lattice_mc Documentation

Release 1.0.3

Benjamin J. Morgan

Aug 30, 2018

Contents:

CHAPTER 1

lattice_mc: A Python Lattice-Gas Monte Carlo Module

PyPI version Build Status DOI JOSS status Documentation Status

`lattice_mc` is Python module for performing (kinetic) lattice-gas Monte Carlo (LGMC) simulations of ionic transport in solid electrolytes.

In solid electrolytes, ionic motion is typically effected by a series of discrete "jumps" where ions move between adjacent lattice sites [*1*]. For dilute mobile ions, ionic trajectories are random walks, and simple analytical expressions exits that relate macroscopic transport coefficients, i.e. diffusion coefficients and ionic conductivities, to the microscopic jump frequency of individual ions [*2*,*3*]. Practical solid electrolytes have high mobile ion concentrations, with significant interparticle interactions producing deviations from the dilute limit random walk behaviour. In general, the *quantitative* effect of interparticle interactions cannot be determined analytically. As an alternative, numerical simulations, such as lattice-gas Monte Carlo methods, can be used to directly calculate these relationships. Lattice-gas Monte Carlo methods are particularly suited to studying how varying properties across broad classes of materials give quantitative differences in macroscopic ionic transport, and can be used to understand the different transport properties of materials with, for example, different crystal structures or mobile ion stoichiometries.

`lattice_mc` has been written to allow materials scientists and solid-state chemists model how the microscopic physics of solid electrolytes (crystal structure, stoichiometry, interaction models) determine macroscopic transport behaviour (diffusion and ionic conduction), with the goal of understand the factors that make different materials more or less useful for specific applications (e.g. Li-ion batteries or fuel cells).

The code allows the programmatic construction of simple lattices (presently implemented are square, honeycomb, and cubic lattices). Lattices with arbitrary geometries can be constructed from a file format that defines the lattice sites and their connectivity, allowing models based on crystallographic data. The algorithms used and interaction models are described in more detail in Ref. [*4*]. Calculated properties include tracer and "jump" diffusion coefficients; where the latter is proportional to the mobility (and hence the conductivity for charged particles) [*5*]; and tracer (single particle) and collective correlation factors, f and fI [*6*]. The simplest interaction model is for "non-interacting" particles, where the only restriction is volume exclusion (two particles cannot simultaneously occupy a single site) [*7*]. Additional interaction models include nearest-neighbour repulsion and on-site energies for inequivalent sites. Simulations are performed using an efficient rejection-free Monte Carlo scheme [*8*].

1.1 Installation

```
pip install lattice_mc
```

Or download the latest release from GitHub

```
https://github.com/bjmorgan/lattice_mc/archive/1.0.0.tar.gz
```

Then install

```
cd lattice_mc
python setup.py install
```

Or you can clone the latest development version:

```
git clone git@github.com:bjmorgan/lattice_mc.git
```

and install the same way.

```
cd lattice_mc
python setup.py install
```

Alternatively, you can install the latest build using `pip`, direct from GitHub, e.g.

```
pip3 install git+https://github.com/bjmorgan/lattice_mc.git
```

1.2 Documentation

Full documentation and examples are contained in a Jupyter notebook at examples/lattice_mc_example.ipynb. The example notebook is also hosted on GitHub.

1.3 Tests

Automated testing of the latest build happens here.

Manual tests can be run using

```
python3 -m unittest discover
```

The code has been tested with Python versions 3.5 and above.

1.4 References

1. C. R. A. Catlow, Sol. Stat. Ionics 8, 89 (1983).
2. R. E. Howard and A. B. Lidiard, Rep. Prog. Phys. 27, 161 (1964).
3. J. H. Harding, Defects and Transport in Ionic Solids, in Ionic Solids at High Temperatures ed. A. M. Stoneham, World Scientific (1989).
4. B. J. Morgan, arXiv: 1707.00491

5. A. Van der Ven et al. Acc. Chem. Res. 46, 1216 (2013).
6. G. E. Murch Sol. Stat. Ionics 7, 177 (1982).
7. R. Kutner Phys. Lett. 81A, 239 (1981).
8. A. F. Voter, Introduction to the Kinetic Monte Carlo Method, in Radiation Effects in Solids, ed. K. E. Sicafus et al., Springer (2007).
9. Morgan and Madden, J. Phys. Condens. Matter 24, 275303 (2012).
10. G. E. Murch & R. J. Thorn, Phil. Mag. 36 529 (1977).

CHAPTER 2

Blocked Lattices

It is possible to construct a simulation lattice where there are no possible jumps; such a lattice is classified as "blocked". To test for this, a `Lattice` object can be queried using the `is_blocked()` method:

```
>>> simulation.lattice.is_blocked()
True
```

If a simulation with a blocked lattice is run, a `BlockedLatticeError` exception is raised:

```
Traceback (most recent call last):
  ...
  self.lattice.jump()
File "/Users/bjm42/source/lattice_mc/lattice_mc/lattice.py", line 228, in jump
  raise BlockedLatticeError('No moves are possible in this lattice')
lattice_mc.error.BlockedLatticeError: No moves are possible in this lattice
```

CHAPTER 3

lattice_mc

3.1 lattice_mc package

3.1.1 Submodules

3.1.2 lattice_mc.atom module

class lattice_mc.atom.**Atom**(*initial_site*)

Bases: object

Atoms are distinguishable particles, each occupying a specific lattice site.

atom_number = 0

dr_squared()

$|dr|^2$, where dr is the total displacement vector for this *Atom*.

Parameters **None** –

Returns $|dr|^2$.

Return type dr_squared (float)

reset()

Reinitialise the stored displacements, number of hops, and list of sites visited for this *Atom*.

Parameters **None** –

Returns None

site

Get or set *self.site* for this *Atom*.

3.1.3 lattice_mc.cluster module

class lattice_mc.cluster.**Cluster**(*sites*)
Bases: object

Clusters are sets of sites.

is_neighbouring(*other_cluster*)
logical check whether the neighbour list for cluster A includes any sites in cluster B

Parameters **other_cluster** (Cluster) – the other cluster we are testing for neighbour connectivity

Returns True if the other cluster neighbours this cluster.

Return type (Bool)

is_periodically_contiguous()
logical check whether a cluster connects with itself across the simulation periodic boundary conditions.

Parameters **none** –

Returns (Bool, Bool, Bool): Contiguity along the x, y, and z coordinate axes

merge(*other_cluster*)
Combine two clusters into a single cluster.

Parameters **other_cluster** (Cluster) – The second cluster to combine.

Returns The combination of both clusters.

Return type (*Cluster*)

remove_sites_from_neighbours(*remove_labels*)
Removes sites from the set of neighbouring sites if these have labels in remove_labels.

Parameters **Remove_labels** (*List) or (Str*) – List of Site labels to be removed from the cluster neighbour set.

Returns None

sites_at_edges()
Finds the six sites with the maximum and minimum coordinates along x, y, and z.

Parameters **None** –

Returns In the order [+x, -x, +y, -y, +z, -z]

Return type (List(List))

size()
Number of sites in this cluster.

Parameters **None** –

Returns The number of sites in this cluster.

Return type (Int)

3.1.4 lattice_mc.global_vars module

3.1.5 lattice_mc.init_lattice module

lattice_mc.init_lattice.**cubic_lattice**(*a*, *b*, *c*, *spacing*)
Generate a cubic lattice.

Parameters

- **a** (*Int*) – Number of lattice repeat units along x.
- **b** (*Int*) – Number of lattice repeat units along y.
- **c** (*Int*) – Number of lattice repeat units along z.
- **spacing** (*Float*) – Distance between lattice sites.

Returns The new lattice

Return type (*Lattice*)

lattice_mc.init_lattice.**honeycomb_lattice**(*a*, *b*, *spacing*, *alternating_sites=False*)
Generate a honeycomb lattice.

Parameters

- **a** (*Int*) – Number of lattice repeat units along x.
- **b** (*Int*) – Number of lattice repeat units along y.
- **spacing** (*Float*) – Distance between lattice sites.
- **alternating_sites** (*Bool, optional*) – Label alternating sites with 'A' and 'B'. Defaults to False.

Returns The new lattice

Return type (*Lattice*)

Notes

The returned lattice is 3D periodic, but all sites and edges lie in the xy plane.

lattice_mc.init_lattice.**lattice_from_sites_file**(*site_file*, *cell_lengths*)
Generate a lattice from a sites file.

Parameters

- **site_file** (*Str*) – Filename for the file containing the site information.
- **cell_lengths** (*List(Float,Float,Float)*) – A list containing the [x, y, z] cell lengths.

Returns The new lattice

Return type (*Lattice*)

Notes

The site information file format is:

<number_of_sites> (Int).

Followed by blocks of data separated by double linebreaks; one block per site.

site: <site number> (Int).
centre: <x> <y> <z> (Float,Float,Float).
neighbours: <list of site numbers of neighbouring sites> (List[Int]).
label: <site group labal> (Str).
energy: <site occupation energy> (Float).

The energy is optional, and will be set to 0.0 if not included.
Line order within each block is not meaningful.
British and American English spellings for centre|center and neighbour|neighbor are accepted.
An example file can be found in the examples directory.

lattice_mc.init_lattice.**square_lattice**(*a*, *b*, *spacing*)

Generate a square lattice.

Parameters

- **a** (*Int*) – Number of lattice repeat units along x.
- **b** (*Int*) – Number of lattice repeat units along y.
- **spacing** (*Float*) – Distance between lattice sites.

Returns The new lattice

Return type (*Lattice*)

Notes

The returned lattice is 3D periodic, but all sites and edges lie in the xy plane.

3.1.6 lattice_mc.jump module

class lattice_mc.jump.**Jump**(*initial_site*, *final_site*, *nearest_neighbour_energy=False*, *coordination_number_energy=False*, *jump_lookup_table=None*)

Bases: object

boltzmann_factor()

Boltzmann probability factor for accepting this jump, following the Metropolis algorithm.

Parameters **None** –

Returns Metropolis relative probability of accepting this jump.

Return type (Float)

coordination_number_delta_E()

Coordination-number dependent energy conrtibution to the change in system energy if this jump were accepted.

Parameters **None** –

Returns delta E (coordination-number)

Return type (Float)

delta_E()

The change in system energy if this jump were accepted.

Parameters **None** –

Returns delta E

Return type (Float)

dr(*cell_lengths*)
Particle displacement vector for this jump

Parameters **cell_lengths** (*np.array(x,y,z)*) – Cell lengths for the orthogonal simulation cell.

Returns (np.array(x,y,z)): dr

nearest_neighbour_delta_E()
Nearest-neighbour interaction contribution to the change in system energy if this jump were accepted.

Parameters **None** –

Returns delta E (nearest-neighbour)

Return type (Float)

rate()
Average rate for this jump. Calculated as (v_0 * P_jump).

Parameters **None** –

Returns The average rate for this jump.

Return type (Float)

relative_probability

relative_probability_from_lookup_table(*jump_lookup_table*)
Relative probability of accepting this jump from a lookup-table.

Parameters **jump_lookup_table** (LookupTable) – the lookup table to be used for this jump.

Returns relative probability of accepting this jump.

Return type (Float)

3.1.7 lattice_mc.lattice module

class lattice_mc.lattice.**Lattice**(*sites*, *cell_lengths*)
Bases: object

Lattice class

connected_site_pairs()
Returns a dictionary of all connections between pair of sites (by site label). e.g. for a linear lattice A-B-C will return:

```
{ 'A' : [ 'B' ], 'B' : [ 'A', 'C' ], 'C' : [ 'B' ] }
```

Parameters **None** –

Returns A dictionary of neighbouring site types in the lattice.

Return type site_connections (Dict{Str List[Str]})

connected_sites(*site_labels=None*)
Searches the lattice to find sets of sites that are contiguously neighbouring. Mutually exclusive sets of contiguous sites are returned as Cluster objects.

Parameters **(** (*site_labels*) – obj:(List(Str)|Set(Str)|Str), optional): Labels for sites to be considered in the search. This can be a list:

```
[ 'A', 'B' ]
```

a set:

```
( 'A', 'B' )
```

or a string:

```
'A'.
```

Returns List of Cluster objects for groups of contiguous sites.

Return type (List(*Cluster*))

detached_sites(*site_labels=None*)
Returns all sites in the lattice (optionally from the set of sites with specific labels) that are not part of a percolating network. This is determined from clusters of connected sites that do not wrap round to themselves through a periodic boundary.

Parameters **site_labels** (*String or List(String)*) – Lables of sites to be considered.

Returns List of sites not in a periodic percolating network.

Return type (List(*Site*))

enforce_periodic_boundary_conditions()
Ensure that all lattice sites are within the central periodic image of the simulation cell. Sites that are outside the central simulation cell are mapped back into this cell.

Parameters **None** –

Returns None

initialise_site_lookup_table()
Create a lookup table allowing sites in this lattice to be queried using *self.site_lookup[n]* where *n* is the identifying site numbe.

Parameters **None** –

Returns None

is_blocked()
Check whether there are any possible jumps.

Parameters **None** –

Returns True if there are no possible jumps. Otherwise returns False.

Return type (Bool)

jump()
Select a jump at random from all potential jumps, then update the lattice state.

Parameters **None** –

Returns None

max_site_coordination_numbers()
Returns a dictionary of the maximum coordination number for each site label. e.g.:

```
{ 'A' : 4, 'B' : 4 }
```

Parameters **none** –

Returns

Int)): dictionary of maxmimum coordination number for each site label.

Return type max_coordination_numbers (Dict(Str

occupied_site_numbers()
List of site id numbers for all sites that are occupied.

Parameters **None** –

Returns List of site id numbers for occupied sites.

Return type List(Int)

occupied_sites()
The set of sites occupied by atoms.

Parameters **None** –

Returns List of sites that are occupied.

Return type List(*Site*)

populate_sites(*number_of_atoms*, *selected_sites=None*)
Populate the lattice sites with a specific number of atoms.

Parameters

- **number_of_atoms** (*Int*) – The number of atoms to populate the lattice sites with.
- **(** (*selected_sites*) – obj:List, optional): List of site labels if only some sites are to be occupied. Defaults to None.

Returns None

potential_jumps()
All nearest-neighbour jumps not blocked by volume exclusion (i.e. from occupied to neighbouring unoccupied sites).

Parameters **None** –

Returns List of possible jumps.

Return type (List(*Jump*))

reset()
Reset all time-dependent counters for this lattice and its constituent sites

Parameters **None** –

Returns None

select_sites(*site_labels*)
Selects sites in the lattice with specified labels.

Parameters **site_labels** (*List(Str)|Set(Str)|Str*) – Labels of sites to select. This can be a List ['A', 'B'], a Set ('A', 'B'), or a String 'A'.

Returns List of sites with labels given by *site_labels*.

Return type (List(*Site*))

set_cn_energies(*cn_energies*)
Set the coordination number dependent energies for this lattice.

Parameters **(Dict(Str** (*cn_energies*) – Dict(Int:Float))): Dictionary of dictionaries specifying the coordination number dependent energies for each site type. e.g.:

```
{ 'A' : { 0 : 0.0, 1 : 1.0, 2 : 2.0 }, 'B' : { 0 : 0.0, 1 : 2.0 } }
```

Returns None

set_nn_energy(*delta_E*)
Set the lattice nearest-neighbour energy.

Parameters **delta_E** (*Float*) – The nearest-neighbour energy E_nn.

Returns None

set_site_energies(*energies*)
Set the energies for every site in the lattice according to the site labels.

Parameters **(Dict(Str** (*energies*) – Float): Dictionary of energies for each site label, e.g.:

```
{ 'A' : 1.0, 'B', 0.0 }
```

Returns None

site_coordination_numbers()
Returns a dictionary of the coordination numbers for each site label. e.g.:

```
{ 'A' : { 4 }, 'B' : { 2, 4 } }
```

Parameters **none** –

Returns

Set(Int))): dictionary of coordination numbers for each site label.

Return type coordination_numbers (Dict(Str

site_occupation_statistics()
Average site occupation for each site type

Parameters **None** –

Returns

Float)): Dictionary of occupation statistics, e.g.:

```
{ 'A' : 2.5, 'B' : 25.3 }
```

Return type (Dict(Str

site_specific_coordination_numbers()
Returns a dictionary of coordination numbers for each site type.

Parameters **None** –

Returns

List(Int))) : Dictionary of coordination numbers for each site type, e.g.:

```
{ 'A' : [ 2, 4 ], 'B' : [ 2 ] }
```

Return type (Dict(Str

site_with_id(*number*)
Select the site with a specific id number.

Parameters **number** (*Int*) – The identifying number for a specific site.

Returns The site with id number equal to *number*

Return type (*Site*)

transmute_sites(*old_site_label*, *new_site_label*, *n_sites_to_change*)
Selects a random subset of sites with a specific label and gives them a different label.

Parameters

- **old_site_label** (*String or List(String)*) – Site label(s) of the sites to be modified..
- **new_site_label** (*String*) – Site label to be applied to the modified sites.
- **n_sites_to_change** (*Int*) – Number of sites to modify.

Returns None

update(*jump*)
Update the lattice state by accepting a specific jump

Parameters **jump** (Jump) – The jump that has been accepted.

Returns None.

update_site_occupation_times(*delta_t*)
Increase the time occupied for all occupied sites by delta t

Parameters **delta_t** (*Float*) – Timestep.

Returns None

vacant_site_numbers()
List of site id numbers for all sites that are vacant.

Parameters **None** –

Returns List of site id numbers for vacant sites.

Return type List(Int)

vacant_sites()
The set of sites not occupied by atoms.

Parameters **None** –

Returns List of sites that are vacant.

Return type List(*Site*)

3.1.8 lattice_mc.lattice_site module

class lattice_mc.lattice_site.**Site**(*number*, *coordinates*, *neighbours*, *energy*, *label*, *cn_energies=None*)
Bases: object

Site class

cn_occupation_energy(*delta_occupation=None*)
The coordination-number dependent energy for this site.

Parameters **(** (*delta_occupation*) – obj:Dict(Str:Int), optional): A dictionary of a change in (site-type specific) coordination number, e.g. { 'A' : 1, 'B' : -1 }. If this is not None, the coordination-number dependent energy is calculated including these changes in neighbour-site occupations. Defaults to None

Returns The coordination-number dependent energy for this site.

Return type (Float)

index = 0

nn_occupation()
The number of occupied nearest-neighbour sites.

Parameters **None** –

Returns The number of occupied nearest-neighbour sites.

Return type (Int)

set_cn_occupation_energies(*cn_energies*)
Set the coordination-number dependent energies for this site.

Parameters **(Dict(Int** (*cn_energies*) – Float)): Dictionary of coordination number dependent site energies, e.g. { 0 : 0.0, 1 : 0.5 }.

Returns None

site_specific_neighbours()
Returns the number of neighbouring sites, classified by site type.

Parameters **None** –

Returns Int)): Dictionary of neighboring sites, classified by site label, e.g. { 'A' : 1, 'B' : 1 }.

Return type (Dict(Str

site_specific_nn_occupation()
Returns the number of occupied nearest neighbour sites, classified by site type.

Parameters **None** –

Returns Int)): Dictionary of nearest-neighbour occupied site numbers, classified by site label, e.g. { 'A' : 2, 'B' : 1 }.

Return type (Dict(Str

3.1.9 lattice_mc.lookup_table module

class lattice_mc.lookup_table.**LookupTable**(*lattice*, *hamiltonian*)
Bases: object

LookupTable class

generate_nearest_neighbour_lookup_table()
Construct a look-up table of relative jump probabilities for a nearest-neighbour interaction Hamiltonian.

Parameters **None.** –

Returns None.

relative_probability(*l1*, *l2*, *c1*, *c2*)
The relative probability for a jump between two sites with specific site types and coordination numbers.

Parameters

- **l1** (*Str*) – Site label for the initial site.
- **l2** (*Str*) – Site label for the final site.
- **c1** (*Int*) – Coordination number for the initial site.
- **c2** (*Int*) – Coordination number for the final site.

Returns The relative probability of this jump occurring.

Return type (Float)

lattice_mc.lookup_table.**metropolis**(*delta_E*)
Boltzmann probability factor for an event with an energy change *delta_E*, following the Metropolis algorithm.

Parameters **delta_E** (*Float*) – The change in energy.

Returns Metropolis relative probability for this event.

Return type (Float)

3.1.10 lattice_mc.options module

class lattice_mc.options.**Options**
Bases: object

Object for storing options for setting up and running a simulation.

read_lattice_from_file(*filename*)
Set the filename for the sites file used to define the simulation lattice.

Parameters **filename** (*Str*) – The filename for the sites file.

Returns None

set_cn_energies(*cn_energies*)
Set the coordination-number dependent energies.

Parameters **(Dict(Str** (*cn_energies*) – Dict(Int:Float))): Dict of Dicts specifying the coordination-number dependent energies for each site type. e.g. { 'A' : { 0 : 0.0, 1 : 1.0 }, 'B' : { 0 : 0.0, 1 : 2.0 } }

Returns None

set_cn_energy_scaling(*cn_energy_scaling*)
Set a scaling factor for the coordination-number dependent energies.

Parameters **cn_energy_scaling** (*Float*) – Coordination-number dependent energy scaling.

Returns None

set_lattice_cell_lengths(*cell_lengths*)
Set the lattice cell lengths for a simulation.

Parameters **cell_lengths** (*np.array(x,y,z)*) – Lattice cell lengths.

Returns None

set_nn_energy_scaling(*energy_scaling*)
Set a scaling factor for the nearest-neighbour interaction energy.

Parameters **energy_scaling** (*Float*) – Nearest-neighbour energy scaling.

Returns None

set_number_of_atoms(*number_of_atoms*)
Set the number of atoms present in the simulation.

Parameters **number_of_atoms** (*int*) – The number of atoms.

Returns None

set_number_of_equilibration_jumps(*number_of_equilibration_jumps*)
Set the number of equilibration jumps for a simulation.

Parameters **number_of_equilibration_jumps** (*Int*) – The number of equiibration jumps.

Returns None

set_number_of_jumps(*number_of_jumps*)
Set the on-site energies for each site type.

Parameters **(Dict(Str** (*site_energies*) – Float)): On-site energies for each site type label. e.g. { 'A' : 1.0, 'B', -1.0 }

Returns None

set_site_energies(*site_energies*)
Set the on-site energies for each site type.

Parameters **(Dict(Str** (*site_energies*) – Float)): On-site energies for each site type label. e.g. { 'A' : 1.0, 'B', -1.0 }

Returns None

3.1.11 lattice_mc.simulation module

class lattice_mc.simulation.**Simulation**
Bases: object

Simulation class

average_site_occupations
Average site occupation numbers.

Parameters **None** –

Returns

Float)): Average site occupation numbers for each site label, e.g. { 'A' : 12.4, 'B' : 231.2 }

Return type (Dict(Str

collective_correlation
Returns the collective correlation factor, f_I

Parameters **None** –

Returns The collective correlation factor, f_I.

Return type (Float)

collective_diffusion_coefficient
Returns the collective or "jump" diffusion coefficient, D_J.

Parameters **None** –

Returns The collective diffusion coefficient, D_J.

Return type (Float)

collective_diffusion_coefficient_per_atom
The collective diffusion coefficient per atom. D_J / n_atoms.

Parameters **None** –

Returns The collective diffusion coefficient per atom, D_J / n_atoms.

Return type (Float)

define_lattice_from_file(*filename*, *cell_lengths*)

Set up the simulation lattice from a file containing site data. Uses *init_lattice.lattice_from_sites_file*, which defines the site file spec.

Parameters

- **filename** (*Str*) – sites file filename.
- **cell_lengths** (*List(x,y,z)*) – cell lengths for the simulation cell.

Returns None

is_initialised()
Check whether the simulation has been initialised.

Parameters **None** –

Returns None

old_collective_correlation
Returns the collective correlation factor, f_I

Parameters **None** –

Returns The collective correlation factor, f_I.

Return type (Float)

Notes

This function assumes that the jump distance between sites has been normalised to a=1. If the jumps distance is not equal to 1 then the value returned by this function should be divided by a^2. Even better, use self.collective_correlation

old_tracer_correlation
Deprecated tracer correlation factor for this simulation.

Parameters **None** –

Returns The tracer correlation factor, f.

Return type (Float)

Notes

This function assumes that the jump distance between sites has been normalised to a=1. If the jump distance is not equal to 1 then the value returned by this function should be divided by a^2. Even better, use *self.tracer_correlation*.

reset()
Reset all counters for this simulation.

Parameters **None** –

Returns None

run(*for_time=None*)
Run the simulation.

Parameters **(** (*for_time*) – obj:Float, optional): If *for_time* is set, then run the simulation until a set amount of time has passed. Otherwise, run the simulation for a set number of jumps. Defaults to None.

Returns None

set_cn_energies(*cn_energies*)
Set the coordination-number dependent energies for this simulation.

Parameters **(Dict(Str** (*cn_energies*) – Dict(Int:Float))): Dict of Dicts specficying the coordination-number dependent energies. e.g. { 'A' : { 0 : 0.0, 1: 0.5 }, 'B' : { 0 : -0.5, 1 : -2.0 } }

Returns None

set_nn_energy(*nn_energy*)
Set the nearest-neighbour energy for this simulation.

Parameters **nn_energy** (*Float*) – nearest-neigbour energy.

Returns None

set_number_of_atoms(*n*, *selected_sites=None*)
Set the number of atoms for the simulation, and populate the simulation lattice.

Parameters

- **n** (*Int*) – Number of atoms for this simulation.
- **(** (*selected_sites*) – obj:(List|Set|String), optional): Selects a subset of site types to be populated with atoms. Defaults to None.

Returns None

set_number_of_equilibration_jumps(*n*)
Set the number of equilibration jumps for this simulation.

Parameters **n** (*Int*) – number of equilibration jumps

Returns None

set_number_of_jumps(*n*)
Set the number of jumps for this simulation.

Parameters **n** (*Int*) – number of jumps

Returns None

set_site_energies(*site_energies*)
Set the on-site energies for this simulation.

Parameters **(Dict(Str** (*site_energies*) – Float)): Dict of energies for each site type. e.g. { 'A' : 0.0,. 'B' : 1.0 }

Returns None

setup_lookup_table(*hamiltonian='nearest-neighbour'*)
Create a jump-probability look-up table corresponding to the appropriate Hamiltonian.

Parameters **hamiltonian** (*Str, optional*) – String specifying the simulation Hamiltonian. valid values are 'nearest-neighbour' (default) and 'coordination_number'.

Returns None

tracer_correlation
Tracer correlation factor, f.

Parameters **None** –

Returns The tracer correlation factor, f.

Return type (Float)

tracer_diffusion_coefficient
Tracer diffusion coefficient, D*.

Parameters **None** –

Returns The tracer diffusion coefficient, D*.

Return type (Float)

3.1.12 lattice_mc.species module

class lattice_mc.species.**Species**(*atoms*)
Bases: object

Species class.

Contains methods that operate on sets of Atom objects

collective_correlation()
Collective correlation factor, f_I, for these atoms.

Parameters **None** –

Returns The collective correlation factor, f_I, for these atoms.

Return type (Float)

collective_dr_squared()
Squared sum of total displacements for these atoms.

Parameters **None** –

Returns The square of the summed total displacements for these atoms.

Return type (Float)

occupations(*site_label*)
Number of these atoms occupying a specific site type.

Parameters **site_label** (*Str*) – Label for the site type being considered.

Returns Number of atoms occupying sites of type *site_label*.

Return type (Int)

sites_occupied()
Returns a list of sites occupied by these atoms.

Parameters **None** –

Returns List of sites occupied by these atoms.

Return type (List)

sum_dr_squared()
Sum of squared total displacements for these atoms.

Parameters **None** –

Returns The sum of squared total displacements for these atoms.

Return type (Float)

summed_dr2()
Sum of squared individual displacements for these atoms.

Parameters **None** –

Returns The sum of squared individual displacements for these atoms.

Return type (Float)

tracer_correlation()
Tracer correlation factor, f, for these atoms.

Parameters **None** –

Returns The tracer correlation factor, f, for these atoms.

Return type (Float)

3.1.13 lattice_mc.transitions module

class lattice_mc.transitions.**Transitions**(*jumps*)
Bases: object

Transitions class

Contains methods that operate on sets of Jumps.

cumulative_probabilities()
Cumulative sum of the relative probabilities for all possible jumps.

Parameters **None** –

Returns Cumulative sum of relative jump probabilities.

Return type (np.array)

random()
Select a jump at random with appropriate relative probabilities.

Parameters **None** –

Returns The randomly selected Jump.

Return type (*Jump*)

time_to_jump()
The timestep until the next jump.

Parameters **None** –

Returns The timestep until the next jump.

Return type (Float)

3.1.14 Module contents

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index

M

N

O

P

R

S

T

U

V