KineCluE v1.0: Input documentation

Thomas Schuler, Luca Messina, Maylise Nastar

August 1, 2019

Contents

1 Introduction

KineCluE is an open-source software distributed under the GNU Lesser General Public License (GNU-LGPL), version 3. For more details and the full licensing terms, see the COPYING and COPYING.LESSER files in the main directory.

KineCluE aims at computing cluster transport coefficients for general homogeneous crystallographic systems from the atomistic description of evolution mechanisms. It is based on the self-consistent mean-field theory (see references below). It is written in Python 3.6 and supported for this version only. Technical references can be found in T. Schuler, L. Messina and M. Nastar, Computational Materials Science xxxx (2019). Users are required to cite the above reference when using KineCluE or part of KineCluE in their own work.

References for the self-consistent mean-field theory:

– M. Nastar *et al.*, Philosophical Magazine **80** 155-184 (2000)
– M. Nastar *et al.*, Philosophical Magazine **85** 3767-3794 (2005)
– V. Barbe *et al.*, Philosophical Magazine **86** 3503-3535 (2006)
– T. Garnier *et al.*, Physical Review B **88**, 134201 (2013)
– M. Nastar, Physical Review B **90** 144101 (2014)
– L. Messina *et al.*, Physical Review B **90**, 104203 (2014)
– T. Schuler *et al.*, Physical Review B **93** 224101 (2016)

Contact information:

– Thomas Schuler (thomas.schuler@cea.fr)
– Luca Messina (messina@kth.se)
– Maylise Nastar - (maylise.nastar@cea.fr)

2 Installing and running the code

The code requires Python 3.6 and the following python modules: *copy*, *datetime*, *itertools*, *logging*, *mpmath*, *numpy*, *os*, *_pickle*, *psutil*, *random*, *scipy*, *shutil*, *sympy*, *sys*, *time*, *warnings*. All of them should be available in the traditional Python distributions, or easily retrievable. For instance, installing Python 3.6 with the Anaconda distribution is a simple way to get all required modules. We refer to the general Python documentation for detailed instructions about the installation of Python and its modules. The user can either set up an environment variable *python3* pointing towards the path of the python 3.6 executable, or include this path in the first line of the *kineclue_main.py* and *kineclue_num.py* files. Third option is to call manually the python executable each time the code is launched.

For a given cluster, two calculations are required: the analytical formulation of the problem with *kineclue_main.py*, and the numerical application with *kineclue_num.py*. The third file *kinepy.py* is a module containing the classes and functions needed by the other two scripts. It should be placed either in the same directory as the other scripts, or in the directory that contains all the other installed Python modules. Running KineCluE consists therefore of three steps:

1. Compute the analytical expressions with *kineclue_main.py* by running:

   » *path_to_kineclue_directory*/kineclue_main.py *main_input_file*

   The analytical results will be stored in a *analytical_kineclue_ouput.pkl* file that is then used by *kineclue_num.py*.

2. Running *kineclue_main.py* has produced as well two files *configurations.txt* and *jump_frequencies.txt*, which contain lists of symmetry-unique configurations and jump frequencies. The user has to provide numerical values of the equilibrium and saddle-point energies as well as prefactors, either manually in these two files, or by means of an interaction file (see INTERACTIONMODEL). It is also possible to leave the default values, or to let the code assign random values (see RANDOM).

3. Perform the numerical computation of the transport coefficients with *kineclue_num.py* by running:

   » *path_to_kineclue_directory*/kineclue_num.py *num_input_file*

Note that it is possible to run the numerical part of the code several times using the same analytical results, for instance if one wants to study the effect of set of jump frequencies or binding energies. In case the prefactor and energy numerical values do not need to be manually input, it is also possible to prepare all inputs beforehand and launch both codes in a row by using the RUNNUM keyword, or alternatively with a one-line command as follows:

» kineclue_main.py *main_input_file* ; kineclue_num.py *num_input_file*

Detailed instructions on how to fill in the input files for both the analytical and numerical part of the code are provided in Sections 3 and 4. Then, Section 5.2 contains a summary and a brief explanation of the produced output files and how they can be used.

3 Main input file

In the input files, each command is given by keywords (or tags), introduced by "&" symbols. Keywords can be written in lower or upper case indifferently, and in any order. Anything that is written before the first "&" symbol is considered as comment and will be disregarded by the code. By using the "#" symbol, it is also possible to include in-line comments on any line of the input, or comment out an entire line.

As a first example, for the simple case of substitutional solute diffusion in face-centered cubic systems, the minimum input file for the main program is the following:

```
& KINETICRANGE 2.0
& CRYSTAL fcc
+0.0 +0.5 +0.5
+0.5 +0.0 +0.5
+0.5 +0.5 +0.0
& UNIQUEPOS 1
o 0 0 0
& SPECIES 2
1 1 vac
1 1 sub_solute
& JUMPMECH
%% 2 vac_jump
o 1 0 0 0 > 1 0 .5 .5
o 0 0 .5 .5 > 0 0 0 0
%% 2 substitutional_solute_exchange
o 2 0 .5 .5 > 2 0 0 0
o 1 0 0 0 > 1 0 .5 .5
```

Below follow the descriptions of each keyword and the required formats. The necessary keywords are:

– & KINETICRANGE
– & CRYSTAL
– & UNIQUEPOS
– & SPECIES
– & JUMPMECH

and the optional keywords are:

– & DIRECTORY

- & CPG
- & NORMAL
- & STRAIN
- & BASIS
- & INICONF
- & PRINTXYZ
- & PRINTSYMEQS
- & BULKSPECIES
- & KIRALOOP
- & RUNNUM

3.1 & KINETICRANGE

Description The kinetic range "KiRa" defines the cluster radius. The code will include all configurations where any two components of the cluster are at most at KiRa distance from each other, as well as all kinetic trajectories between these configurations. With this keyword, the user can also define a thermodynamic range "ThRa", according to which any two components farther away than ThRa do not interact thermodynamically. For instance, for a pair of components A and B each having a single site occupation, all configurations where the AB distance is larger than ThRa and smaller than KiRa are assumed to have the same energy (arbitrarily set to zero). If species A has two site occupancies (for instance octahedral and tetrahedral sites), then all configurations where the AB distance is larger than ThRa and smaller than KiRa are divided into two sets of configurations (one for each type of site occupation for A), and all configurations within a set are given the same energy. Of course, there can be more than two different site occupancies. As a consequence, all these configurations can be described by a single variable, which greatly decreases the computing time for both the analytical and numerical parts. Moreover, the amount of symmetry-unique configurations and jump frequencies to be considered in the numerical treatment depends on ThRa only, which makes the compilation of the numerical values easier when for instance checking the convergence of transport coefficients with increasing KiRa.

Format One number (float), given in units of lattice parameter. The second number (float) for ThRa is optional and is also given in units of lattice parameter. If not specified, or larger than KiRa (which is physically nonsensical), the code assumes ThRa=KiRa.

Example To set the kinetic range to 4 lattice parameters, and the thermodynamic range to 1.5 lattice parameters:

```
& KINETICRANGE 4.00 1.50
```

3.2 & CRYSTAL

Description Lattice vectors defining the periodicity of the system. By default, the code creates a Bravais lattice (one basis atom only) based on these three vectors, but non-Bravais lattices can be generated using the optional BASIS keyword. It is always more efficient to reduce the crystal to its primitive cell. For instance, the calculation of transport coefficients for a face-centered cubic lattice will be much more efficient if CRYSTAL defines the primitive cell with only one basis atom instead of the cubic cell with 4 basis atoms. The code supports 1-, 2- and 3-dimensional systems. In all keywords except STRAIN , atomic coordinates should always have 3 components. If one is working in a 2-dimensional system, then the third coordinate should always be set to 0; if one is working in a 1-dimensional system, then the second and third coordinates should always be set to zero.

Format One crystal name (string). The crystal and the associated symmetry operations are created during the calculation and then saved into a file *crystalname.pkl.* Then n lines each containing n float numbers each, each line corresponding to one lattice vector, given in a n-dimensional orthonormal framework, in units of lattice parameter. Note that the division sign "/" and the square-root command "sqrt()" can be used to declare these vectors.

Example 3D Body-centered cubic system:

```
& CRYSTAL bcc
+0.5 +0.5 -0.5
+0.5 -0.5 +0.5
-0.5 +0.5 +0.5
```

Example 3D Hexagonal compact system:

```
& CRYSTAL hexagonal
+1.0 +0.0 +0.0
-0.5 sqrt(3)/2 +0.0
+0.0 +0.0 sqrt(8/3)
```

Example 2D square lattice:

```
& CRYSTAL 2d_cubic
+1.0 +0.0
+0.0 +1.0
```

3.3 & UNIQUEPOS

Description Symmetrically unique lattice positions inside the supercell defined by CRYSTAL . The aim is to declare specific positions (sublattices)–either on- or off-lattice–that can be occupied by the cluster components, e.g. crystal sites for vacancies or substitutional atoms as well as interstitial sites or dumbbell configurations. Periodic boundary conditions are applied for sites located outside the supercell, and all positions are reduced to the symmetrically distinct ones only.

Format One integer number specifying how many positions are defined. Then, one line for each position, each containing one letter "o" or "s" followed by three float numbers. The letter specifies the coordinate system in which the lattice position is given, either orthonormal ("o") or supercell ("s"), i.e., with respect to the vectors defined in CRYSTAL . For off-centered atoms (e.g. dumbbells), the letter d representing a small displacement from the lattice site is used to define the position. In particular, for dumbbell-type defects it is sufficient to define the position of one of the two atoms in the dumbbell. Since transport coefficients are not affected by the numerical value of d, the code will automatically assign it to a small value, aimed exclusively at determining the correct symmetries of the system. Arithmetic operations with d are also allowed (e.g. $0.5 + d$).

Example This first example describes a system where solutes and point defects can occupy two types of lattice sites, described here in orthonormal coordinates: substitutional site (first one) and interstitial site (for cubic lattices).

```
& UNIQUEPOS 2
o 0.0 0.0 0.0
o 0.5 0.0 0.0
```

Example In this second example, the substitutional site is still available, but atoms can also be placed in off-centered positions and exist as dumbbells with [1 1 0] or [1 1 1] orientations. In this way, the code can handle cases where dumbbells can assume different orientations depending on their composition (i.e. pure vs mixed dumbbells).

```
& UNIQUEPOS 3
o 0.0 0.0 0.0
o d d 0.0
o d d d
```

3.4 & SPECIES

Description Specifies the species and components of the cluster. There can be several components of the same species, i.e. distinct components sharing similar properties in terms of site occupations, jump mechanisms and chemical potentials (for instance two or more vacancies). In the current version of KineCluE, the number of components is constant throughout the whole calculation (canonical ensemble). It is important to keep this feature in mind, especially when defining jump mechanisms.

Format One integer number corresponding to the number of species. Then, one line for each species, each containing $N+1$ numbers and one string, where N is the number of positions listed in UNIQUEPOS . The first number is an integer and corresponds to the number of components of this species. The total number of components in a cluster is thus the sum of the first number of each line in SPECIES . Note that computational time increases fast with the number of components. The N following numbers–one for each lattice position–define if that species can occupy the corresponding lattice position (the order corresponds to the one in UNIQUEPOS). The value for these "permissions" can be 0 (not allowed), 1 (allowed), or $-m$, where m is an integer number corresponding to any species. A negative permission $-m$ marks that a given sublattice is allowed only when a component of species number m is also present in the same site at the same time. For instance, a species with permissions "$1, 0, -2$" can occupy the third sublattice in UNIQUEPOS only if the species defined in second position is on that site as well. This enables the definition of mixed-dumbbell formation (see example below) or interstitial atoms located on a vacancy site. Finally, the string simply denotes the name of this species; it does not affect the calculation but it is used for output purposes.

Example Cubic system containing a substitutional vacancy and two interstitial solutes:

```
& UNIQUEPOS 2
o 0.0 0.0 0.0
o 0.5 0.0 0.0
& SPECIES 2
1 1 0 vacancy
2 0 1 interstitial_solute
```

Example Cubic system containing one substitutional solute and one self-interstitial atom. The self-interstitial atom can occupy the second sublattice only (corresponding to a dumbbell configuration), while the solute can be either on the crystal sites, or on a dumbbell off-centered position but only if the self-interstitial atom is also located on that same site. Note that the number of atoms in the system does not vary between these two configurations (see the JUMPMECH keyword below for a detailed example).

```
& UNIQUEPOS 2
o 0.0 0.0 0.0
o d d 0
& SPECIES
1 1 -2 solute
1 0 1 self-interstitial
```

3.5 & JUMPMECH

Description With this keyword, users define the jump mechanisms allowed in the system. A jump mechanism is described by a set of constraints defining the relative positions of various species required for a jump to happen. This is a crucial part of the code: all positions defined in UNIQUEPOS should be reachable using the defined set of jumps, otherwise some configurations will not be explored. This is checked by the code, and a warning is produced in the log file in case some configurations seem unreachable. Also, the user must make sure that in each mechanism the number of components (and the number of bulk atoms in the implicit lattice) does not vary, especially when dealing with point defects. For instance, if the cluster only contains one interstitial solute, this solute cannot jump into a substitutional position, because the code has no means to know what happened to the bulk atom sitting there. If this bulk atom formed a dumbbell with one of its neighbors, it would be legitimate from a matter conservation standpoint, but it would entail the creation of an additional component (a self-interstitial atom, in addition to the substitutional solute), and this is not supported by the current version of KineCluE. Note that all symmetry equivalent mechanisms will be computed automatically. Note that when looking for valid jump mechanism for a specific configuration, the code checks that no unconstrained component is located close to any of the constrained sites. The unconstrained components are the ones that were not assigned position in the jump mechanism definition; the constrained sites are all the sites (initial and final positions) appearing in the jump mechanism constraints.

Format Below the JUMPMECH keyword, each jump mechanism starts with a line containing the "%%" symbol followed by a number (number of constraints) and a string defining the jump mechanism name. The same names will be used by the numerical input as well, to define KRAACTIVATION energies. Then, a new line (one for each constraint) defines the jump. Each line contains a letter "o" or "s", marking the coordinate system in which the positions are given (the code will check that each position corresponds to one that is defined in UNIQUEPOS). Then, an integer indicates the species that is constrained in the initial position (1 for the first species described in SPECIES , 2 for the second one, *etc.*), followed by three float numbers (which may contain the letter *d* as for UNIQUEPOS in the case of dumbbells). Then, the symbol ">" separates the initial (before jump) from the final (after jump) positions for this constraint. This symbol is followed by four numbers, using the same format as the initial state: one integer for the final species, and three floats for the final position. Note that in the current version, final and initial species must be identical for a given constraint, as the code does not handle transmutations. Species 0 stands for bulk species, which translates in the code in the fact that none of the components of the cluster are allowed on this site. Be careful when setting up bulk species constraints in jumps as it may modify the jump symmetry and generate more jumps than what is really wanted. Always check the coherency between the configuration (including bulk constraint) after and before the jump. The code will automatically check the consistency of each jump mechanisms in terms of lattice permissions for each species and component overlapping, but it does not verify if bulk atoms are correctly defined on actual crystal sites, so the user must pay attention to it. Furthermore, the code will look for all symmetry equivalents of these jumps, automatically add reversed jumps if necessary, and check that two jump mechanisms defined in the input are not symmetrically equivalent. Note that not all components of a cluster need to be constrained in each jump mechanism.

Example Vacancy jump and vacancy-solute exchange in face-centered cubic system (species 1 is the vacancy, species 2 is the substitutional solute):

```
& JUMPMECH
%% 2 vac_jump_fcc
o 1 0.0 0.0 0.0 > 1 0.5 0.5 0.0
o 0 0.5 0.5 0.0 > 0 0.0 0.0 0.0
%% 2 vac_sol_exchange_fcc
s 1 0.0 0.0 0.0 > 1 1.0 0.0 0.0
s 2 1.0 0.0 0.0 > 2 0.0 0.0 0.0
```

The second constraint of the first jump ("vac_jump_fcc") specifies that the destination site of the vacancy must not be occupied by other system components, *i.e.* it is occupied by a bulk atom or empty. It is important to understand that only jump *mechanisms* are given in the input, for instance a vacancy exchange with a neighboring (implicit) bulk atom (first mechanism above "vac_jump_fcc"). The code will automatically generate all "versions" of this mechanism, i.e. with solute being on first, second, etc. nearest-neighbor of the vacancy.

Example Interstitial solute jump in body-centered cubic system (solute is species 2):

```
& JUMPMECH
%% 2 inter_jump_bcc
o 2 0.5 0.0 0.0 > 2 0.5 0.5 0.0
o 0 0.5 0.5 0.0 > 0 0.5 0.0 0.0
```

Example Interstitial solute second nearest-neighbor jump in the vicinity of an immobile vacancy (vacancy is species 1, solute is species 2):

```
& JUMPMECH
%% 3 inter_2NNjump_bcc
o 2 0.5 0.0 0.0 > 2 0.0 0.0 0.5
o 1 0.0 0.0 0.0 > 1 0.0 0.0 0.0
o 0 0.0 0.0 0.5 > 0 0.5 0.0 0.0
```

Example Dissociation of a [110] mixed dumbbell (solute+self-interstitial atom) in body-centered cubic systems (species 2 is the solute). The code will automatically add the reverse mechanism, i.e., the association of the mixed dumbbell from a solute-pure dumbbell pair in first nearest-neighbor position. Note that a mixed dumbbell is seen by the code as a two-component cluster (a dumbbell and a solute).

```
& JUMPMECH
%% 2 mixed_dumbbell_dissociation
o 1 d d 0 > 1 0.5 0.5-d 0.5-d
o 2 -d -d 0 > 2 0.0 0.0 0.0
```

3.6 & DIRECTORY

Description Directory where all inputs and outputs will be stored.

Format String. If the directory does not exist, it will be created. A "/" sign will be automatically added at the end of the string, if not already present. If the input string is *cwd* (for "current working directory"), the output directory will be the one where the run command has been executed. This is useful when the user wants to run the code directly from a pre-created output directory (for instance, to re-run a calculation, or for automatic scripts), but be aware that any results already present in that folder will be overwritten.

Default "./CALC". This folder will be created if it does not exist.

Examples

```
& DIRECTORY folder1/folder2/output_folder
& DIRECTORY folder1/folder2/output_folder/
& DIRECTORY cwd
```

3.7 & CPG

Description In the self-consistent mean-field theory, transport coefficients are computed from fluxes in response to a chemical potential gradient (cpg) along a specific direction. Hence the calculation provides transport coefficients along this cpg direction. If the NORMAL keyword is specified, the code will compute the transport coefficients in perpendicular directions as well, always as a response to a driving force in the cpg direction. In an isotropic crystal such as cubic crystals, diffusion properties do not depend on the diffusion direction, hence the cpg vector. Nevertheless the choice of a cpg direction affects the efficiency of the calculation, because the number of crystal symmetries along the cpg direction is directly related to the number of effective interactions n that will be created. In the numerical part of the code, most of the time is spent inverting nxn matrices. Generally speaking, an efficient choice of cpg direction is one where the number of effective interactions is the smallest possible (unless one is looking for transport properties in a specific direction, then one should take the cpg along that direction).

Format One letter "o" or "s" to specify the coordinate system, and three float numbers to indicate the direction. The vector will be automatically normalized.

Default The default value is the first lattice vector defined in CRYSTAL .

Example

```
& CPG o 1.0 1.0 0.0
```

3.8 & NORMAL

Description Typically in strained systems, a chemical potential gradient in a given direction can eventually produce net species fluxes in other directions. NORMAL specifies a second direction along which the code will compute the flux resulting from a driving force in the CPG direction. This direction must not be parallel to the cpg direction, but it does not have to be perpendicular to it. The third diffusion direction is chosen as a cross product between CPG and NORMAL directions. Note that to get species fluxes in the NORMAL direction in response to a driving force along the same NORMAL direction, it is necessary to perform a second calculation using a different type of input where that direction is set with the CPG keyword.

Format Same as CPG .

Default If NORMAL is not defined, transport coefficients are computed in the CPG direction only (which is faster).

Example

```
& CPG o 1.0 0.0 0.0
& NORMAL o 1.0 1.0 0.0
```

3.9 & STRAIN

Description Allows to compute the transport coefficients under strain, by defining the deformation (strain) matrix that is applied to the lattice vectors defined in CRYSTAL . This can reduce the symmetry of the system, splitting symmetrically equivalent defects (UNIQUEPOS) and jump mechanisms (JUMPMECH) into non-equivalent ones. The code will perform the broken-symmetry analysis automatically if a non-zero STRAIN matrix is detected: the new found broken-symmetry defects and jump mechanisms will be listed in the main output, together with the corresponding symmetry matrices. The latter matrices are useful to perform the necessary rotations of the corresponding elastic dipoles in the numerical part (see keywords KRAACTIVATION and NUMSTRAIN). In addition, the code will also produce a *crystal_name_strained.pkl* (where *name* is the name of the crystal defined in CRYSTAL) file for the strained crystal, as well as a new input file listing the broken-symmetry elements (same name as the input file but with the _*strained.txt* suffix). The latter files can be used to re-run an analytical computation, if needed, without performing the broken-symmetry analysis.

Format A letter "s" or "o" defining the system coordinates for the strain matrix. Then, same format as CRYSTAL , but the matrix components correspond to strains. The letter "e" can be used to define a generic displacement length, whose magnitude can be assigned as a range of values in the numerical part. Arithmetic operations, powers ("**") and square roots ("sqrt()") are permitted. Note that all the coordinates in UNIQUEPOS , JUMPMECH , BASIS , as well as the directions in CPG , NORMAL are given in the non-strained crystal. The strained crystal matrix **SC** is given as **SC=(I+S)C** where **I** is the identity matrix, **S** is the strain matrix defined with keyword STRAIN and **C** is the crystal matrix defined with keyword CRYSTAL .

Default The strain matrix is full of zeros.

Example Volume-conserving shear strain:

```
& STRAIN o
0.0 e/2 0.0
e/2 0.0 0.0
0.0 0.0 ((e/2)**2)/(1-(e/2)**2)
```

3.10 & BASIS

Description For non-Bravais lattices, allows the description of systems with multiple basis atoms (*e.g.* hexagonal compact, diamond, 2D honeycomb lattice...). Note that it is not necessary to specify all basis atoms in UNIQUEPOS because they are symmetrically equivalent; thus, only one basis position should be added in UNIQUEPOS , even though it is not mandatory. In a nutshell BASIS coordinates are used to create the crystal and set up symmetry operations, while UNIQUEPOS coordinates indicate which sites are actually going to be occupied by each species.

Format A letter "o" or "s" (coordinate system) and an integer indicating the number of basis atoms. Then a line for each basis atom containing one integer and three float values. Symbols such as "/" or "sqrt()" are accepted. The first integer number represents the "species" of the basis atom. The value of this number does not have any meaning and has no connexion to the SPECIES keyword. When looking for symmetry operations, the code only considers translations between basis atoms having the same species. Hence adding a basis atom of a new species (i.e. simply a different integer number) is a way to reduce the symmetry of the system and produce more complex crystals, notably ordered systems.

Default Bravais lattice, i.e., only one basis position located in (0 0 0).

Example Bravais lattice (default value)

```
& BASIS o 1
0 0. 0. 0.
```

Example Hexagonal closed-packed systems (the two basis atoms are equivalent so they are assigned the same "species" number):

```
& CRYSTAL hcp
+1.0 +0.0 +0.0
-0.5 0.5*sqrt(3) +0.0
+0.0 +0.0 sqrt(8/3)
& BASIS s 2
0 1/3 2/3 1/4
0 2/3 1/3 3/4
```

Example 3D diamond structure (again the two basis atoms are equivalent):

```
& CRYSTAL diamond
0.5 0.5 0.0
0.5 0.0 0.5
0.0 0.5 0.5
& BASIS s 2
0 0 0 0
0 1/4 1/4 1/4
```

Example Zincblende-type structure (here the two basis atoms are made different, which reduces the symmetry of the system compared with the diamond structure above):

```
& CRYSTAL diamond
0.5 0.5 0.0
0.5 0.0 0.5
0.0 0.5 0.5
& BASIS s 2
0 0 0 0
1 1/4 1/4 1/4
```

Example 2D honeycomb lattice:

```
& CRYSTAL honeycomb
3/2 sqrt(3)/2
0.0 sqrt(3)
& BASIS s 2
0 0.0 0.0 0.0
0 2/3 2/3 0.0
```

3.11 & INICONF

Description Initial configuration of the cluster. Starting from this configuration, jump mechanisms and symmetry operations are applied successively to explore the configuration space. Note that if jump mechanisms allow to connect all possible cluster configurations, as described in UNIQUEPOS and SPECIES , then the final results are independent of INICONF .

Format An integer number corresponding to the number of components in the system (must be consistent with the description in SPECIES , same number of components and species). For each component, a line contains one letter ("o" or "s", coordinate system), one integer (species number, as in JUMPMECH) and three floats (position coordinates). The code will check that each component has at least one neighbor at a distance lower than KINETICRANGE (cluster connectivity), and that the configuration is consistent with the sublattice occupancy permissions defined in SPECIES and UNIQUEPOS . If any of these requirements is not fulfilled, the code will attempt finding a valid initial configuration automatically (in this case, a warning will be printed in the main output).

Default Based on SPECIES and UNIQUEPOS , the code attempts to find a valid initial configuration automatically. In case of failure, execution will stop, and a valid initial configuration will have to be input manually.

Example Start exploration of configuration space from self-interstitial atom (species 1) and solute (species 2) pair in first nearest-neighbor configuration in a body-centered cubic crystal:

```
& INICONF 2
o 1 d d d
o 2 0.5 0.5 0.5
```

3.12 & PRINTXYZ

Description This keyword is used to print configurations and jump frequencies in a format (such as *.xyz*) that can be read by visualization softwares. The files (one for each symmetry-unique configuration and jump frequency) are created inside two sub-folders in DIRECTORY called *CONFIGURATIONS* and *JUMP_FREQUENCIES*. In each sub-folder, the file numbering corresponds to that in the output files *"configurations.txt"* and *"jump_frequencies.txt"*. The latter are created by *kineclue_main.py* and are used to supply the energy and prefactor values to *kineclue_num.py*. Each file in the *CONFIGURATIONS* folder contains the number of components and a list of species names and coordinates. In the files of the *JUMP_FREQUENCIES* folder, there are three additional coordinates for each component, corresponding to its position after the jump, and preceded by the ">" symbol. Be aware that for thermodynamically dissociated configurations, the coordinates after the jump are always given with respect to the first component of the cluster. Hence it may seem that this component is not moving, while in fact it is. The jump mechanism name tells you what species are moving. Note that this option can create a large number of files.

Format One letter "o" or "s" for the coordinate system, and then a string or integer specifying if bulk atoms must be added. If "s" is chosen as the first letter, lattice vectors are also written in the file. If no string/integer is specified, only cluster components are printed. If the keyword "wrap" is given, then bulk atoms are added around cluster components. If an integer n is given, n bulk supercells are added in each of the three supercell directions.

Default These files are not written.

Examples Below are the possible values of the PRINTXYZ keyword.

```
& PRINTXYZ o
& PRINTXYZ s
& PRINTXYZ o wrap
& PRINTXYZ s wrap
& PRINTXYZ o 3
& PRINTXYZ s 3
```

3.13 & PRINTSYMEQS

Description Prints a file called *symeqs.dat* showing the symmetry equivalent of defects (defined in UNIQUEPOS) and jump mechanisms (defined in JUMPMECH) as computed by the code. This feature is intended to help the user check that all expected symmetry equivalents are there, and that none is unwanted. The format of jumps is similar to that of the input file: initial and final positions are separated by a ">" sign, and on each side the species positions are listed in the species order defined in the input file. Unless the "bulk" argument is given, all constraints for bulk species are not output to improve readability. If the list in this file is different from the expected one, the problem lies either in the definition of the supercell or in the definition of the defects/jump mechanisms.

Default This file is not written.

Examples Below are the possible values of the PRINTSYMEQS keyword. Providing no arguments after the keyword is strictly equivalent to the "s" argument.

```
& PRINTSYMEQS
& PRINTSYMEQS o
& PRINTSYMEQS s
& PRINTSYMEQS o bulk
& PRINTSYMEQS s bulk
```

3.14 & BULKSPECIES

Description This feature is intended to handle self-interstitial atoms in dumbbell configurations, meaning dumbbell defects created by adding a matrix atom to another one on the same site. This keyword takes a list of integers as arguments, which correspond to the indices of the species whose symmetry will be treated as if they were bulk atoms (index 1 correspond to the first species given in the SPECIES keyword). This feature has been tested only on dumbbells, so we strongly discourage its use for any other purpose. Be aware that performing calculations with bulk dumbbells without the BULKSPECIES keyword leads to an erroneous counting of the number of symmetry-equivalent configurations and jumps. Namely, all configurations entailing a pure dumbbell (surrounded by no matter what) will be double counted because by default the code distinguishes between a dumbbell in one orientation (for instance [110]) and the opposite orientation ($[\overline{1}\overline{1}0]$), whereas for bulk dumbbells the two orientations are symmetrically equivalent. This will yield an incorrect cluster transport coefficient since some–but not all–configurations are double-counted. Note also that in case the BULKSPECIES keyword is not used, the user must provide in the input file two instances of a pure-dumbbell jump, one for the forward jump and one for the backward jump (see example below). See also the input file examples in the *input_files* directory.

Format A list of integers specifying which species (same order as in SPECIES) are treated as bulk species regarding symmetries.

Default This feature is ignored.

Example In this example, the dumbbell (species 1) will be treated as a bulk species, hence a [110] dumbbell is regarded as equal to a $[\overline{1}\overline{1}0]$ dumbbell, unless a solute is occupying the same site as the dumbbell (in which case a mixed dumbbell is formed and the two above orientations are regarded as different).

```
& UNIQUEPOS 2
o 0.0 0.0 0.0
o d d 0.0
& SPECIES 2
1 0 1 self-interstitial-atom
1 1 -1 solute
& BULKSPECIES 1
& JUMPMECH
%% 2 dumbbell_jump
o 1 d d 0.0 > 1 0.5 0.5-d 0.5-d
o 0 0.5 0.5 0.5 > 0 0.0 0.0 0.0
```

Example In this counterexample, the BULKSPECIES feature is not activated. A [110] dumbbell is regarded as different from a $[\overline{11}0]$ dumbbell, hence two pure-dumbbell jumps are needed: a forward rotation-translation jump from the center to the (0.5, 0.5, 0.5) site, and a backward counterpart towards the (-0.5, -0.5, -0.5) site. Note that the final position of the dumbbell in the two jumps must not be perfectly symmetric with respect to the center $\left([x,y,z]^{\text{forward}}_{\text{end}} \neq -[x,y,z]^{\text{backward}}_{\text{end}}\right)$. The pure dumbbell configurations will be accounted for twice, which results in slower calculations and incorrect transport coefficients unless 1/2 factors are added for all of these configurations and the associated jump frequencies in the numerical input files.

```
& UNIQUEPOS 2
o 0.0 0.0 0.0
o d d 0.0
& SPECIES 2
1 0 1 self-interstitial-atom
1 1 -1 solute
& JUMPMECH
%% 2 dumbbell_jump
o 1 d d 0.0 > 1 0.5 0.5-d 0.5-d
o 0 0.5 0.5 0.5 > 0 0.0 0.0 0.0
%% 2 dumbbell_back_jump
o 1 d d 0.0 > 1 -0.5 -0.5-d -0.5-d
o 0 -0.5 -0.5 -0.5 > 0 0.0 0.0 0.0
```

3.15 & KIRALOOP

Description This keyword is required to perform the automatic convergence study of transport coefficients as a function of the kinetic radius in the numerical part of the code, using the same KIRALOOP keyword. When using this keyword, the kinetic radius (KINETICRANGE) must be set to the largest desired value, while it is advisable to choose a smaller value for the thermodynamic range, which will correspond in the numerical file to the lowest KiRa. Be aware that using this keyword, the code will be slower and the ".pkl" file containing the results of the analytical part of the code will be larger.

Format No argument is required.

Default The symbolic calculation is performed at the fixed kinetic radius defined with KINETICRANGE , and it will not be possible to use the KIRALOOP keyword in the numerical part of the code.

Example In this example, the code will create the necessary variables to perform, in the numerical part of the code, a convergence study in the kinetic radius interval that goes from ThRa to KiRa. If ThRa is omitted, only one KiRa value will be included in the convergence study.

```
& KINETICRANGE KiRa ThRa
& KIRALOOP
```

3.16 & RUNNUM

Description The RUNNUM keyword allows to run both parts of the code–*kineclue_ main.py* and *kineclue_ num.py*– one after the other. This is useful when the user does not need to analyze and modify manually the output files from the analytical part of the code. There are three main cases when this option should be used: 1) calculation of self-diffusion correlation factors because all configuration and jump frequencies are at their default numerical values; 2) using the RANDOM keyword; 3) using the INTERACTIONMODEL keyword because the code performs the analysis of all configurations automatically.

Format A string corresponding to the location and/or name of the numerical input file (see Sec. 4 for a comprehensive description of this file). Note that the DIRECTORY keyword in the numerical input–if it exists–will automatically be changed to match the one defined with the DIRECTORY keyword in the analytical input.

Default If this keyword does not belong to the input, the code will simply stop once the analytical part of the calculation is completed. The numerical part can then be launched using the following command as long as the same DIRECTORY keyword is specified in both analytical and numerical input files:

```
path_to_kineclue_directory/kineclue_num.py    num_input_file
```

Example

```
& RUNNUM num_input_file_path
```

4 Numerical input file

The numerical input file is used to perform the actual numerical calculation of cluster transport coefficients, after the analytical calculation has been performed. The analytical part of the code produces various outputs:

- ".pkl" files which are binary files, read by the numerical part of the code, and not intended to be read by humans;
- "configurations.txt" and "jump_frequencies.txt" which are described below, and basically contain all symmetry unique configurations and jump frequencies found by the code;
- "kineclue_main.log" which copies the standard output of the code. Before moving on to the numerical part of the code, users are advised to read this file and make sure the calculation went on as expected. At minimum, users should take a look at all the lines containing a double exclamation point "!!" which identifies warnings and/or default behavior of the code;
- optionally, a "CONFIGURATIONS" and "JUMP_FREQUENCIES" folders are created, as well as a "symeqs.dat" file. Information about these are given in PRINTXYZ and PRINTSYMEQS keywords.

Configuration file

The configuration file *configurations.txt* contains one line for each thermodynamically-distinct (i.e. symmetry-unique) configuration. On each line, the first number is the internal reference index, followed by a prefactor (no units), a binding energy (eV, positive means attraction), a string (explained hereafter) if the configuration is considered as dissociated by the code (i.e. the distance between some of the components are larger than the thermodynamic range value, see KINETICRANGE), an integer corresponding to the number of symmetrically equivalent configurations, and a list of coordinates (in the orthonormal framework) for each component in the cluster. For dissociated configurations, the string starts by letter "d" and is followed by a series of character separated by "|" symbols to specify subconfigurations in the cluster which are farther away than the kinetic range. Each subconfiguration is represented by a list of numbers separated by "_" symbols. Each one gives the number of components of each species (in the same order as in the main input) in the subconfiguration.

If the prefactor is -1.0 (default value) then the corresponding binding energy is zero, or random in case the keyword RANDOM is activated. Any negative value other than -1.0 will produce an error, while a positive value entails that the code will use the binding energy set by the user in this file. The prefactor actually represents the exponential of the entropy contribution to the system energy in this particular configuration, with respect to a reference configuration which is chosen by the user. Configurational entropy contributions are already accounted for by the code and should not be included in the prefactor value. Thus this prefactor theoretically contains the effect of vibrational, electronic and magnetic excitations, if any. Finally, setting the prefactor to zero removes the configuration from the calculation (for instance, if the configuration is unstable). In this case, for the sake of consistency, do not forget to set the corresponding jump frequencies to zero; alternatively, it is possible to leave the default value of these frequencies, in which case the code will automatically set them to zero.

Jump frequency file

The jump frequency file *jump_frequencies.txt* contains one line for each symmetry-unique jump frequency. Each line contains the internal reference index, followed by a relative prefactor (no units), a saddle-point energy (eV), two integers corresponding to the configuration index (same as in the configuration file) of initial and final configurations, the jump mechanism name and the number of symmetrically equivalent jumps. The jump mechanism name is used for Kinetically-Resolved Activation (KRA) approximations of saddle-point energies (see KRAACTIVATION). The saddle-point energy is the same for forward and backward jumps if detailed balance is fulfilled (which is assumed throughout the code), so the order of the configuration indices (initial vs final) does not matter. If the prefactor is -1.0 (default value) the saddle-point energy is estimated with the KRA approximation or with the INTERACTIONMODEL file. With a positive prefactor, the saddle-point energy input by the user is used. Again, the prefactor corresponds to the exponential of the vibrational, electronic and magnetic entropies, configurational contributions being already accounted for by the code. A common prefactor (reference entropy contribution) can be set with the PREFACTOR keyword. Jump frequencies can be removed from the calculation by setting their prefactor to zero.

Input of numerical values

There are two ways to supply numerical values for binding and saddle-point energies:

1. by manually modifying the files *configurations.txt*, and *jump_frequencies.txt*;
2. by using the INTERACTIONMODEL keyword to read the set of energies (binding and saddle-point) from a file containing the results of atomic scale calculations (for instance *ab initio* or interatomic potential).

Note that the binding energies in the configuration file and the saddle-point energies in the jump frequency file are defined with respect to the same reference energy, corresponding to the energy at which the cluster binding energy is null (usually the energy of one of the possible dissociated configurations).

After each run of *kineclue_num*, the values of the binding and saddle-point energies used in the calculation are available in the *num_configurations.txt* and *num_frequencies.txt* output files. It is useful to check this data, for instance when using the RANDOM keyword, or to check that the INTERACTIONMODEL was applied correctly, or to copy some of the values manually in the files *configurations.txt* and *jump_frequency.txt* for further numerical calculations. Note that jump frequency values computed with the KRA approximation might be temperature dependent with an effective saddle-point energy that does not evolve linearly with temperature; in this case, the values found in *num_frequencies.txt* are evaluated at the midpoint between the minimum and maximum temperatures given in the TEMPERATURES keyword. In strain calculations, such values are given at zero strain, while the corresponding strain-dependent values are to be found in the output files named *strain_num_conf.txt* and *strain_num_freq.txt*.

Input keywords

For the input file of *kineclue_main.py*, the necessary keywords are:

- & KRAACTIVATION
- & TEMPERATURES

and the optional keywords are:

- & DIRECTORY
- & NUMSTRAIN
- & OUTPUT
- & BATCH
- & BATCHCREATE
- & RANDOM
- & LATTPARAM
- & PREFACTOR
- & UNITS
- & MOB
- & SENSITIVITY
- & OUTOPTIONS
- & PRECISION
- & INTERACTIONMODEL
- & KIRALOOP
- & RANDERROR

These keywords are described below.

4.1 & TEMPERATURES

Description Temperature range (in K) over which the numerical calculation is performed.

Format Three integer numbers: minimum, maximum, step.

Example Compute the transport coefficients between 300 and 1000 K, at 10-K intervals:

```
& TEMPERATURES 300 1000 10
```

4.2 & KRAACTIVATION

Description Activation energies Q_j and prefactors ν_j for each jump mechanism, to be used in the kinetically-resolved activation model [*e.g.* A. Van der Ven *et al.*, Physical Review Letters **94** 045901 (2005)] (KRA, also known as linear interpolation of migration barriers (LIMB) or final-initial saddle point energy (FISE)). In this model, the saddle-point energy of a jump of type j is obtained from a reference energy Q_j for that jump and an average of the binding energies E_m^{b} and E_n^{b} of the initial and final configurations m and n as follows:

$$E_{j,mn}^{\mathrm{sp}} = E_{j,nm}^{\mathrm{sp}} = Q_j - \frac{1}{2}(E_m^{\mathrm{b}} + E_n^{\mathrm{b}}), \tag{1}$$

(positive binding stands for attractive interactions). The associated jump frequency is:

$$W_{mn}^{j} = \nu_j \exp\left(-E_{j,mn}^{\mathrm{sp}}/k_{\mathrm{B}}T\right), \tag{2}$$

where k_{B} is the Boltzmann constant, T is the absolute temperature and ν_j is the relative prefactor for this jump mechanism. The user must provide one value of Q_j, and optionally a value of ν_j, for each jump mechanism. If any of the jump mechanisms defined in the analytical part is missing, the calculation will fail. Any saddle-point energy provided in the *jump_frequency.txt* file with a positive prefactor will override the KRA calculation. For any saddle-point identified from the interaction file (INTERACTIONMODEL keyword), the interaction energy will be added to the KRA calculation.

Users should be aware that for systems under strain, additional jump mechanisms might be produced due to symmetry breaking: in this case, the user must check their automatically-created names in the **input_main_strained.txt* or in the *kineclue_main.log* file and provide a value of Q_j for each of these additional jumps. In addition, the user may provide the elastic-dipole values at the saddle point for each of the reference jumps, in the format specified below. The variation of binding and saddle-point energies as functions of strain are computed following linear elasticity theory (see Varvenne *et al.*, Phys. Rev. B **88**, 134102 (2013) for full details).

Format One line for each jump mechanism in the main (or strained) input. The line contains a string, corresponding to the jump mechanism name which must match that of the analytical calculation, and one or or two float numbers: if two numbers are provided, the first one is the relative prefactor (ν_j, no units), and the second one the activation energy $Q_j > 0$, in eV. Alternatively, it is possible to provide the activation energy only, in which case the prefactor will be automatically taken as equal to unity. Note that the total prefactor of the jump is calculated as the product of ν_j and the general prefactor ν_0 (in THz) defined with the PREFACTOR keyword.

The one or two floats can be optionally followed by a character "s" or "o" and 9 float numbers, used to assign the saddle-point elastic dipole for the reference jump mechanism, in case of strain calculations. The letter specifies the coordinate system as in the analytical input, whereas the 9 floats represent the components of the elastic-dipole matrix (in eV), in the following order: P_{11}, P_{12}, P_{13}, P_{21}, P_{22}, P_{23}, P_{31}, P_{32}, P_{33}. In 1- and 2-dimensional systems, user should still provide 9 components, and set the ones that are irrelevant to zero. Be sure to check which jump is assigned to each jump name in the strained input produced by the code, so that each jump is being assigned the correct elastic dipole.

Examples Below are examples of possible formats for the KRAACTIVATION keyword.

```
& KRAACTIVATION
3nn_vac 0.67 # activation energy only
3nn_vac 0.27 0.67 # relative prefactor and activation energy
# activation energy and elastic dipole in supercell coordinates
3nn_vac 0.67 s 0.1 0.02 0.0 0.02 0.1 0.0 0.0 0.0 0.2
# relative prefactor, activation energy and elastic dipole in orthonormal coordinates
3nn_vac 0.27 0.67 o 0.1 0.02 0.0 0.02 0.1 0.0 0.0 0.0 0.2
```

4.3 & DIRECTORY

Description Directory containing the results of the analytical calculation. Note that in case a directory was specified in the analytical calculation, then the same directory must be specified here.

Format String. The directory must exist. Analogously to the analytical input, a "/" sign is automatically added if not present to the end of the string, and it is possible to use the "cwd" option to run the code in the current working directory.

Default The same as for the analytical part of the code: "./CALC"

Examples

```
& DIRECTORY folder1/folder2/output_folder
& DIRECTORY folder1/folder2/output_folder/
& DIRECTORY cwd
```

4.4 & NUMSTRAIN

Description Strain range (no units) over which the numerical calculation is performed. The code will perform a loop over each strain value, substituting it into the e variable of the STRAIN keyword. In case no strained crystal file is found in the working directory, the code will neglect the strain range and perform all calculations at $e = 0$. When performing strain calculations, one can provide the values of the elastic dipoles for each equilibrium and saddle-point configuration (see KRAACTIVATION and INTERACTIONMODEL). When TEMPERATURES and NUMSTRAIN contain at least two values each, the code automatically uses *matplotlib* to plot transport coefficients as a function of strain and temperatures and saves the resulting plot as a pdf file in the DIRECTORY . Black contour lines may appear: solid lines indicate a change of sign, and dashed lines are spaced by two orders of magnitude, to help in visualizing the evolution of cluster transport coefficients.

Format Three integer numbers: minimum, maximum, step. Note that the strain values are not given in percents but in fractions.

Default Strain is set to zero.

Examples Calculation of transport coefficients between -1% and 1% strain, at 0.1% intervals:

```
& NUMSTRAIN -1e-2 1e-2 1e-3
& NUMSTRAIN -0.01 0.01 0.001
```

4.5 & OUTPUT

Description Name of the output files. The main output will be named "*output*_0.dat"; then, more files starting with the same *output* string will be written in case the keywords in OUTOPTIONS are activated.

Format One string.

Default The default value for the output file(s) is: "numerical_kineclue_output". The file(s) will be located in the DIRECTORY .

Example

```
& OUTPUT example_output_name
```

4.6 & BATCH

Description This option is designed to run batch calculations on a single system with the same kinetic and thermodynamic ranges. It is especially useful to analyze the effect of a given jump frequency (or a set of them), on cluster transport coefficients. In the DIRECTORY , the user must create a subdirectory called "batch" (lower case) which contains a series of files *configurations_ Y.txt* and *jump_frequencies_ Y.txt*, where Y is an integer number. In some cases, these files can also be created automatically using the BATCHCREATE keyword. The code will perform a numerical calculation and produce outputs with extension _ Y in the batch subdirectory for each value of Y. If both BATCH and RANDOM keywords are used simultaneously, there is no need for the user to create the "batch" sub-directory nor supply the set of *configurations_ Y.txt* and *jump_frequencies_ Y.txt* files. The code will always use the *configurations.txt* and *jump_frequencies.txt* files in the DIRECTORY and pick new random numbers for non user–specified quantities at each iteration.

Format Three integer numbers: begin value, end value, step for the Y variable.

Default No batch calculation is performed.

Example

```
& BATCH 0 10 1
```

4.7 & BATCHCREATE

Description This option is designed to create a series of files to use in conjunction with the BATCH keyword, for instance to vary the numerical value of binding or saddle-point energies. Before using this option, one must run some reference numerical calculation because the files *num_freq.txt* and *num_ conf.txt* (automatically produced by the numerical part of the code) are used by this function. Note that this option will erase the contents of the "batch" folder if it exists. For each configuration and jump frequency given as argument (see format below), the code will create files *configurations_ Y.txt* and *jump_frequencies_ Y.txt* (where Y is an integer number) where all binding and saddle-point energy values are kept equal to the ones in files *configurations.txt* and *jump_frequencies.txt* except the one(s) indicated which is (are) shifted by some energy value, taking the value in files *num_ conf.txt* and *num_freq.txt* as references. In addition to the aforementioned files, this function creates a summary file called *batch_ table.txt* which contains for each value of "Y" the index of configuration/jump frequency that has been changed, and the chosen value. When two values are changed at the same time, a "*" symbol separates indices and energy values.

Format One float (maximum energy variation), one integer number (number of points), and a series of labels in one of three formats: an or an-m or an*m. "a" is a letter that must be either "c" (configuration) or "j" (jump frequency); "n" and "m" are integer numbers corresponding to jump frequency or configurations indexes, as in the first column of files *configurations.txt* and *jump_frequencies.txt* specifying which configurations/jump frequencies will be modified in the batch files. "an" only modifies the configuration/jump frequency numbered "n", while "an-m" modifies all those between index "n" and index "m". Finally, "an*m" explores the "plane" of possible values for both "n" and "m" configurations/jump frequencies energies, meaning that both energies will be made to vary at the same time.

Default No batch files are created.

Example In this example, all specified binding and saddle-point energies will be varied between -0.5 and 0.5 eV (first float) by step of 0.1 eV (0.5/5). This will be done for configurations 2,12,24,25,26,27 and jump frequencies 4,5,6,9, for a total of (2x5+1)*10=110 calculations. Also binding energies for both configurations 4 and 12 will be varied together (totalizing (2x5+1)*(2x5+1)=121 additional calculations).

```
& BATCHCREATE 0.5 5 c2 c12 j4-6 c24-27 j9 c4*12
```

4.8 & RANDOM

Description Assigns random binding energies over a specified range for each configuration in *configurations.txt* whose prefactor is left equal to -1.0. Note that random binding energies affect all jump frequencies that are not explicitly assigned a saddle-point energy in the *jump_frequencies.txt* file, as the KRA model is applied (see KRAACTIVATION).

Format Two float numbers: minimum and maximum for the random binding energy generation, given in eV. Positive binding energies mark attractive interactions.

Default No random binding energies. All configurations with prefactor -1.0 in the *configurations.txt* file have zero binding energy (non-interacting configuration).

Example

```
& RANDOM -0.2 0.2
```

4.9 & RANDERROR

Description Shifts all binding and saddle-point energies by some random value (one value for each energy) to study how transport coefficients vary when input parameters are changed. This keyword is designed to be used with the BATCH keyword to perform statistical analysis. There is no need to create the *batch* folder or any additional files when using RANDERROR .

Format One or two float numbers: minimum and maximum for the random energy shift generation, given in eV. If only one number is provided, the maximum is taken as the absolute value of this number, and the minimum is taken as the opposite of this maximum. Then a word can be added among *allbut* or *nonebut*, followed by a sequence of configuration and/or jump frequency indexes *an* or *an-m*, using the same format as in the BATCHCREATE keyword (except the * notation does not apply here). If the *allbut* string was given, all binding and saddle point energies will be shifted except the ones listed; on the contrary, if the *nonebut* string was given, the energies listed afterwards will be the only ones to undergo a random shift. If non of these two string was given, all binding and saddle-point energies are affected by the random shift. Note that a negative shift leads to lower-energy configurations (higher binding energies or lower saddle-point energies) while a positive shift leads to higher-energy configurations (lower binding energies or higher saddle-point energies). Keep in mind that shifts in configuration energies will automatically produce shifts in KRA-computed saddle point energies (see KRAACTIVATION keyword), even if these jump frequencies were "blocked" with *allbut* or *nonebut* keywords.

Default No random energy shifts, which is equivalent to setting both minimum and maximum values to 0.

Example

```
& RANDERROR 0.05
& RANDERROR -0.08 0.12
& RANDERROR 0.05 allbut c2 j3-7 c4-5 j12
& RANDERROR -0.1 0.1 nonebut c2 j3-7 c4-5 j12
```

4.10 & LATTPARAM

Description Lattice parameter of the system, which represents the unit in which all distances in the code are defined. Cluster transport coefficients are always proportional to the square of the lattice parameter.

Format One float number (in Å).

Default 1.0 Å

Example

```
& LATTPARAM 2.831
```

4.11 & PREFACTOR

Description Global migration rate prefactor that applies to all jumps in the system (corresponding to the ν_0 quantity in KRAACTIVATION).

Format One float number (in THz).

Default 1.0 THz

Example

```
& PREFACTOR 10
```

4.12 & UNITS

Description Defines the transport coefficient units in the output files. Cluster transport coefficients are usually given in m^2/s, while total transport coefficients are generally in $(\mathrm{m}\cdot\mathrm{s}\cdot\mathrm{eV})^{-1}$. To convert one unit to the other, the code automatically computes the atomic volume, as well as the temperatures in energy units ($k_\mathrm{B}T$). The atomic volume is computed as the volume of the supercell defined in CRYSTAL is divided by the number of positions defined in BASIS . The "debug" unit simply multiplies the default value by 10^8, because default values for LATTPARAM and PREFACTOR amount to 10^{-8} m^2/s. So it removes this "physical" unit factor, which may help in debugging tricky systems.

Format One string among 7 accepted values: "m2/s", "cm2/s", "m2/s/eV", "cm2/s/eV", "/m/s/eV", "/m/s" or "debug".

Default "m2/s" as for tracer diffusion coefficients.

Examples

```
& UNITS m2/s # default value
& UNITS cm2/s
& UNITS m2/s/eV
& UNITS cm2/s/eV
& UNITS /m/s
& UNITS /m/s/eV
& UNITS debug
```

4.13 & MOB

Description If included in the input, this keyword activates the computation of the mobility coefficients (by repeating the loop over temperatures and strains). Note that without mobility coefficients it is not possible to compute the exchange coefficients (*EX* keyword in the OUTOPTIONS keyword).

Format Nothing, just include the MOB keyword itself if mobility coefficients are to be computed.

Default The mobility coefficients are not computed, just the cluster transport coefficients.

Example

```
& MOB
```

4.14 & PRECISION

Description Defines the numerical precision of the calculation. With respect to the default precision (15 digits), this keyword activates another routine for linear algebra operations, which considerably increases the computational time (so even indicating a precision of 15 digits or less with this keyword will make the calculation much slower). A good way to evaluate if a higher precision than the default one is necessary is to check if the transport coefficient matrix is not perfectly symmetric (this happens most likely at low temperatures and/or for large saddle-point energy differences). We recommend to use the PRECISION keyword only if the matrix is not symmetric within the desired precision, or for any other result requiring high accuracy (such as for instance correlation factors). Note that when the PRECISION keyword is used, the SENSITIVITY keyword is ignored.

Format Integer value giving the number of precision digits for numerical calculations.

Default Numerical calculations and linear algebra operations are performed to the default working precision (15 digits). This does not necessarily entail the results being accurate up to the 15th digit, because most likely the most important source of uncertainty will come from the input data (binding and saddle-point energies).

Example Increase the precision in the transport coefficient calculation to 30 decimal digits:

```
& PRECISION 30
```

4.15 & NDIGITS

Description Sets the number of decimal digits that the results are printed with. It can be necessary in cases where the differences between transport coefficients are very small (smaller than the default number of digits). It affects the transport and mobility coefficients in the main output file, as well as the results in the auxiliary output files (see section OUTOPTIONS). NDIGITS cannot be larger than PRECISION – in the opposite case, it is automatically set equal to PRECISION .

Format Integer value corresponding to the number of desired decimal digits.

Default Default value is 6 decimal digits. In case NDIGITS > PRECISION , the code sets NDIGITS = PRECISION automatically.

Example

```
& NDIGITS 12

& NDIGITS 24
& PRECISION 30  Increase default precision to get 24 decimal digits
```

4.16 & OUTOPTIONS

Description Prints more output files containing, depending on the selected options: exchange coefficients, the uncorrelated part of the transport coefficient matrix, drag ratios, correlation factors, or transport coefficients in directions normal to the CPG . Each output file has the same name as the one defined in the OUTPUT keyword, with the addition of a suffix corresponding to the options below.

Format Any of the following strings (also more than one, or even all of them):

- "*EX*" for exchange coefficients $E_{\alpha\beta} = L_{\alpha\beta} - M$ (output file: "*outputname*_0EX.dat");
- "*UC*" for uncorrelated contributions $L^0_{\alpha\beta}$ (output file: "*outputname*_0UC.dat");
- "*DR*" for drag ratios $L_{\alpha\beta}/L_{\beta\beta}$ (output file: "*outputname*_0DR.dat");
- "*CF*" for correlation factors $L_{\alpha\beta}/L^0_{\alpha\beta}$ (output file: "*outputname*_0CF.dat");

– "*ND*" for transport coefficients in directions normal to the CPG . In this case, two additional files are created: "*outputname*_1.dat" and "*outputname*_2.dat", where direction 1 is the direction defined with the NORMAL keyword in the analytical part, and direction 2 is the third automatically-computed one, perpendicular to both CPG and NORMAL . In case the NORMAL keyword was not defined in the analytical input, this option will be ignored and the coefficients computed only along the CPG direction (file "*outputname*_0.dat").

Default If no OUTOPTIONS keyword is given, the code computes only cluster transport coefficients in the CPG direction and produces only the output file "*outputname*_0.dat".

Examples Any combination of the above options is accepted and the order does not matter.

```
& OUTOPTIONS UC
& OUTOPTIONS EX DR
& OUTOPTIONS EX UC DR CF ND
```

4.17 & KIRALOOP

Description With KIRALOOP the code automatically performs a convergence study of transport coefficients as a function of the kinetic range. This feature can only be used if the keyword KIRALOOP was used in the analytical part of the code. The convergence study is performed from the ThRa to the KiRa values defined in the analytical part, on a discrete set of values selected in such a way that at least one effective interaction is added from one value to the next one. The number of valid KiRa values found can be retrieved from the *kineclue_main.log* file. The user can choose to perform the study on all values, or on a subset of them (see next paragraph). A full calculation of transport coefficients is performed for each kinetic range value, each temperature in TEMPERATURES and each strain in NUMSTRAIN . One output file for each strain-temperature-direction triplet is produced in a sub-folder of DIRECTORY called *CONVERGENCE_STUDY*. In these files, the first column is the kinetic radius used for the calculation and following columns are: strain, temperature, partition function, partition function without interactions, transport coefficients and relative error of the off-diagonal coefficients times the cluster partition function with respect to the value obtained for the maximum kinetic range (i.e. the one given with the KINETICRANGE keyword in the main input file). Note that mobility, dissociation coefficients and eventually site interactions are not computed during the convergence analysis. Information about the practical use of convergence analysis results are provided in Sec. 5.4.

Format One integer n specifying how many (1 out of n) of KiRa values found in the analytical part of the code will be used in the convergence study. If no integer is specified, the code assumes $n = 1$ (i.e. all KiRa values are used).

Default Without the KIRALOOP keyword, the convergence study will not be performed. If it is given with no argument, it is strictly equivalent to KIRALOOP 1.

Examples Perform the convergence study with 1 out of 3 values among those found in the analytical part of the code.

```
& KIRALOOP 3
```

4.18 & INTERACTIONMODEL

Description Reads energies, prefactors, and optionally elastic dipoles from an interaction model file that must have a specific format described below. This is useful for instance to provide values from *ab initio* or molecular dynamics calculations, and to run the two steps of KineCluE in an automated fashion, without the need of inputting the numerical values manually. Also, for strain calculations this is the only way to provide the elastic-dipole values for equilibrium configurations and saddle points that are not already defined in the KRAACTIVATION keyword. The code will read the provided configurations and jump frequencies, compare them with the ones output in the analytical calculation, and set binding and saddle-point energies accordingly. Similar to JUMPMECH , the "bulk" label can be used in jump frequencies to specify lattice sites that must not be occupied by other cluster components. Note that it is possible to provide also binding energies for sub-configurations, *i.e.*, configurations that contain only a subset of the cluster components: for instance, for a three-component cluster, it is possible to provide pair binding energies, which will be added to each configuration containing the specified sub-configuration. In the current version, an analogous sub-jump feature (same as sub-configuration but for jump mechanisms) is not yet implemented, so only jumps which constrain all cluster components will be taken into account and the saddle-point energy in the interaction model file is assigned to this jump (no additive feature for jumps). Nevertheless, note that if a configuration is thermodynamically dissociated (at least one component is farther away than the thermodynamic range) then the codes looks for jumps in all sub-clusters. This ensures that a pair of atoms will behave in the same way if it is isolated (cluster of two components) or dissociated from the other components of a larger cluster. In the interaction model file, jumps that do not constrain all cluster components will only be read for thermodynamically dissociated configurations.

Note that all values defined in the interaction model file will override the data contained in *configurations.txt* and *jump_frequencies.txt* having a prefactor equal to -1, as well as the RANDOM keyword. In any case, the user can check the correct application of the interaction model either in the main output *kineclue_num.log*, or in the *num_conf.txt* and *num_freq.txt* files. Also, refer to Sec. 5.3 for more information about how numerical values are assigned.

Format (in num input file) After the INTERACTIONMODEL keyword, a string that specifies the path to the interaction model file.

Example

```
& INTERACTIONMODEL mypath/mypath2/interaction_model_file.txt
```

Format (of interaction model file) The interaction model file must have a specific format. Configurations and jump frequencies are described in a list of entries, each introduced by the "&" symbol, followed by a capital letter: "C" for configuration or "J" for jump frequency. The interaction is ignored if any other letter is given, or if the "&" symbol is preceded by a "#" sign (with no blank spaces!). Afterwards, on the same line, a second letter ("o" or "s") specifies the coordinate system for the atomic positions, and an integer number marks the number of components in this entry (including bulk atoms if any). This is then followed by a series of float numbers, among one of the following possibilities:

– 1 float: binding energy of a configuration (positive means attraction) or saddle-point energy of a jump frequency (positive means repulsion). Energies are given in eV.

– 2 floats: the first number is the relative prefactor for that configuration or jump, while the second one is the binding or saddle-point energy.

– 10-11 floats: energy, or prefactor–energy pair, followed by the nine components of the elastic dipole (in eV) given in the following order: P_{11}, P_{12}, P_{13}, P_{21}, P_{22}, P_{23}, P_{31}, P_{32}, P_{33}. In strain calculations, if no elastic dipole is provided, the code will assume a null matrix.

After the line introduced by "&", the code expects one line for each component of the interaction (including bulk atoms) in the configuration or jump frequency, to define their positions, consistently with the number of components specified in the previous line. Each of these lines has a species name (which must be "bulk" or match those used in the SPECIES keyword of the analytical input file) followed by three floats (the atomic coordinates of the species). For jump frequencies, the first three floats define the initial position, and are followed by other three floats for the final position, separated by the ">" symbol. The letter "d" can be used to define dumbbell positions, in the same way as in UNIQUEPOS or JUMPMECH . Bulk atoms are designated with the species name "bulk". Note that any species labeled with a name different from those in SPECIES or "bulk" will be ignored. If the input values come from atomistic calculations such as *ab initio* or molecular dynamics, it is advisable to report the same coordinates as those of the supercell. This is especially important when the elastic-dipole matrix is provided, as this matrix changes with the orientation of the configuration.

Note that only symmetry unique interactions are required: if the file contains two or more interactions that are symmetrically equivalent, they will be double counted. The numerical values actually used in the calculations can be checked in the *num_conf.txt* and *num_freq.txt* files. The output file of the numerical part of the code *kineclue_num.log* gives the number of symmetry equivalent interactions found, and the number of interactions found in each configuration/jump frequency of the cluster.

Configuration and jump frequency prefactors should include only vibrational, magnetic or electronic entropy contributions, as the configurational entropy is already taken into account in the code. It is good to keep in mind as well that in the code the binding free energy of a configuration is additive. For instance, for a three-component cluster containing three pair sub-configurations, the total binding energy of the cluster is computed as the sum of the three pair binding energies. See Sec. 5.3 for detailed information about how binding and saddle-point energies are assigned their numerical value.

Example Example of interaction model file in a FCC system (cf. Section 3). The first nearest-neighbor interaction has an attractive binding energy of 0.3 eV. The binding energy of the second nearest-neighbor one is 0.1 eV, is given in supercell coordinates, and comes with a non-null elastic dipole. The third entry describes a first nearest-neighbor jump of an isolated vacancy. The fourth one will be ignored, while the fifth one reports a slightly attractive third nearest-neighbor interaction. Note that the isolated vacancy jump entry will be ignored if the cluster under study is anything but a vacancy monomer.

```
& C o 2 1.0 0.3 # prefactor and binding energy
vac 0. 0. 0.
sub_solute 0.5 0.5 0.

# binding energy and elastic dipole
& C s 2 0.1 0.01 0.0 0.0 0.0 0.01 0.0 0.0 0.0 0.01
sub_solute 1. 0. 1.
vac 0. 0. 0.

& J s 2 0.6 # saddle-point energy
vac 0. 0. 0. > 1. 0. 0.
bulk 1. 0. 0. > 0. 0. 0.

#& C s 2 0.5 0.1 # this configuration is ignored
vac 0. 0. 0.
sub_solute 1. 0.5 0.5

& C o 2 2.0 -0.02 # prefactor and repulsive binding energy
vac 0. 0. 0.
sub_solute 1. 0.5 0.5
```

Default Without the INTERACTIONMODEL keyword, no interaction model is used, and the numerical values for binding energies and saddle-point energies are taken from the *configurations.txt* and *jump_frequencies.txt* files (or computed randomly if the RANDOM keyword is activated).

4.19 & SENSITIVITY

Description A study is performed to determine how sensitive transport coefficients are to jump frequencies, and identify the most relevant ones. To this end, derivatives of each matrix with respect to a set of jump frequencies are computed. The gradient of transport coefficient in the jump frequency space gives the direction of maximum variation, hence it allows to identify the most important jump frequencies. For the sensitivity analysis, the set of jump frequencies that are used can be reduced by the user. Be aware that this sensitivity analysis is local, and depends on the values of jump frequencies as they appear in the *jump_frequencies.txt* file, eventually modified by keywords RANDOM and INTERACTIONMODEL . The sensitivity analysis will be ignored if the PRECISION keyword is used.

Output For each temperature in TEMPERATURES and each strain in NUMSTRAIN , the sensitivity study creates a file *IF_Tx_Sy.dat* in a sub-folder of DIRECTORY called *SENSITIVITY* ("x" is the temperature, "y" is the strain, and the "IF" prefix stands for Important Frequencies). These files show the normalized values of the cluster transport coefficient gradient components (gradient with respect to jump frequencies). Each line corresponds to a partial derivative of cluster transport coefficients with respect to a given jump frequency, as specified in the first column of the file using the same numbering as in file *jump_frequencies.txt*. Then, each column corresponds to a transport coefficient with the format "Ls1s2_dir" for the species "s1" species "s2" component of the cluster Onsager matrix in direction "dir".

Format One integer, followed by an optional list of integers. The first integer is the number of directions where this study is performed. Three values are accepted: 1 (only CPG direction), 2 (CPG and NORMAL directions) and 3 (all three directions). The optional list of integer specifies jump frequency indexes (as they appear in the first column of the *jump_frequencies.txt* file). The jump frequencies in this list are considered as "blocked" which means that the derivative of cluster transport coefficients with respect to this particular jump frequency is not computed. Hence, this list allows users to "block" saddle-point energies that have already been computed by reliable methods, and perform the sensitivity study only on jump frequencies that have been estimated less precisely. It is interesting to first run a numerical calculation with the SENSITIVITY keyword to identify the two most important jump frequencies, and then use the BATCHCREATE and BATCH keywords to automatically compute transport coefficients for various possible values of these two frequencies and get a feeling of how much cluster transport coefficients vary. Because this is a local analysis, several iterations might be necessary to get to a point where cluster transport coefficients do not vary much with the remaining "unblocked" jump frequencies.

Example

```
& SENSITIVITY 1 12 4 6 8-10
& SENSITIVITY 2
```

With the first line in the numerical input, the code performs the sensitivity analysis in the CPG direction only. All derivatives of transport coefficients with respect to jump frequencies will be computed except for jump frequencies numbered 4, 6, 8, 9, 10 and 12 in the *jump_frequencies.txt* file. With the second line, sensitivity study is performed on both CPG and NORMAL directions (if the latter was given in the main input file) and partial derivatives of transport coefficients are computed with respect to all jump frequencies in the *jump_frequencies.txt* file.

Default No sensitivity study is performed

5 Miscellaneous (but important) information

5.1 What can I do if...

- **The code does not found any symmetry operations, or some of them are obviously missing**: this may be a periodic boundary condition issue, if you have BASIS atoms located near the edges of the box defined in CRYSTAL . You can edit the *kinepy.py* file and change the value of the *tol_pbc* parameter which is basically a tolerance for applying periodic boundary conditions; the "reference" box in which all atoms must fit is not taken with respect to the $(0,0,0)$ coordinate but rather with respect to the $-tol\_pbc * (1,1,1)$ coordinate (in supercell units). The default value is $tol\_pbc = 0.001$.

- **One of my diagonal transport coefficient or the mobility is negative**: this is most probably due to cluster finite-size effects and should disappear as you increase the KINETICRANGE , even though you might need a very large KINETICRANGE before all diagonal coefficients become positive again. This may occur when isolated atoms having a correlated trajectory are interacting with other atoms. The value at infinite KINETICRANGE should be the transport coefficient obtained for a calculation containing the isolated atom only.

5.2 Description of the numerical output files

These files contain numerical results from KineCluE and are named according to the OUTPUT keyword followed by _n where $n \epsilon \{0,1,2\}$ depending on the diffusion direction: 0 is the CPG direction; 1 is the NORMAL direction; 2 is the vector product between both of them. Note that outputs for directions 1 and 2 are not created unless the "ND" argument is provided in OUTOPTIONS . Each line of the output file corresponds to a pair (strain, temperature) and various column are given (numbering starts at 0): strain [no units]; temperature T [K]; $1000/T$ [/K]; cluster partition function Z [no units]; number of cluster configurations Z_0 [no units]; total dissociation frequency [/s]; cluster lifetime [s]; upper-triangular part of the cluster Onsager matrix, printed line by line, e.g. L_{00}, L_{01}, L_{02}, L_{11}, L_{12}, L_{22} for a 3-species cluster [units depend on the UNITS keyword]; slowest diagonal coefficient, characterizing the slowest species in the cluster and providing an upper bound for the cluster mobility [same units as transport coefficients]; detailed dissociation coefficients, that is one for each possible sub-clusters after dissociation [/s]. A sub-cluster configuration is denoted by series of m numbers (m being the number of species) separated by "|" symbols. For instance "001|210" is the dissociation of the cluster into a monomer of species 3 and a 3-body cluster containing two atoms of species 1 and 1 atom of species 2. Another example: "001|100|110" is the dissociation into a monomer of species 3, a monomer of species 1 and a pair of species 1 and 2.

If the MOB keyword was used, the slowest diagonal coefficient is removed and replaced by 3 columns namely the mobility [same units as transport coefficients]; the ratio between the mobility and the diagonal coefficient of the slowest species which may sometimes give some indication about convergence with respect to the kinetic radius [no units]; the mean-free path of the cluster considered as isolated [m].

If the OUTOPTIONS keyword is used, additional outputs files are created for each tag among "UC", "CF", "EX" and "DR". The first three columns are identical to the main numerical output file: strain, temperature and 1000 over temperature. Then come all the coefficients corresponding to each specific tag (see OUTOPTIONS) and the headers allow for easy identification of which column corresponds to which coefficients. Note that species are numbered starting from 0, such that L_{00} is the diagonal coefficient of the first species.

5.3 From analytical to numerical jump frequencies

The analytical part of the code produces mainly three arrays of symbolic expressions from which cluster transport coefficients are computed. In order to do that, numerical values must be assigned to each and every symbolic variables, representing either stable configurations or saddle-point energies. In KineCluE, a saddle-point configuration represents a jump between two stable configurations. The assignment of numerical values is done in the numerical part of the code and relies on files *configurations.txt* and *jump_frequencies.txt* as well as parameters defined in the numerical input file.

The *configurations.txt* file contains one line for each stable configuration found by the analytical part of the code, and each line has two pieces of information that users can change: the prefactor (second column, no units because it is the exponential of entropic contributions to the free energy, apart from the configurational entropy contribution which is automatically taken care of in the code) and the binding energy (third column, in eV, positive means attraction). The numerical part of the code goes through each line of this file, one at a time and first looks at the value of the prefactor:

- if prefactor $<= 0$ and $\neq -1$: this configuration is removed from the calculation, as well as all jumps that have this configuration as initial or final configuration. Note that prefactors should not be set lower than 0 because it is the exponential of some entropy contributions (except for the default -1 value, which is a tag used by the code);
- if prefactor > 0: the energetics of this configuration are set according to the data in the *configurations.txt* file: the prefactor and binding energy are chosen as the values supplied by the user for this particular configuration. The prefactor should contain vibrational, electronic and magnetic entropy contributions, if any, the configurational entropy contribution being already taken care of automatically by the code. Binding energies are all defined with respect to a single implicit reference configuration, the choice of which is left to the user with the condition that this reference configuration should contain the same type and number of components as the cluster under study. Note that users cannot supply elastic dipoles for each configuration using the *configurations.txt* file;
- if prefactor = -1–which is the default value–one of three situations can arise:
  - if the INTERACTIONMODEL keyword was supplied the binding energy is set to 0 and the prefactor is set to 1. Then the code goes through all configurations supplied in the INTERACTIONMODEL file–as well as all their symmetry equivalent configurations–and for each configuration or sub-configuration that matches the one under study, the prefactor is multiplied by the associated value, and the corresponding binding energy is added. Note that the binding energies supplied in the INTERACTIONMODEL file may contain the elastic dipole contribution. As an example, let us consider a 3-body component, its binding energy will be the sum of the 3-body interaction energy and the 3 possible 2-body interaction energies, provided all these configurations and sub-configurations are part of the INTERACTIONMODEL file. Similarly, the prefactor will be the product of all 4 prefactors;
  - if no INTERACTIONMODEL keyword was supplied but the RANDOM keyword appears in the numerical input file, the prefactor is set to 1 while the binding energy is chosen at random between the two bounds associated with the RANDOM keyword;

  - if none of INTERACTIONMODEL and RANDOM keywords are used, the configuration is set to its default value: prefactor set to 1 and binding energy set to 0.

The numerical values set by the code can be found in the *num_ conf.txt* file. In case these values are temperature and/or strain dependent, they are given at zero strain and at the temperature halfway between minimum and maximum temperatures (TEMPERATURES keyword). In any case, an effective binding energy E_i^{eff} is computed for each configuration i as a function of strain ε and temperature T and used later on to apply the KRA model (see KRAACTIVATION):

$$E_i^{eff}\left(T,\varepsilon\right) = k_B T \ln\left(p_i \exp\left(\frac{E_i\left(\varepsilon\right)}{k_B T}\right)\right), \tag{3}$$

where k_B is the Boltzmann constant, p_i is the prefactor and E_i the binding energy of configuration i.

Once all configurations are set up with a numerical value, the code applies a similar procedure for each jump frequency–characterized by the prefactor and energy of the saddle-point configuration–but this time it uses the *jump_frequencies.txt* file. Again the prefactor is given in the second column and the saddle-point energy in the third column. Be careful, for saddle-point energies, a positive value means repulsion with respect to the aforementioned reference configuration (whereas for binding energies, positive means attraction).

- if prefactor $<=0$ and $\neq -1$: this jump frequency is removed from the calculation. Note that prefactors should not be set lower than 0 because it is the exponential of some entropy contributions for the saddle-point configuration (except for the default -1 value, which is a tag used by the code);
- if prefactor >0: the energetics of this saddle-point are set according to the data in the *jump_frequencies.txt* file: the prefactor and saddle-point energy are chosen as the values supplied by the user for this particular jump frequency. The prefactor should contain vibrational, electronic and magnetic entropy contributions, if any, the configurational entropy contribution being already taken care of automatically by the code. Saddle-point energies are defined with respect to the same reference state as binding energies. Note that users cannot supply elastic dipoles for each saddle-point configuration using the *jump_frequencies.txt* file.
- if prefactor $=$ -1–which is the default value–one of three situations can arise:
  - if either the initial or final configuration associated with this jump (columns 4 and 5 of the *jump_frequencies.txt* file) was set with a zero prefactor, this jump will also be set with a zero prefactor, meaning removed from the calculation;
  - if the file associated with the INTERACTIONMODEL keyword does not provide this particular jumps, the KRA model is applied, using the parameters from the KRAACTIVATION keyword.The initial and final state energies are taken as the effective binding energies of each configuration. The activation energy Q_{ij}^{eff} is taken as:

$$Q_{ij}^{eff}\left(T,\varepsilon\right) = k_B T \ln\left(p_{ij} \exp\left(\frac{Q_{ij}\left(\varepsilon\right)}{k_B T}\right)\right), \tag{4}$$

    where p_{ij} and Q_{ij} are the prefactor and activation energy given in the KRAACTIVATION keyword for the jump mechanism under study. If an elastic dipole was given for this jump mechanism, Q_{ij} will be a function of strain. The prefactor is thus set to 1 while the saddle-point energy E_{ij}^{sp} is set to:

$$E_{ij}^{sp}\left(T,\varepsilon\right) = Q_{ij}^{eff}\left(T,\varepsilon\right) - \frac{1}{2}\left(E_j^{eff}\left(T,\varepsilon\right) + E_j^{eff}\left(T,\varepsilon\right)\right). \tag{5}$$

  - if the file associated with the INTERACTIONMODEL keyword exists and contains this jump, then the prefactor, saddle-point energy and eventually elastic dipole are set to the values in the INTERACTION-MODEL file. Note that sub-jumps–i.e. jumps that do not constrain all the components of the cluster–are not considered. In the case where two jumps in the INTERACTIONMODEL file would correspond to the one under study, only the prefactor and saddle-point energy values from the second one are taken into account.

Whichever way numerical values are assigned to jump frequencies, they are all multiplied by a common value a^2pu where a is the lattice parameter (LATTPARAM keyword), p the global prefactor (PREFACTOR keyword) and u is a units conversion factor (UNITS keyword). The numerical values set by the code can be found in the *num_freq.txt* file. In case these values are temperature and/or strain dependent, they are given at no strain and at the temperature halfway between minimum and maximum temperatures (TEMPERATURES keyword).

5.4 From cluster transport coefficients to transport coefficients and how to select a relevant kinetic radius

KineCluE aims at computing cluster transport coefficients. Each cluster is characterized by a length–the kinetic radius–and all cluster-related properties such as partition function, transport coefficients, mobility and dissociation coefficients depend on the choice of the kinetic radius. Users are strongly advised to take some time to look at how these properties vary with the kinetic radius and set it with a meaningful value for the system under study. It is important to keep in my mind that the separation of a system in well-defined and non-overlapping cluster regions is suitable to compute transport properties for dilute systems but it remains arbitrary. This decomposition relies on the following hypothesis: each cluster is able to reach local equilibrium between all of its internal configurations before interacting with another cluster, which is basically saying that the system must be sufficiently dilute. The choice of the kinetic radius value is thus a compromise between: having a large enough radius to ensure that all significant kinetic trajectories are included in the cluster (i.e. for instance, once a pair dissociates, both defects should behave nearly as if they were monomers and not continue interacting); having a small enough radius which serves two purposes: first the system should be dilute, and the concentration at which this hypothesis breaks down decreases when cluster's size increase; second, if clusters are very large then cluster's properties include contributions from all sub-clusters (e.g. monomer contributions in the coefficients associated to with the pair).

In a nutshell, the appropriate kinetic radius is the smallest value leading to converged $Z\left(\alpha\beta\right)L_{\alpha\beta}\left(\alpha\beta\right)$, where $Z\left(\alpha\beta\right)$ is the partition function of the $\alpha\beta$ pair cluster and $L_{\alpha\beta}\left(\alpha\beta\right)$ is the off-diagonal cluster transport coefficient for the same cluster, both quantities being computed by KineCluE. α and β are different chemical species and/or defects. Determining meaningful kinetic radius values for mono-species systems is more tricky. To understand how we arrive at this methodology, we have to take a detailed look at how transport coefficients–which characterize the full system–are obtained from cluster transport coefficients:

$$L_{\alpha\beta} = \sum_c [c] L_{\alpha\beta}(c), \tag{6}$$

where $L_{\alpha\beta}$ is the transport coefficient for the full system, relating the flux of species α to a chemical potential of species β, $L_{\alpha\beta}(c)$ is the cluster transport coefficient for cluster c and $[c]$ is the concentration of cluster c per unit volume (if $L_{\alpha\beta}(c)$ is given in m^2/s). Cluster concentrations can be computed for instance in a low-temperature expansion formalism [cf. T. Schuler *et al.*, Physical Review Letters **115** 015501 (2015) and references therein]. Assuming that the system is sufficiently dilute and obeys local equilibrium:

$$[\alpha]_{tot} = \sum_c n_c^\alpha [c] = \sum_c n_c^\alpha Z\left(c\right) \prod_\beta [\beta]^{n_c^\beta}, \tag{7}$$

where n_c^α is the number of atoms or defects of species α in cluster c, $Z\left(c\right)$ is the partition function of cluster c and the product runs over all species β in the system. $[\alpha]_{tot}$ is the total concentration of atoms of species α while $[\alpha]$ is the concentration of isolated atoms of species α. With one such equation for each species in the system, a non-linear system of equations with unknowns $[\alpha]$ can be solved in the canonical ensemble where total species concentrations are known. In the grand-canonical ensemble where species chemical potentials μ_α are known, it is straightforward to compute total species concentrations and cluster concentrations since $[\alpha]$ is proportional to $\exp\left(\mu_\alpha/k_BT\right)$.

For larger concentrations (on the order of 1 at% even though this number is temperature and system dependent) a first order correction can be added to the monomer concentration to take into account (partially) cluster overlap issues, using the so-called counter-terms in low-temperature expansions:

$$[\alpha]_{tot} = [\alpha]\left(1 - \sum_{c_2} n_c^\alpha Z_0\left(c\right) \prod_\beta [\beta]^{n_c^\beta}\right) + \sum_{c_{2+}} n_c^\alpha Z\left(c\right) \prod_\beta [\beta]^{n_c^\beta}, \tag{8}$$

where the first term of the right-hand side is the α monomer concentration, the sum over c_2 runs over pair clusters and the sum over c_{2+} runs over pairs and larger clusters and $Z_0\left(c\right)$ is computed by KineCluE as the number of configurations of cluster c. This correction is not exact and is only a first-order correction. Thus, it is not intended to perform calculations for more concentrated systems where the cluster transport coefficients formalism becomes questionable, but rather to set a concentration boundary below which the dilute solid solution hypothesis is rather safe. Users are encouraged to check that $1 \gg \sum_{c_2} n_c^\alpha Z_0\left(c\right) \prod_\beta [\beta]^{n_c^\beta}$ to ensure the validity of the dilute hypothesis.

Now let us come back to the choice of the kinetic radius. Total transport coefficients are well-known physically meaningful quantities. In an infinitely dilute system containing only monomers and pairs, Eq. 6 tels us that the off-diagonal coefficient comes exclusively from the pair cluster, because there cannot be flux coupling between two species in a cluster containing only one species. So the off-diagonal coefficient in the total Onsager matrix is reduced to the off-diagonal contribution in the pair cluster containing species α and β. Thus, the latter contribution–$[\alpha][\beta]Z\left(\alpha\beta\right)L_{\alpha\beta}(\alpha\beta)$–should converge towards a well-defined physical quantity. The $[\alpha][\beta]$ product being a function of chemical potentials only, it depends on conditions external to the system under study and the total off-diagonal transport coefficients are proportional to this product. Hence $Z\left(\alpha\beta\right)L_{\alpha\beta}(\alpha\beta)$ is the quantity that must converge and this is why it is used to choose a relevant kinetic radius. Depending on the target application a threshold should be set by the user, for instance 5 % in relative error. Then, using the KIRALOOP keyword, one can compute the quantity $Z\left(\alpha\beta\right)L_{\alpha\beta}(\alpha\beta)$ for various kinetic radii and pick the smallest value where it drops below the chosen threshold.

Once the kinetic radius is set for the pair it should theoretically be kept identical for larger cluster. If not, strange configurations can arise, for instance a three-body cluster where no pair sub-cluster are within the pair kinetic radius. Conversely, in the case where the triplet kinetic radius is larger than the pair kinetic radius, one can imagine 3 components interacting as three pairs (i.e. all pair distances are below the pair kinetic radius) while non-interacting as a 3-body cluster. Hence it is simpler and more physical to keep the same kinetic radius for all clusters.