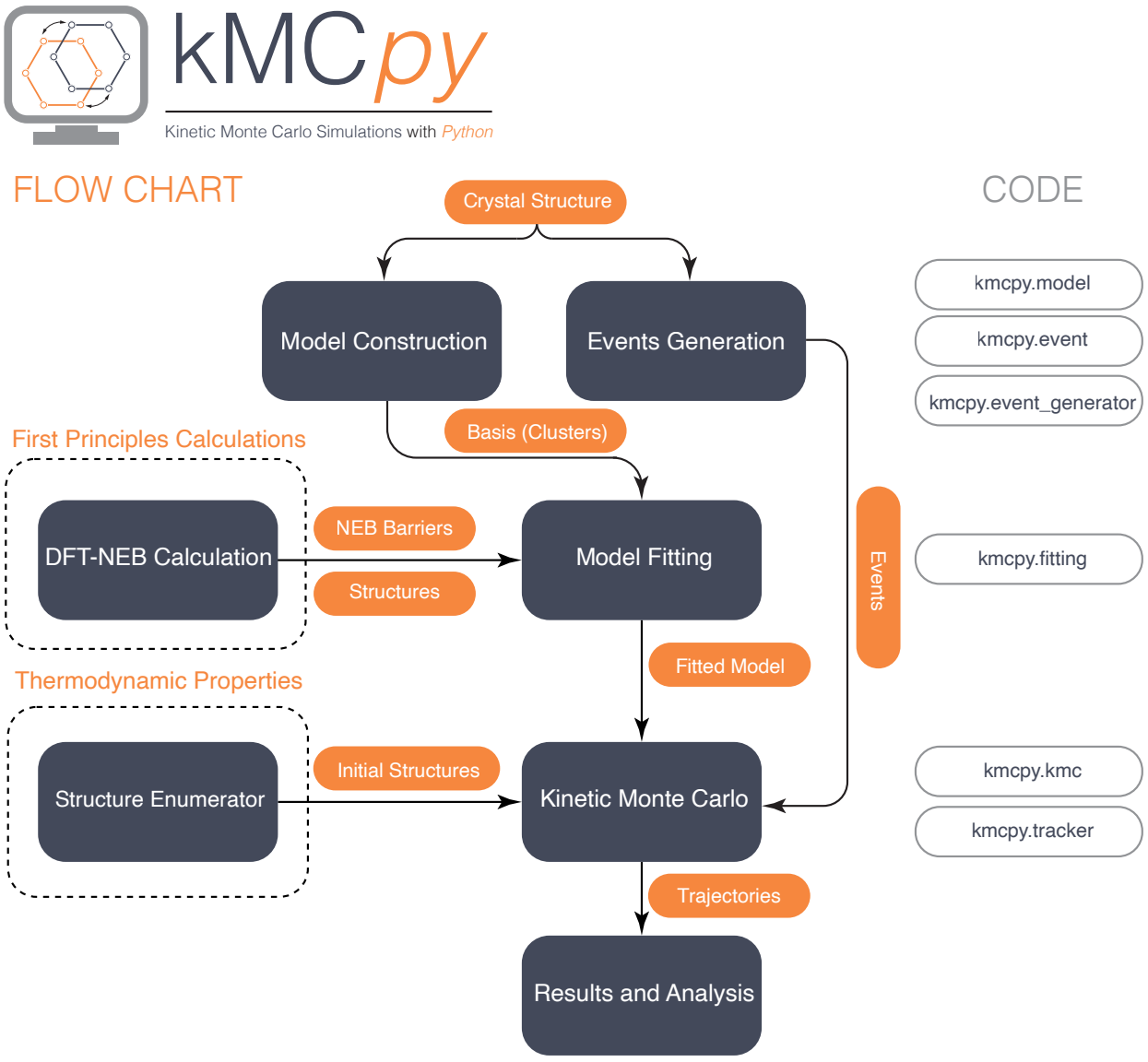

Figure 3: Left Workflow of `kMCpy` package. **Right** `python` classes of `kMCpy` used for each stage of execution. Note that the initial structures for running kMC simulations are obtained from a structure enumerator. Migration barriers are computed from DFT-NEB calculations (dashed boxes). In the current version, only the LCE model has been implemented.

crystal structures, construct a LCE model and generate a list of possible migration events (see Sec. 3.2 and 3.3). Starting from a list of DFT-NEB computed barriers, `kMCpy` fits a LCE model (see Sec. 3.4). The constructed model and events are then used to run kMC simulations with input structures from that are either the thermodynamic ground state(s) or any other user defined structures (Sec. 3.5). The trajectory of each mobile species are stored and analysed with a `Tracker` class as implemented in `kMCpy` (Sec. 3.6). Examples of input and output files are provided in Sec. 3.7. All `python` classes mentioned in the following sections can be stored in a human-readable JSON format and can be re-initialized after each stage.

3.2. Model Construction

Before running a kMC simulation, a representative lattice model must be constructed to compute the barriers efficiently for any local environment, and the current version of `kMCpy` uses the LCE framework. However, the modular nature of `kMCpy` is such that other lattice models can also be used.

The LCE is implemented in the `LocalClusterExpansion` class in `kMCpy.model`. The local environment in the LCE model is described using a migration unit (`MigrationUnit`), which is defined as a representative collection of sites centered around a given activated state (AS) where possible migration events can take place. The migration unit is generated using the user-specified cutoff radius. As a result, when a local environment is imported, an "occupation vector" (see

Sec. A) will be constructed based on the atomic species at each site.

Subsequently, all clusters within a migration unit are found by enumerating all points, pairs, triplets, etc. via a cutoff radius (specified by the user) for each type of cluster. All symmetrically equivalent clusters are then grouped as orbits, which become elements in the "correlation vector" (see Sec. A). Clusters and orbits are coded into the `kMCpy.model` as the `Cluster` and the `Orbit` classes. In addition to the species and the atomic coordinates, `Cluster` and `Orbit` both have functions to compute "correlation" (see Sec. A) for a given orbit based on the occupation of sites.

3.3. Generating Events

A kMC simulation needs a list of all possible migration events within a given simulation cell prior to its execution. Therefore, in `kMCpy`, we handle migration events using the `Event` class in `kmcpy.event`. `Event` stores the indices of two sites (e.g., initial and final sites) involved in the migration event, as well as the indices of all sites within the surrounding migration unit, i.e., local environment indices. `Event` also has built-in functions to compute the correlation vector, the migration barrier and the hopping probability after an occupation vector has been assigned.

We enumerate all `Event` objects prior to the kMC run using a wrapper function, `event_generator.generate_events()`, which receives the `LocalClusterExpansion` as input and loops through all migration units in the whole simulation cell to generate all possible events. From a given identifier of mobile species (i.e., the `mobile_ion_identifier` parameter), two sites indices involved in the migration event, namely, sites that the ion hops from and hops into, are identified. Subsequently, the indices of all sites in the current migration unit are stored for calculating the migration barrier.

3.4. Model Fitting

In order to fit DFT-NEB barriers using the LCE model, we have implemented the `Fitting` class in `kmcpy.fitting`. This function performs fitting by interfacing with the python package `scikit-learn`[81]. `kMCpy` also stores fitted results (i.e., the KECIs) in a portable JSON format. The current implementation of `kMCpy` uses the "LASSO" regression[82] to perform fitting. Indeed, LASSO limits the selection of orbits in the fit to the most important ones. LASSO requires a user-specified α parameter to reduce the total number of selected orbits. The `Fitting` class stores the fitting history, e.g. α and weights used during LASSO regression, for keeping a record and to fine tune the LCE.

Note that a LCE typically fits the E_{KRA} that is obtained from NEB calculations. Therefore, the ΔE_{end} term in Eq. 8 can be computed either from CMC (e.g., by interfacing other codes, such as "CASM" [83]) or by fitting a separate LCE model. `kMCpy` has the flexibility to adopt either approaches to determine E_{end}. In case a LCE is used for fitting E_{end} also, then the E_{end} data extracted from NEB calculations is used as an input for the fitting process.

3.5. Kinetic Monte Carlo

The `KMC` class in `kmcpy.kmc` can be used to perform kMC simulations. Multiple (e.g., 50) kMC runs should be done to eliminate the dependency of the results on the starting configurations. Initial structures of kMC runs are taken either from the thermodynamic ground state(s) (e.g., from CMC or GCMC simulations), or from a structure enumerator (see Fig. 3). Auxiliary tools are provided in `kmcpy.tools.gather_mc_data` to extract occupation vectors from a structure in the crystallographic information file (CIF) format.

The general process of rf-kMC is described in Fig. 2, and an example of a standard output of both the initialization and the execution processes of kMC is shown in Fig. 4a and b. The `KMC` class is firstly initialized using a size specification of the simulation (super)cell, the initial occupations, the fitted model, the generated events, and a reference crystal structure. When the LCE model is used, information about clusters, orbits, and the KECIs are provided. `kMCpy` then "walks" through all available migration events and evaluates the occupations, correlation vectors, and hopping frequencies, given a simulation temperature and a ν^*.

Upon initialization of the kMC, the `KMC.run()` function is called to perform the kMC simulation by supplying the total number of equilibration and sampling steps, respectively. The equilibration steps are not explicitly shown in Fig. 2a. A `Tracker` object is initialized once the equilibration process is complete (see Sec. 3.6). As shown in Fig. 2b, for each kMC step, an event k is randomly proposed using `KMC.propose()`, based on Eq. 12.

After the proposed event is executed, the related occupations, correlations, and hopping frequencies are updated. Since a given site may be involved in multiple migration events, all events with sites associated with the proposed event are updated. The number of events that require updation after a proposed event is defined as the coupling strength of events, which can influence the computational performance of kMC (see Sec. 4). We use a pre-computed table (`event_kernel`) to quickly identify all events that need to be updated.

3.6. Tracking Diffusion

To follow the displacements of all mobile species with respect to their original positions and to count the number of hops of each mobile ion, `kMCpy` uses a `Tracker` class in `kmcpy.tracker`. This class is activated only after the equilibration is complete. The `Tracker` is initialized with the initial occupation vectors, a reference crystal structure of the simulation cell, the formal charge on migrating species, the dimensionality of the overall diffusion process, the average hopping distances (in Å), the simulation temperature, and ν^*. The initial location of each migrating species is recorded and their displacement vectors and counters are set to zero during initialization. During each kMC step, `Tracker.update()` updates the displacement vector (taking into account periodic boundary conditions) and the hopping counter of the migrating species involved in a proposed event.

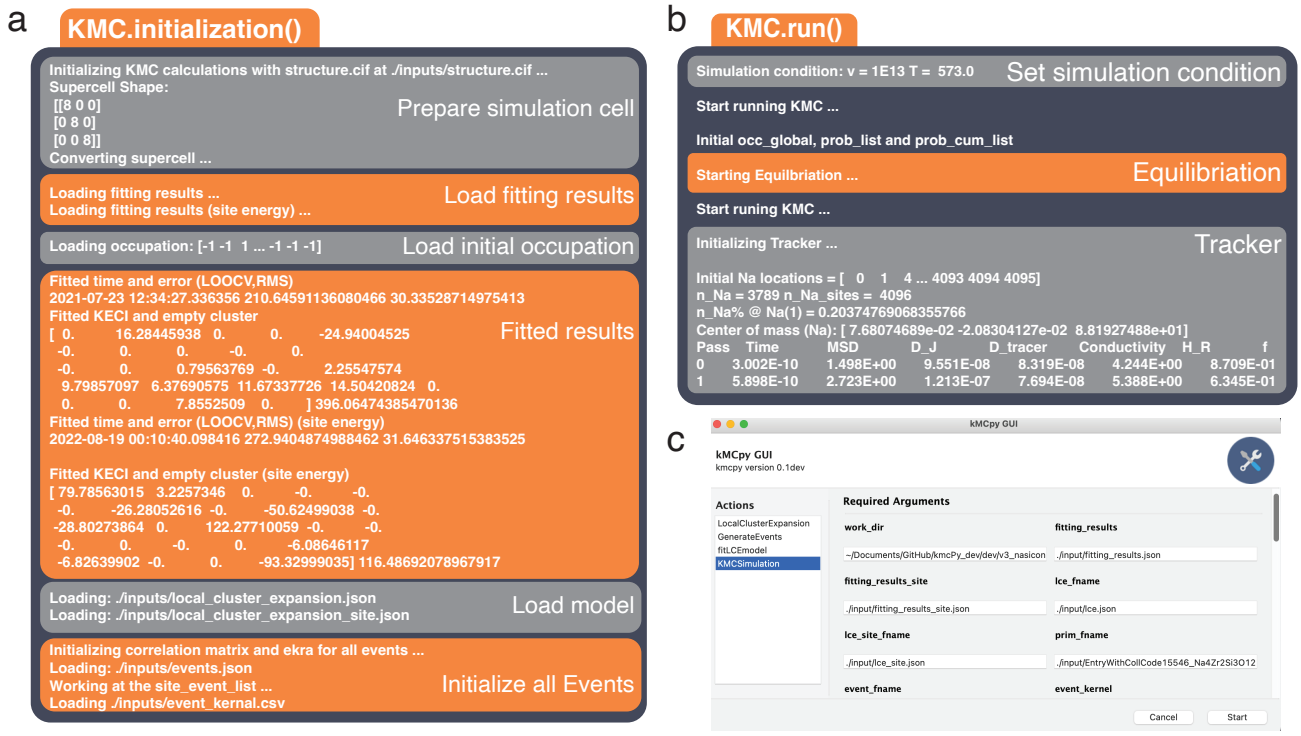

Figure 4: Screenshots of initialization (`KMC.initialization()`, in **a**) and execution (`KMC.run()`, in **b**) of kMC. `KMC.initialization()` prints the input parameters and `KMC.run()` shows the computed results. A `Tracker` object is intialized and subsequently called at the end of `KMC.run()`. **c** shows the graphic user interface (GUI) of `kMCpy` relying on the `python` library `Gooey`.

Using Eqs. 2-7 in Sec. 2, `Tracker` computes transport properties, such as MSD, D_J, D^*, σ, f, and H_R from the displacements of all migrating ions. The chemical diffusivity, D_c (Eq. 2) can be computed once Θ is identified for systems with variable compositions, such as electrodes [26]. `Tracker.summary()` and `Tracker.write_results()` routines print and save the simulation results, respectively.

3.7. Input and Output Files

The inputs required by `kMCpy` (Fig. 3a) can be prepared in JSON format, through the use of Jupyter notebooks for instance. Sample input files are provided in the `input_example` folder of our Github repository (Footnote 1). Further, there is a command line wrapper to execute `kMCpy` from the command line, which can be found in `kmcpy.executable.wrapper`. Users can also customize their own workflow by importing specific modules, as described in the previous sub-sections.

An example of a standard output of `KMC.initialization()` and `KMC.run()` are shown in Fig. 4a and b. `kMCpy` prints the information imported from the JSON input files and sets the parameters described in Sec. 3.5, Sec. 3.6, and Fig. 2.

`kmcpy.executable.gui_wrapper` offers a graphical user interface (GUI) as shown in Fig. 4c, which builds upon the `python` library `Gooey` [84]. This GUI covers all required and optional arguments for each step, providing a convenient way to test different parameters and for educational/demonstration purposes as well.

A required task in the Actions box must be chosen in the GUI interface, in accordance to the descriptions in Sec. 3.2 to Sec. 3.5. Next, all essential input parameters required for this task must be provided. For example, if `KMCSimulation` is chosen, one must provide: the work directory, the initial occupation, the original crystal structure, the fitted LCE model, the generated events, a value of ν^*, and the simulation temperature. The documentation is available via a website (`https://kmcpy.readthedocs.io`) with details on all input parameters to run `kMCpy` [85]. By clicking the "Start" button, `kMCpy` will perform the selected task with the standard output of the simulation (similar to the command line output of Fig. 4a and b) displayed in a separate pop-up window.

4. Performance of `kMCpy`

`kMCpy` has been developed in `python`, a high-level, human-interpretable language that combines flexibility and ease of programming. Since `python` is one of the most widely used programming languages[69, 70] both in the fields of materials informatics and data science, it provides a set of readily available tools and libraries that can be used to accelerate the development of new codes and libraries. Among them, we utilise a JIT compiler, `Numba` [71], to increase the computational performance of `kMCpy`. Specifically, `Numba` translates the most numerically demanding part of `kMCpy` into optimized machine code.

We emphasize that kMCpy is a serial code, i.e., a single kMC run is executed on a single CPU core. However, multiple kMC runs can be executed simultaneously on a multi-core platform, such as a high performance computing server or on the cloud. For example, different initial structures can be generated for a system and a kMC run for each initial structure can be run in parallel, thus reducing compute time.

Computationally, the intensive part of kMCpy is evaluating the correlation vector for each Event. Therefore, the size of the basis set (i.e., number of clusters and orbits), the total number events, and the coupling strengths between different events (see Sec. A) can crucially influence the determination of the correlation vector and the computational performance. For example, the basis-set size controls the computational cost of updating the correlation vector of a single event, whereas the total number of unique events and the coupling strengths between events set the total number of events to be updated during each kMC step. These quantities are usually coupled with each other, i.e., larger cutoff radii usually lead to larger basis sets, and in turn, stronger coupling between events, resulting in an increased computational cost of the kMC run.

We benchmarked[2] the computational performance of kMCpy, with the data compiled in Table 1, which shows the time required to prepare inputs and run a very short simulation on a test system[3]. The process of input preparation includes construction of the model, fitting of the model, and events generation, which in total takes less than 10s. 100 kMC passes (51,200 steps per pass) on this model takes ∼3 minutes, indicating that the time required for preparing the input is generally marginal compared to the kMC simulation itself.

Eq. 14 shows fitted dependencies between the simulation time and the three major factors:

$$t \propto f_{\text{Numba}}(N_{\text{cell}}) \times N_{\text{cluster}}^{1.16} \times N_{\text{cell}}^{1.01} \times N_{\text{coupling}}^{1.10} \quad (14)$$

where N_{cluster}, N_{cell}, N_{coupling} are the number of clusters, size of the supercell, and the coupling strength, respectively. f_{Numba} denotes the acceleration effect using the Numba routines on the computational time. The benchmark results are shown in Fig. 5.

The average simulation time using different cell sizes (indicated by total number of atoms per simulation box) is depicted in Fig. 5a. Without Numba, the elapsed time per kMC step remains approximately constant $\sim 10^{-1}$ s (dashed black line). Numba accelerates significantly the kMC simulation by factor of ∼2 (10^{-3} s, solid black line). The speedup by Numba is weakened when the simulation cell becomes larger. Therefore, Numba can enable access to longer and larger scale simulations with kMCpy [28]. As 1 kMC pass is just the total number of available sites within the simulation cell, the run time per kMC pass grows linearly when the cell size (i.e., number of atoms) become larger. Fig. 5b and c demonstrate the effect of the coupling strength between events as well as the basis set size, which also contribute a quasi-linear increase towards the run time per kMC pass. These results show that the time complexity of the implemented kMC algorithm is $O(\sim N)$.

Table 1
Time (in s) distribution for bootstrapping and running a kMC simulation of a test model. Details of hardware information and input model to perform these benchmarks are mentioned in Footnotes 2 and 3.

Action	Time Spent (s)
Model Generation (212 clusters)	2.1
Model Fitting (19 orbits)	2.6
Events Generation (6,144 events)	1.31
kMC simulation (100 passes/51,200 steps)	168

5. Conclusion

In summary, we presented kMCpy, a light-weight open-source python package to perform kMC simulations of ionic transport in crystalline solids, with inputs from DFT calculations. kMCpy and its implemented workflow provide a framework to the scientific community to predict transport properties of any crystalline solid with high accuracy and performance. The design of kMCpy should facilitate its use on most available computational platforms from standard laptops to high performance supercomputers. The modular framework makes it highly customizable and easily programmable. By utilizing the JIT compiler – Numba, kMCpy achieves high computational performance. Both the input and the output files of kMCpy rely on the human-readable JSON format, which is easy to distribute. Future developments of kMCpy include: i) utilizing GPU-based acceleration for better performance, ii) developing a thermodynamic (CMC/GCMC) module and a structure enumerator, and iii) adding additional models for the evaluation of E_{KRA} that are alternative to LCE.

A. Nomenclature

- **Site**: $i \in \{0, 1, ..., N-1\}$ is a site in a simulation cell with N sites. i is a unique global index of a site.
- **Occupation**: in a Chebyshev basis σ_i has a value of ± 1 for site i, e.g., occupied (-1) or unoccupied $(+1)$, or species A (-1) and species B $(+1)$.
- **Occupation Vector**: $\vec{\sigma} = [\sigma_0, \sigma_1, ...\sigma_{N-1}]$ is a vector of occupations in a simulation cell.
- **Migration unit**: $M = [i_0, i_1, ..., i_m]$ is a collection of all sites within a specific cutoff radii around the centre of a migration unit, where m is the total number of sites within that migration unit. There are multiple migration units in the simulation cell.

[2] All benchmarks were performed on a 2020-year model 13-inch Apple MacBook Pro with a M1 chipset (8 core CPU + 8 core GPU) and 16 GB of RAM.

[3] A LCE model for Na ion migration was built with a ∼6 Å cutoff radius for point, pair and triplet clusters on $Na_{1+x}Zr_2P_xSi_{3-x}O_{12}$, yielding a total of 19 unique orbits and 212 possible clusters. The LCE model was fitted with data from DFT-NEB calculations. 6144 Na-ion hopping events were generated in a $8 \times 8 \times 8$ supercell lattice.

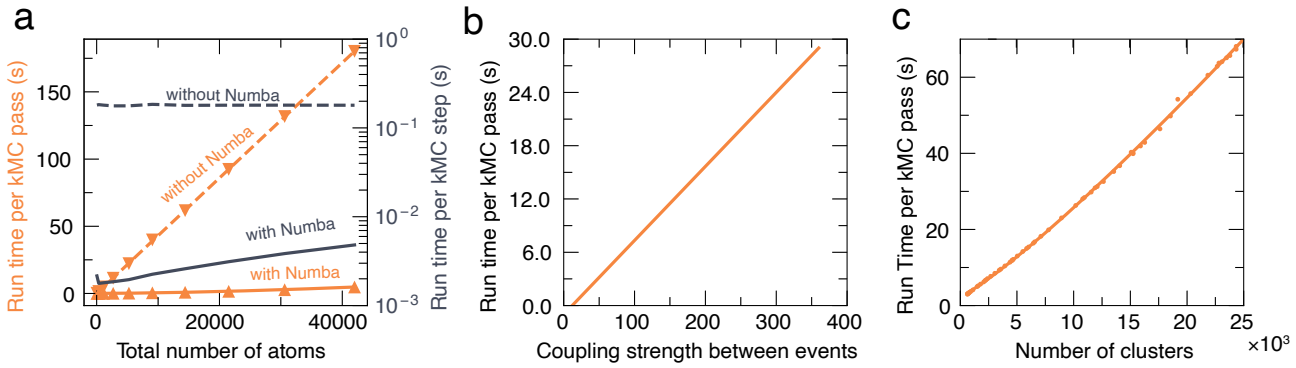

Figure 5: a shows the average computing time, per kMC pass (left y-axis, orange lines) and per kMC step (right y-axis, black lines), as a function of total number of atoms for kMC simulations of $Na_{1+x}Zr_2P_xSi_{3-x}O_{12}$, scaling from $1\times1\times1$ (42 atoms) to $10\times10\times10$ supercells (42,000 atoms). The details of this model can be found in Ref. [28]. Solid and dashed lines refers to the time consumption with and without `Numba`, respectively. The LCE model contains 19 unique orbits and 212 possible clusters, with a coupling strength of 12. **b** shows the relationship between the time consumption and coupling strength between events (defined in Section A) performed on a $8\times8\times8$ supercell lattice. **c** describes the effect of basis set size (number of clusters/orbits in LCE model) on simulation time, using the $8\times8\times8$ supercell.

- **Sublattice Site**: $i \in \{0, 1, ..., n-1\}$ is a site within a migration unit with n sites.
- **Distance Matrix: D**: Distance matrix of a migration unit is a $m \times m$ matrix where m has been defined above. Matrix elements d_{ij} are the Cartesian distances between site i and j within a migration unit.
- **Cluster**: $[i_0, i_1, ...i_n]$ is a collection of sublattice sites (i) within a migration unit with a length of n. Presently there are four types of cluster implemented in the code:

  Point: a cluster containing 1 site, $n = 1$;

  Pair: a cluster containing 2 sites, $n = 2$;

  Triplet: a cluster containing 3 sites, $n = 3$;

  Quadruplet: a cluster containing 4 sites, $n = 4$;

  Note, the order-size of these clusters can be easily extended beyond 4.
- **Cluster Function**: $\phi_\alpha(\vec{\sigma}) = \prod_{i\in\alpha}\sigma_i$ is a product of all occupations of all sites that belong to a cluster.
- **Orbit**, $[\alpha[0], \alpha[1], ...\alpha[m]]$ is a collection of symmetrically equivalent clusters with a multiplicity of m within a migration unit.
- **Correlation**: $\phi_O(\vec{\sigma}) = \sum_{\alpha\in O}\phi_\alpha(\vec{\sigma})$ is the summation of cluster functions of all symmetrically equivalent clusters within a migration unit that belongs to an orbit.
- **Correlation Vector**:

$$\vec{\phi}(\vec{\sigma}) = [\phi_{O[0]}(\vec{\sigma}), \phi_{O[1]}(\vec{\sigma}), ..., \phi_{O[n]}(\vec{\sigma})] \quad (15)$$

  is a collection of all correlations for each orbit within a migration unit with a length of n. This is also the basis-set size.
- **Cluster Expansion Model**:

$$E(\vec{\sigma}) = V_0 + \sum_\alpha V_\alpha\phi_\alpha(\vec{\sigma}) \quad (16)$$

  where E is total energy (typically the DFT total energy, and V_0 and V_α are called effective cluster interactions (ECIs) for each cluster, which are fitted from first-principles calculations. The summation polynomials are usually truncated to specific cluster size (e.g., quadruplet, quintuplet). All clusters belong to the same orbit shares the same V_α.
- **Local Cluster Expansion Model** uses a local cluster expansion model,

$$E_{\mathrm{KRA}}(\vec{\sigma}) = K_0 + \sum_\alpha K_\alpha\phi_\alpha(\vec{\sigma}) \quad (17)$$

  where E_{KRA} is the kinetic resolved activation energy barrier which is independent of migration directions. K_0 and K_α are called kinetic effective cluster interactions (KECIs), which are fitted from first-principles NEB calculations. The directional dependent activation energy can further be recovered using

$$E_b = E_{\mathrm{KRA}}(\vec{\sigma}_{AS}) + \frac{1}{2}\Delta E_{\mathrm{end}} \quad (18)$$

  where $\vec{\sigma}_{AS}$ is the occupation vector at the activated state and E_b is the activation energy barrier from initial to final images. ΔE_{end} is the total energy difference between the final and initial images, respectively:

$$\Delta E_{\mathrm{end}} = E(\vec{\sigma}_{\mathrm{final}}) - E(\vec{\sigma}_{\mathrm{initial}}) \quad (19)$$

- **Event** is a swap of occupation values between two hopping sites.

- **Pass** is defined as the total number of mutable sites in the simulation cell.
- **Coupling Strength Between Events** is the total number of events that need to be updated after an event has been executed.

Acknowledgment

We acknowledge funding from the National Research Foundation under his NRF Fellowship NRFF12-2020-0012. The computational work was performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

CRediT authorship contribution statement

Zeyu Deng: Conceptualization of the study, Methodology, Software development, Writing the Initial Draft. **Tara P. Mishra:** Software development, Writing the Initial Draft. **Weihang Xie:** Software development, Writing the Initial Draft. **Daanyal Ahmed Saeed:** Software development, Writing the Initial Draft. **Gopalakrishnan Sai Gautam:** Methodology, Writing the Initial Draft. **Pieremanuele Canepa:** Conceptualization of the study, Methodology, Project Management, Procurement of Funding, Writing the Initial Draft.

References

[1] J. A. McCammon, B. R. Gelin, M. Karplus, Dynamics of folded proteins, Nature 267 (5612) (1977) 585–590. doi:10.1038/267585a0.
URL http://www.nature.com/articles/267585a0

[2] M. Karplus, J. Kuriyan, Molecular dynamics and protein function, Proceedings of the National Academy of Sciences 102 (19) (2005) 6679–6685. doi:10.1073/pnas.0408930102.
URL https://pnas.org/doi/full/10.1073/pnas.0408930102

[3] J. L. Klepeis, K. Lindorff-Larsen, R. O. Dror, D. E. Shaw, Long-timescale molecular dynamics simulations of protein structure and function, Current Opinion in Structural Biology 19 (2) (2009) 120–127. doi:10.1016/j.sbi.2009.03.004.
URL https://linkinghub.elsevier.com/retrieve/pii/S0959440X09000372

[4] S. P. Ong, O. Andreussi, Y. Wu, N. Marzari, G. Ceder, Electrochemical Windows of Room-Temperature Ionic Liquids from Molecular Dynamics and Density Functional Theory Calculations, Chemistry of Materials 23 (11) (2011) 2979–2986. doi:10.1021/cm200679y.
URL https://pubs.acs.org/doi/10.1021/cm200679y

[5] Y. Mo, S. P. Ong, G. Ceder, Insights into Diffusion Mechanisms in P2 Layered Oxide Materials by First-Principles Calculations, Chemistry of Materials 26 (18) (2014) 5208–5214. doi:10.1021/cm501563f.
URL https://pubs.acs.org/doi/10.1021/cm501563f

[6] Y. Wang, W. D. Richards, S. P. Ong, L. J. Miara, J. C. Kim, Y. Mo, G. Ceder, Design principles for solid-state lithium superionic conductors, Nature Materials 14 (10) (2015) 1026–1031. doi:10.1038/nmat4369.
URL https://www.nature.com/articles/nmat4369

[7] D. R. Herschbach, Molecular Dynamics of Elementary Chemical Reactions(Nobel Lecture), Angewandte Chemie International Edition in English 26 (12) (1987) 1221–1243. doi:10.1002/anie.198712211.
URL https://onlinelibrary.wiley.com/doi/10.1002/anie.198712211

[8] I. R. Craig, D. E. Manolopoulos, Chemical reaction rates from ring polymer molecular dynamics, The Journal of Chemical Physics 122 (8) (2005) 084106. doi:10.1063/1.1850093.
URL http://aip.scitation.org/doi/10.1063/1.1850093

[9] J. F. Lutsko, Stress and elastic constants in anisotropic solids: Molecular dynamics techniques, Journal of Applied Physics 64 (3) (1988) 1152–1154. doi:10.1063/1.341877.
URL http://aip.scitation.org/doi/10.1063/1.341877

[10] F. F. Abraham, D. Brodbeck, W. E. Rudge, X. Xu, A molecular dynamics investigation of rapid fracture mechanics, Journal of the Mechanics and Physics of Solids 45 (9) (1997) 1595–1619. doi:10.1016/S0022-5096(96)00103-2.
URL https://linkinghub.elsevier.com/retrieve/pii/S0022509696001032

[11] C. Yang, U. Tartaglino, B. N. Persson, A multiscale molecular dynamics approach to contact mechanics, The European Physical Journal E 19 (1) (2006) 47–58. doi:10.1140/epje/e2006-00004-9.
URL https://link.springer.com/10.1140/epje/e2006-00004-9

[12] H. Rafii-Tabar, H. Shodja, M. Darabi, A. Dahi, Molecular dynamics simulation of crack propagation in fcc materials containing clusters of impurities, Mechanics of Materials 38 (3) (2006) 243–252. doi:10.1016/j.mechmat.2005.06.006.
URL https://linkinghub.elsevier.com/retrieve/pii/S0167663605001043

[13] D. Frenkel, B. Smit, Understanding molecular simulation: from algorithms to applications, 2nd Edition, no. 1 in Computational science series, Academic Press, San Diego, 2002.

[14] R. W. Balluffi, S. M. Allen, W. C. Carter, R. A. Kemper, Kinetics of materials, J. Wiley & Sons, Hoboken, N.J, 2005.

[15] T. Hansson, C. Oostenbrink, W. van Gunsteren, Molecular dynamics simulations, Current Opinion in Structural Biology 12 (2) (2002) 190–196. doi:10.1016/S0959-440X(02)00308-1.
URL https://linkinghub.elsevier.com/retrieve/pii/S0959440X02003081

[16] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, S. J. Plimpton, LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Computer Physics Communications 271 (2022) 108171. doi:10.1016/j.cpc.2021.108171.
URL https://linkinghub.elsevier.com/retrieve/pii/S0010465521002836

[17] G. te Velde, F. M. Bickelhaupt, E. J. Baerends, C. Fonseca Guerra, S. J. A. van Gisbergen, J. G. Snijders, T. Ziegler, Chemistry with ADF, Journal of Computational Chemistry 22 (9) (2001) 931–967. doi:10.1002/jcc.1056.
URL https://onlinelibrary.wiley.com/doi/10.1002/jcc.1056

[18] J. C. Phillips, D. J. Hardy, J. D. C. Maia, J. E. Stone, J. V. Ribeiro, R. C. Bernardi, R. Buch, G. Fiorin, J. Hénin, W. Jiang, R. McGreevy, M. C. R. Melo, B. K. Radak, R. D. Skeel, A. Singharoy, Y. Wang, B. Roux, A. Aksimentiev, Z. Luthey-Schulten, L. V. Kalé, K. Schulten, C. Chipot, E. Tajkhorshid, Scalable molecular dynamics on CPU and GPU architectures with NAMD, The Journal of Chemical Physics 153 (4) (2020) 044130. doi:10.1063/5.0014475.
URL http://aip.scitation.org/doi/10.1063/5.0014475

[19] E. Aprà, E. J. Bylaska, W. A. de Jong, N. Govind, K. Kowalski, T. P. Straatsma, M. Valiev, H. J. J. van Dam, Y. Alexeev, J. Anchell, V. Anisimov, F. W. Aquino, R. Atta-Fynn, J. Autschbach, N. P. Bauman, J. C. Becca, D. E. Bernholdt, K. Bhaskaran-Nair, S. Bogatko, P. Borowski, J. Boschen, J. Brabec, A. Bruner, E. Cauat, Y. Chen, G. N. Chuev, C. J. Cramer, J. Daily, M. J. O. Deegan, T. H. Dunning, M. Dupuis, K. G. Dyall, G. I. Fann, S. A. Fischer, A. Fonari, H. Frachtl, L. Gagliardi, J. Garza, N. Gawande, S. Ghosh, K. Glaesemann, A. W. GÄ¶tz, J. Hammond, V. Helms, E. D. Hermes, K. Hirao, S. Hirata, M. Jacquelin, L. Jensen, B. G. Johnson, H. Jónsson, R. A. Kendall, M. Klemm, R. Kobayashi, V. Konkov, S. Krishnamoorthy, M. Krishnan, Z. Lin, R. D. Lins, R. J. Littlefield, A. J. Logsdail, K. Lopata, W. Ma, A. V. Marenich, J. M. del Campo, D. Mejia-Rodriguez, J. E. Moore, J. M. Mullin, T. Nakajima, D. R. Nascimento, J. A. Nichols, P. J. Nichols, J. Nieplocha, A. O. de-la Roza, B. Palmer, A. Panyala, T. Pirojsirikul, B. Peng, R. Peverati,

J. Pittner, L. Pollack, R. M. Richard, P. Sadayappan, G. C. Schatz, W. A. Shelton, D. W. Silverstein, D. M. A. Smith, T. A. Soares, D. Song, M. Swart, H. L. Taylor, G. S. Thomas, V. Tipparaju, D. G. Truhlar, K. Tsemekhman, T. V. Voorhis, Á. Vázquez-Mayagoitia, P. Verma, O. Villa, A. Vishnu, K. D. Vogiatzis, D. Wang, J. H. Weare, M. J. Williamson, T. L. Windus, K. Woliński, A. T. Wong, Q. Wu, C. Yang, Q. Yu, M. Zacharias, Z. Zhang, Y. Zhao, R. J. Harrison, NWChem: Past, present, and future, The Journal of Chemical Physics 152 (18) (2020) 184102. doi:10.1063/5.0004997.
URL http://aip.scitation.org/doi/10.1063/5.0004997

[20] R. Salomon-Ferrer, D. A. Case, R. C. Walker, An overview of the Amber biomolecular simulation package: Amber biomolecular simulation package, Wiley Interdisciplinary Reviews: Computational Molecular Science 3 (2) (2013) 198–210. doi:10.1002/wcms.1121.
URL https://onlinelibrary.wiley.com/doi/10.1002/wcms.1121

[21] T. D. Kühne, M. Iannuzzi, M. Del Ben, V. V. Rybkin, P. Seewald, F. Stein, T. Laino, R. Z. Khaliullin, O. Schütt, F. Schiffmann, D. Golze, J. Wilhelm, S. Chulkov, M. H. Bani-Hashemian, V. Weber, U. Borštnik, M. Taillefumier, A. S. Jakobovits, A. Lazzaro, H. Pabst, T. Müller, R. Schade, M. Guidon, S. Andermatt, N. Holmberg, G. K. Schenter, A. Hehn, A. Bussy, F. Belleflamme, G. Tabacchi, A. Glöß, M. Lass, I. Bethune, C. J. Mundy, C. Plessl, M. Watkins, J. VandeVondele, M. Krack, J. Hutter, CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations, The Journal of Chemical Physics 152 (19) (2020) 194103. doi:10.1063/5.0007045.
URL http://aip.scitation.org/doi/10.1063/5.0007045

[22] A. Bortz, M. Kalos, J. Lebowitz, A new algorithm for Monte Carlo simulation of Ising spin systems, Journal of Computational Physics 17 (1) (1975) 10–18. doi:10.1016/0021-9991(75)90060-1.
URL https://linkinghub.elsevier.com/retrieve/pii/0021999175900601

[23] D. T. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions, Journal of Computational Physics 22 (4) (1976) 403–434. doi:10.1016/0021-9991(76)90041-3.
URL https://linkinghub.elsevier.com/retrieve/pii/0021999176900413

[24] D. T. Gillespie, Exact stochastic simulation of coupled chemical reactions, The Journal of Physical Chemistry 81 (25) (1977) 2340–2361. doi:10.1021/j100540a008.
URL https://pubs.acs.org/doi/10.1021/j100540a008

[25] A. Van der Ven, G. Ceder, M. Asta, P. D. Tepesch, First-principles theory of ionic diffusion with nondilute carriers, Physical Review B 64 (18) (2001) 184307. doi:10.1103/PhysRevB.64.184307.
URL https://link.aps.org/doi/10.1103/PhysRevB.64.184307

[26] A. Van Der Ven, Z. Deng, S. Banerjee, S. P. Ong, Rechargeable Alkali-Ion Battery Materials: Theory and Computation, Chemical Reviews 120 (14) (2020) 6977–7019. doi:10.1021/acs.chemrev.9b00601.
URL https://pubs.acs.org/doi/abs/10.1021/acs.chemrev.9b00601

[27] P. Xiao, G. Henkelman, Kinetic Monte Carlo Study of Li Intercalation in $LiFePO_4$, ACS Nano 12 (1) (2018) 844–851. doi:10.1021/acsnano.7b08278.
URL https://pubs.acs.org/doi/10.1021/acsnano.7b08278

[28] Z. Deng, T. P. Mishra, E. Mahayoni, Q. Ma, A. J. K. Tieu, O. Guillon, J.-N. Chotard, V. Seznec, A. K. Cheetham, C. Masquelier, G. S. Gautam, P. Canepa, Fundamental investigations on the sodium-ion transport properties of mixed polyanion solid-state battery electrolytes, Nature Communications 13 (1) (2022) 4470. doi:10.1038/s41467-022-32190-7.
URL https://www.nature.com/articles/s41467-022-32190-7

[29] R. Pornprasertsuk, T. Holme, F. B. Prinz, Kinetic Monte Carlo Simulations of Solid Oxide Fuel Cell, Journal of The Electrochemical Society 156 (12) (2009) B1406. doi:10.1149/1.3232209.
URL https://iopscience.iop.org/article/10.1149/1.3232209

[30] A. Modak, M. Lusk, Kinetic Monte Carlo simulation of a solid-oxide fuel cell: I. Open-circuit voltage and double layer structure, Solid State Ionics 176 (29-30) (2005) 2181–2191. doi:10.1016/j.ssi.2005.06.007.
URL https://linkinghub.elsevier.com/retrieve/pii/S0167273805002614

[31] M. Andersen, C. Panosetti, K. Reuter, A practical guide to surface kinetic Monte Carlo simulations, Frontiers in Chemistry 7 (APR) (2019) 1–24, arXiv: 1904.02561. doi:10.3389/fchem.2019.00202.

[32] M. Pineda, M. Stamatakis, Kinetic Monte Carlo simulations for heterogeneous catalysis: Fundamentals, current status, and challenges, The Journal of Chemical Physics 156 (12) (2022) 120902. doi:10.1063/5.0083251.
URL https://aip.scitation.org/doi/10.1063/5.0083251

[33] C.-H. Huang, L. Gharaee, Y. Zhao, P. Erhart, J. Marian, Mechanism of nucleation and incipient growth of Re clusters in irradiated W-Re alloys from kinetic Monte Carlo simulations, Physical Review B 96 (9) (2017) 094108. doi:10.1103/PhysRevB.96.094108.
URL https://link.aps.org/doi/10.1103/PhysRevB.96.094108

[34] A. Evteev, E. Levchenko, I. Belova, G. Murch, Shrinking kinetics by vacancy diffusion of hollow binary alloy nanospheres driven by the Gibbs–Thomson effect, Philosophical Magazine 88 (10) (2008) 1525–1541. doi:10.1080/14786430802213413.
URL http://www.tandfonline.com/doi/abs/10.1080/14786430802213413

[35] C. Li, T. Nilson, L. Cao, T. Mueller, Predicting activation energies for vacancy-mediated diffusion in alloys using a transition-state cluster expansion, Physical Review Materials 5 (1) (2021) 013803. doi:10.1103/PhysRevMaterials.5.013803.
URL https://link.aps.org/doi/10.1103/PhysRevMaterials.5.013803

[36] X. Han, R. McAfee, J. C. Yang, Development of a Versatile Kinetic Monte Carlo Code to Simulate Physical Processes in Thin Film Nucleation and Growth, Multidiscipline Modeling in Materials and Structures 3 (1) (2007) 43–54. doi:10.1163/157361107781360068.
URL https://www.emerald.com/insight/content/doi/10.1163/157361107781360068/full/html

[37] M. Apostolopoulou, R. Day, R. Hull, M. Stamatakis, A. Striolo, A kinetic Monte Carlo approach to study fluid transport in pore networks, The Journal of Chemical Physics 147 (13) (2017) 134703. doi:10.1063/1.4985885.
URL http://aip.scitation.org/doi/10.1063/1.4985885

[38] A. Van der Ven, J. C. Thomas, Q. Xu, B. Swoboda, D. Morgan, Nondilute diffusion from first principles: Li diffusion in Li x TiS 2, Physical Review B 78 (10) (2008) 104306. doi:10.1103/PhysRevB.78.104306.
URL https://link.aps.org/doi/10.1103/PhysRevB.78.104306

[39] J. Bhattacharya, A. Van der Ven, Phase stability and nondilute Li diffusion in spinel Li 1 + x Ti 2 O 4, Physical Review B 81 (10) (2010) 104304. doi:10.1103/PhysRevB.81.104304.
URL https://link.aps.org/doi/10.1103/PhysRevB.81.104304

[40] Y. Gao, T. P. Mishra, S.-H. Bo, G. Sai Gautam, P. Canepa, Design and Characterization of Host Frameworks for Facile Magnesium Transport, Annual Review of Materials Research 52 (1) (2022) 129–158. doi:10.1146/annurev-matsci-081420-041617.
URL https://www.annualreviews.org/doi/10.1146/annurev-matsci-081420-041617

[41] W. K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika 57 (1) (1970) 97–109. doi:10.1093/biomet/57.1.97.
URL https://academic.oup.com/biomet/article/57/1/97/284580

[42] A. Magna, S. Coffa, L. Colombo, A lattice kinetic Monte Carlo code for the description of vacancy diffusion and self-organization in Si, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 148 (1-4) (1999) 262–267. doi:10.1016/S0168-583X(98)00798-8.
URL https://linkinghub.elsevier.com/retrieve/pii/S0168583X98007988

[43] D. J. Dooling, L. J. Broadbelt, Generic Monte Carlo Tool for Kinetic Modeling, Industrial & Engineering Chemistry Research 40 (2) (2001) 522–529. doi:10.1021/ie000310q.
URL https://pubs.acs.org/doi/10.1021/ie000310q

[44] S. X. M. Boerrigter, G. P. H. Josten, J. van de Streek, F. F. A. Hollander, J. Los, H. M. Cuppen, P. Bennema, H. Meekes, MONTY: Monte Carlo Crystal Growth on Any Crystal Structure in Any Crystallographic Orientation; Application to Fats, The Journal of Physical Chemistry A 108 (27) (2004) 5894–5902. doi:10.1021/jp049804h.
URL https://pubs.acs.org/doi/10.1021/jp049804h

[45] M. Leetmaa, N. V. Skorodumova, KMCLib: A general framework for lattice kinetic Monte Carlo (KMC) simulations, Computer Physics Communications 185 (9) (2014) 2340–2349. doi:10.1016/j.cpc.2014.04.017.
URL https://linkinghub.elsevier.com/retrieve/pii/S0010465514001519

[46] M. J. Hoffmann, S. Matera, K. Reuter, Kmos: A lattice kinetic Monte Carlo framework, Computer Physics Communications 185 (7) (2014) 2138–2150, arXiv: 1401.5278 Publisher: Elsevier B.V. doi:10.1016/j.cpc.2014.04.003.
URL http://dx.doi.org/10.1016/j.cpc.2014.04.003

[47] J. J. Ramsey, KMCThinFilm: A C++ Framework for the Rapid Development of Lattice Kinetic Monte Carlo (kMC) Simulations of Thin Film Growth, Tech. rep., US Army Research Laboratory (2015).

[48] I. Mitchell, S. Irle, A. J. Page, A global reaction route mapping-based kinetic Monte Carlo algorithm, The Journal of Chemical Physics 145 (2) (2016) 024105. doi:10.1063/1.4954660.
URL http://aip.scitation.org/doi/10.1063/1.4954660

[49] T. Danielson, J. E. Sutton, C. Hin, A. Savara, SQERTSS: Dynamic rank based throttling of transition probabilities in kinetic Monte Carlo simulations, Computer Physics Communications 219 (2017) 149–163. doi:10.1016/j.cpc.2017.05.016.
URL https://linkinghub.elsevier.com/retrieve/pii/S0010465517301583

[50] M. Jørgensen, H. Grönbeck, MonteCoffee: A programmable kinetic Monte Carlo framework, The Journal of Chemical Physics 149 (11) (2018) 114101. doi:10.1063/1.5046635.
URL http://aip.scitation.org/doi/10.1063/1.5046635

[51] J. Li, P. Wei, S. Yang, J. Wu, P. Liu, X. He, Crystal-KMC: parallel software for lattice dynamics monte carlo simulation of metal materials, Tsinghua Science and Technology 23 (4) (2018) 501–510. doi:10.26599/TST.2018.9010107.
URL https://ieeexplore.ieee.org/document/8421558/

[52] K. Li, H. Shang, Y. Zhang, S. Li, B. Wu, D. Wang, L. Zhang, F. Li, D. Chen, Z. Wei, OpenKMC: a KMC design for hundred-billion-atom simulation using millions of cores on Sunway Taihulight, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, Denver Colorado, 2019, pp. 1–16. doi:10.1145/3295500.3356165.
URL https://dl.acm.org/doi/10.1145/3295500.3356165

[53] P. Martin, J. J. Gaitero, J. S. Dolado, H. Manzano, KIMERA: A Kinetic Montecarlo Code for Mineral Dissolution, Minerals 10 (9) (2020) 825. doi:10.3390/min10090825.
URL https://www.mdpi.com/2075-163X/10/9/825

[54] T. P. Schulze, Kinetic Monte Carlo simulations with minimal searching, Physical Review E 65 (3) (2002) 036704. doi:10.1103/PhysRevE.65.036704.
URL https://link.aps.org/doi/10.1103/PhysRevE.65.036704

[55] K. Bernacki, B. Hetényi, B. J. Berne, Multiple "time step" Monte Carlo simulations: Application to charged systems with Ewald summation, The Journal of Chemical Physics 121 (1) (2004) 44. doi:10.1063/1.1755195.
URL http://scitation.aip.org/content/aip/journal/jcp/121/1/10.1063/1.1755195

[56] F. Shi, Y. Shim, J. G. Amar, Parallel kinetic Monte Carlo simulations of two-dimensional island coarsening, Physical Review E 76 (3) (2007) 031607. doi:10.1103/PhysRevE.76.031607.
URL https://link.aps.org/doi/10.1103/PhysRevE.76.031607

[57] L. Xu, G. Henkelman, Adaptive kinetic Monte Carlo for first-principles accelerated dynamics, The Journal of Chemical Physics 129 (11) (2008) 114104. doi:10.1063/1.2976010.
URL http://aip.scitation.org/doi/10.1063/1.2976010

[58] A. Slepoy, A. P. Thompson, S. J. Plimpton, A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks, The Journal of Chemical Physics 128 (20) (2008) 205101. doi:10.1063/1.2919546.
URL http://aip.scitation.org/doi/10.1063/1.2919546

[59] A. Chatterjee, A. F. Voter, Accurate acceleration of kinetic Monte Carlo simulations through the modification of rate constants, The Journal of Chemical Physics 132 (19) (2010) 194101. doi:10.1063/1.3409606.
URL http://aip.scitation.org/doi/10.1063/1.3409606

[60] J. Nielsen, M. d'Avezac, J. Hetherington, M. Stamatakis, Parallel kinetic Monte Carlo simulation framework incorporating accurate models of adsorbate lateral interactions, The Journal of Chemical Physics 139 (22) (2013) 224706. doi:10.1063/1.4840395.
URL http://aip.scitation.org/doi/10.1063/1.4840395

[61] H. Xu, Y. N. Osetsky, R. E. Stoller, Simulating complex atomistic processes: On-the-fly kinetic Monte Carlo scheme with selective active volumes, Physical Review B 84 (13) (2011) 132103. doi:10.1103/PhysRevB.84.132103.
URL https://link.aps.org/doi/10.1103/PhysRevB.84.132103

[62] D. Konwar, V. J. Bhute, A. Chatterjee, An off-lattice, self-learning kinetic Monte Carlo method using local environments, The Journal of Chemical Physics 135 (17) (2011) 174103. doi:10.1063/1.3657834.
URL http://aip.scitation.org/doi/10.1063/1.3657834

[63] M. Stamatakis, D. G. Vlachos, A graph-theoretical kinetic Monte Carlo framework for on-lattice chemical kinetics, The Journal of Chemical Physics 134 (21) (2011) 214115. doi:10.1063/1.3596751.
URL http://aip.scitation.org/doi/10.1063/1.3596751

[64] X. Guo, D. Minakata, J. Crittenden, On-the-Fly Kinetic Monte Carlo Simulation of Aqueous Phase Advanced Oxidation Processes, Environmental Science & Technology 49 (15) (2015) 9230–9236. doi:10.1021/acs.est.5b02034.
URL https://pubs.acs.org/doi/10.1021/acs.est.5b02034

[65] Q. Yang, C. A. Sing-Long, E. J. Reed, Learning reduced kinetic Monte Carlo models of complex chemistry from molecular dynamics, Chemical Science 8 (8) (2017) 5781–5796. doi:10.1039/C7SC01052D.
URL http://xlink.rsc.org/?DOI=C7SC01052D

[66] A. Chatterjee, D. G. Vlachos, Multiscale spatial Monte Carlo simulations: Multigriding, computational singular perturbation, and hierarchical stochastic closures, The Journal of Chemical Physics 124 (6) (2006) 064110. doi:10.1063/1.2166380.
URL http://aip.scitation.org/doi/10.1063/1.2166380

[67] S. D. Collins, A. Chatterjee, D. G. Vlachos, Coarse-grained kinetic Monte Carlo models: Complex lattices, multicomponent systems, and homogenization at the stochastic level, The Journal of Chemical Physics 129 (18) (2008) 184101. doi:10.1063/1.3005225.
URL http://aip.scitation.org/doi/10.1063/1.3005225

[68] Z. Deng, V. Kumar, F. T. Bölle, F. Caro, A. A. Franco, I. E. Castelli, P. Canepa, Z. W. Seh, Towards autonomous high-throughput multiscale modelling of battery interfaces, Energy & Environmental Science 15 (2) (2022) 579–594. doi:10.1039/D1EE02324A.
URL http://xlink.rsc.org/?DOI=D1EE02324A

[69] F. Pérez, B. E. Granger, J. D. Hunter, Python: An Ecosystem for Scientific Computing, Computing in Science & Engineering 13 (2) (2011) 13–21, conference Name: Computing in Science & Engineering. doi:10.1109/MCSE.2010.119.

[70] Top Programming Languages 2022, section: Computing (Aug. 2022).
URL https://spectrum.ieee.org/top-programming-languages-2022

[71] S. K. Lam, A. Pitrou, S. Seibert, Numba: a LLVM-based Python JIT compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM '15, ACM Press, Austin, Texas, 2015, pp. 1–6. doi:10.1145/2833157.2833162.
URL http://dl.acm.org/citation.cfm?doid=2833157.2833162

[72] A. Fick, V. on liquid diffusion, Lond. Edinb. Dublin Philos. Mag. J. Sci. 10 (63) (1855) 30–39. doi:10.1080/14786445508641925.
URL https://www.tandfonline.com/doi/full/10.1080/14786445508641925

[73] A. Fick, Ueber Diffusion, Annalen der Physik und Chemie 170 (1) (1855) 59–86. doi:10.1002/andp.18551700105.
URL https://onlinelibrary.wiley.com/doi/10.1002/andp.18551700105

[74] G. Murch, The haven ratio in fast ionic conductors, Solid State Ionics 7 (3) (1982) 177–198. doi:10.1016/0167-2738(82)90050-9.
URL https://linkinghub.elsevier.com/retrieve/pii/0167273882900509

[75] H. Jónsson, G. Mills, K. W. Jacobsen, Nudged elastic band method for finding minimum energy paths of transitions, in: Classical and Quantum Dynamics in Condensed Phase Simulations, WORLD SCIENTIFIC, LERICI, Villa Marigola, 1998, pp. 385–404. doi:10.1142/9789812839664_0016.
URL http://www.worldscientific.com/doi/abs/10.1142/9789812839664_0016

[76] G. Henkelman, B. P. Uberuaga, H. Jónsson, A climbing image nudged elastic band method for finding saddle points and minimum energy paths, The Journal of Chemical Physics 113 (22) (2000) 9901–9904. doi:10.1063/1.1329672.
URL http://aip.scitation.org/doi/10.1063/1.1329672

[77] A. Van der Ven, J. Thomas, B. Puchala, A. Natarajan, First-Principles Statistical Mechanics of Multicomponent Crystals, Annual Review of Materials Research 48 (1) (2018) 27–55. doi:10.1146/annurev-matsci-070317-124443.
URL https://www.annualreviews.org/doi/10.1146/annurev-matsci-070317-124443

[78] P. Xiao, T. Shi, W. Huang, G. Ceder, Understanding Surface Densified Phases in Ni-Rich Layered Compounds, ACS Energy Letters 4 (4) (2019) 811–818. doi:10.1021/acsenergylett.9b00122.
URL https://pubs.acs.org/doi/10.1021/acsenergylett.9b00122

[79] G. H. Vineyard, Frequency factors and isotope effects in solid state rate processes, Journal of Physics and Chemistry of Solids 3 (1-2) (1957) 121–127. doi:10.1016/0022-3697(57)90059-8.
URL https://linkinghub.elsevier.com/retrieve/pii/0022369757900598

[80] E. Kaxiras, J. Erlebacher, Adatom diffusion by orchestrated exchange on semiconductor surfaces, Physical Review Letters 72 (11) (1994) 1714–1717. doi:10.1103/PhysRevLett.72.1714.
URL https://link.aps.org/doi/10.1103/PhysRevLett.72.1714

[81] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[82] F. Santosa, W. W. Symes, Linear Inversion of Band-Limited Reflection Seismograms, SIAM Journal on Scientific and Statistical Computing 7 (4) (1986) 1307–1330. doi:10.1137/0907087.
URL http://epubs.siam.org/doi/10.1137/0907087

[83] J. C. Thomas, P. Brian, CASM: A Clusters Approach to Statistical Mechanics (Sep. 2022).
URL https://github.com/prisms-center/CASMcode

[84] C. Kiel, Gooey (2022).
URL https://github.com/chriskiehl/Gooey

[85] kMCpy Documentation (Aug. 2022).
URL https://kmcpy.readthedocs.io/en/latest/