Manual for Self-Evolving Atomistic Kinetic Monte Carlo (SEAKMC) Python Package

Tao Liang and Haixuan Xu
December 2022

Copyright © 2022 by Tao Liang and Haixuan Xu.
All rights reserved.

TABLE OF CONTENTS

INTRODUCTION AND GENERAL INFORMATION **13**

Flowchart of SEAKMC_py *14*
  System inputs 15
  Loading Restart/DB 15
  MD/Relaxation 15
  Finding Defects 15
  Charactering AV 15
  Preloading SPs 15
  SNC 15
  SP Search 15
  AV Relaxation 15
  Validating SPs 15
  Post-processing SPs 15
  Recycling 15
  Energy Recalibration 15
  KMC (LEB) 15
  System Relaxation 15

Parallelization Paradigm of SEAKMC_py *16*

Code Availability *18*

Citing *18*

INSTALLATION **19**

INPUT **21**

system *21*
  TempFiles 21
  Inteval4ShowProgress 21
  Restart 21
    Reset_Simulation_Time 21
    LoadRestart 21
    LoadFile 22
    WriteRestart 22
    AVStep4Restart 22
    KMCStep4Restart 22
  Tolerance 22
  angle_tolerance 22
  significant_figures 22
  float_precision 22
  VerySmallNumber 22

kinetic_MC *22*
  NSteps 22

Temp 22
Temp4Time 22
DispStyle 22
Sorting 22
AccStyle 23
EnCut4Transient 23
NMaxBasin 23
Handle_no_Backward 23
Tol4Disp 23
Tol4Barr 23

defect_bank *24*
LoadDB 24
Preload 24
LoadPath 24
Ratio4DispLoad 24
SortDisps 24
Recycle 24
SaveDB 24
SavePath 24
UseSymm 24
FileHeader 24
NMax4DB 24
NMin4DB 24
Tol4Disp 24
Scaling 25
IgnoreType 25

potential *25*
species 25
pair_style 25
FileName 25
Path2Pot 25
pair_coeff 25
masses 25
bondlengths 26
cutneighs 26
coordnums 26
cutneighmax 26
bondlengths4LAS 26
coordnums4LAS 26
OpenKIM 26
    OpenKIM 26
    kim_init 26
    kim_interaction 26
    kim_param 26

force_evaluator *27*
Bin 27
Path2Bin 27
Style 27
nproc 27

- Nproc4Recal 28
- NSteps4Relax 28
- timestep 28
- processors 28
- partition 28
- Screen 28
- LogFile 28
- RinputOpt 28
- RinputMD0 28
- ImportValue4RinputOpt 28
- Keys4ImportValue4RinputOpt 28
- OutFileHeaders 29
- Relaxation 29
  - BoxRelax 29
  - InitTemp4Opt 29
  - TargetTemp4NVT 29
  - NVTSteps4Opt 29

data *30*
- FileName 30
- atom_style 30
- Relaxed 30
- BoxRelax 30
- RinputOpt 30
- MoleDyn 30
- RinputMD 30
- RinputMD0 30
- boundary 31
- units 31
- dimension 31

active_volume *31*
- Style 33
- NActive 34
- NBuffer 34
- NFixed 34
- NPredef 34
- PredefOnly 34
- RT_SetMolID 34
- DefectCenter4RT_SetMolID 34
- R4RT_SetMolID 34
- FCT4RT_SetMolID 34
- cutdefectmax 35
- FindDefects 35
  - Method 35
  - Defects 35
  - ReferenceData 35
  - atom_style4Ref 35
  - DCut4Def 35
- PDReduction 36
- SortD4PDR 36

DCut4PDR 36
RecursiveRed 36
Overlapping 37
Stack4noOverlap 37
DCut4noOverlap 37
Order4Recursive4PDR 37
Order4Recursive4AV 37
Overlap4OrderRecursive 37
NMax4Def 37
DActive 38
DBuffer 38
DFixed 38
NMax4AV 38
NMin4AV 38
Sorting 38
Sort_by 38
SortingSpacer 38
SortingShift 38
SortingBuffer 38
SortingFixed 38
PointGroupSymm 39
NMax4PG 39
TurnoffPBC 39

dynamic_matrix *39*
SNC 39
CalPrefactor 39
NMax4SNC 39
FileName 39
displacement 39
delimiter 39
LowerHalfMat 39
Method4Prefactor 40
VibCut 40

spsearch *40*
Method 48
NSearch 48
SearchBuffer 49
NMax4Trans 49
TrialStepsize 49
MaxStepsize 49
RatioStepsize 49
DecayStyle 49
DecayRate 49
DecaySteps 49
MinStepsize 49
DimerSep 49
TransHorizon 49
En4TransHorizon 50
CheckAng 50

CheckAngSteps 50
AngCut 50
IgnoreSteps 50
NMax4Rot 50
FThres4Rot 50
FMin4Rot 50
FConv 50
EnConv 50
DRatio4Relax 50
Tol4Connect 50
R2Dmax4SPAtom 50
DCut4SPAtom 50
DynCut4SPAtom 50
ActiveOnly4SPConfig 51
ShowIterationResults 51
Inteval4ShowIterationResults 51
ShowVN4ShowIterationResults 51
ShowCoords4ShowIterationResults 51
insituGuidedSPS 52
LocalRelax 52
    LocalRelax 52
    InitTemp4Opt 52
    TargetTemp4NVT 52
    NVTSteps4Opt 52
Preloading 52
    Preload 52
    LoadPath 52
    Method 52
    Ratio4DispLoad 53
    Scaling 53
    SortDisps 53
    FileHeader 53
    CheckSequence 53
    FileHeader4Data 53
HandleVN 54
    CheckAng4Init 54
    AngTol4Init 54
    MaxIter4Init 54
    NMaxRandVN 54
    ResetVN04Preload 54
    RatioVN04Preload 54
    IgnoreSteps 54
    CenterVN 54
    NSteps4CenterVN 54
    RescaleVN 55
    RescaleValue 55
    Int4ComputeScale 55
    TakeMin4MixedRescales 55
    RescaleStyle4LOGV 55
    Period4MA 55

    - XRange4LOGV 55
    - Ratio4Zero4LOGV 55
    - MinValue4LOGV 55
    - MinSpan4LOGV 55
    - PowerOnV 55
    - RescaleStyle4RAS 55
    - XRange4RAS 56
    - Ratio4Zero4RAS 56
    - MinSpan4RAS 56
- force_evaluator 56
    - nproc 56
    - processors 56
    - partition 56
    - Rinput 56
    - RinputOpt 56
    - RinputDM 56
- FixTypes 57
- FixAxesStr 57
- TaskDist 58
- Master_Slave 58

saddle_point 58
- BarrierCut 59
- BarrierMin 59
- EbiasCut 59
- EbiasMin 59
- BackBarrierMin 60
- Prefactor 60
- CalBarrsInData 60
- CalEbiasInData 60
- Thres4Recalib 60
- DAtomCut 60
- DMagCut 60
- DMagMin 60
- DtotCut 60
- DtotMin 60
- DmaxCut 60
- DmaxMin 60
- DsumCut 60
- DsumMin 61
- DsumrCut 61
- DsumrMin 61
- DMagCut_FI 61
- DmagMin_FI 61
- DtotCut_FI 61
- DtotMin_FI 61
- DmaxCut_FI 61
- DmaxMin_FI 61
- DsumCut_FI 61
- DsumMin_FI 61
- DsumrCut_FI 61

- DsumrMin_FI ... 61
- DMagCut_FS ... 61
- DmagMin_FS ... 61
- DtotCut_FS ... 61
- DtotMin_FS ... 61
- DmaxCut_FS ... 62
- DmaxMin_FS ... 62
- DsumCut_FS ... 62
- DsumMin_FS ... 62
- DsumrCut_FS ... 62
- DsumrMin_FS ... 62
- ValidSPs ... 62
  - RealtimeValid ... 62
  - RealtimeDelete ... 62
  - CheckConnectivity ... 62
  - NScreenDisp ... 62
  - NScreenEng ... 62
  - toScreenDisp ... 63
  - toScreenEng ... 63
  - AND4ScreenDE ... 63
  - ScreenDisp ... 63
    - Str4ScreenD ... 63
    - Type4ScreenD ... 63
    - AbsVal4ScreenD ... 63
    - MinVal4ScreenD ... 63
    - MaxVal4ScreenD ... 63
    - AND4ScreenD ... 63
  - ScreenEng ... 63
    - Type4ScreenE ... 63
    - MinVal4ScreeE ... 63
    - MaxVal4ScreenE ... 63
    - AND4ScreenE ... 63
  - MaxRatio4Dmag ... 66
  - MaxRatio4Barr ... 66
  - GroupSP ... 66
  - AngCut4GSP ... 66
  - MagCut4GSP ... 66
  - EnCut4GSP ... 66
  - EnTol4AVSP ... 66
  - Tol4AVSP ... 66
  - FindSPType ... 66
  - AngCut4Type ... 66
  - MagCut4Type ... 67
  - LenCut4Type ... 67
  - EnCut4Type ... 67
  - NMax4Dup ... 67
  - NCommonMin ... 67
  - R2Dmax4Tol ... 67
  - Tol4Disp ... 67

visual ... 68

Screen 68
Log 68
Write_SP_Summary 68
Write_Data_SPs 68
  Write_Data_SPs 68
  Write_KMC_Data 68
  Write_Data_AVs 68
  Write_Prob 68
  DetailOut 68
  SPs4Detail 68
  DispStyle4DataSP 68
  OutputStyle 68
  Sel_iSPs 68
  Offset 69
  DataOutPath 69
  SPOutPath 69
Write_AV_SPs 69
  Write_AV_SPs 69
  Write_Local_AV 69
  Write_Data_AV_SPs 69
  AVOutPath 69
  DispStyle4AVSP 69
RCut4Vis 69
DCut4Vis 69
Invisible 70
Reset_Index 70
ShowBuffer 70
ShowFixed 70

OUTPUT 71

Seakmc.log 71

Seakmc_summary.csv 72

RESTART_istep_idav.restart 73

SPOut 73
KMC_istep_SPs.csv 73
KMC_istep_Prob.csv 74
KMC_istep_Deleted_SPs.csv 74
KMC_istep_DetailSPs.csv 74

DataOut 75
KMC_istep.dat 75
KMC_istep_Data_AVs.dat 75
KMC_istep_Data_SP_offset+isp.dat 75
KMC_istep_Data_FI_offset+isp.dat 75
KMC_istep_Data_Selected_SPs.dat 75
KMC_istep_Data_Selected_FIs.dat 76

DefectBank 76
DB_basin_ibasin_sch.csv 76

DB_b_ibasin_idisp.csv 76

AVOut 76
KMC_istep_Data_AV_idav.dat 76
KMC_istep_AV_idav_SPs.dat 76
KMC_istep_AV_idav_FIs.dat 77
KMC_istep_Data_AV_idav_SPs.dat 77
KMC_istep_Data_AV_idav_FIs.dat 77

ITERATION_RESULTS 77
summary_array.csv 77
VectorN_TranslationStep_Iteration.csv 78
Coords_TranslationStep_Iteration.csv 78

EXAMPLES 79

A vacancy in Fe bcc structure (Fe_vacancy) 79

Guided SPS (GuidedSPS) 79
Generating the database of the saddle points (GenerateSPs) 79
SPS with the database of the saddle points (GuidedSPS) 79

SPS on selected defects (SelectedDefects) 80

Import outputs from SEAKMC_py (ImportOutput) 80

Use Script File "calllammps" (Use_calllammps) 81

REFERENCES 83

INDEX 84

LIST OF FIGURES

Figure 1.1. Flowchart of SEAKMC_py 14
Figure 1.2. Overview of parallelization of SEAKMC_py 16
Figure 1.3. Schematic view of parallization with the master-slave scheme 17
Figure 3.1. Equilibrium bond length (EBL) and energy per bond (EPB) as a function of coordination number (CN) 32
Figure 3.2. Schematic view of constructing active volume 33
Figure 3.3. a) Saddle point search on the potential energy surface; b) Schematic view of a dimer 40
Figure 3.4. The energy and displacements during a dimer search of a 168-atom AV with a vacancy at the center of an Fe BCC system. a) The energy profile as a function of dimer translation steps; b) Atomic displacements vs atom sequence of the saddle point. 41
Figure 3.5. The energy and translation displacements (Trans. disp.) during a dimer search of a 964-atom AV with a vacancy at the center of an Fe BCC system. a) The energy profile and translation displacement as a function of dimer translation steps; b) The energy profile and translation displacement as a function of dimer translation steps with centering VN (CenterVN) and centering VN+translating dimer horizontally (CenterVN + TransHor) center. 42
Figure 3.6. The energy and VN during dimer search of a 964-atom AV with a vacancy at the center of an Fe BCC system. a) The energy profile with SNC [6]; b) The atomic VN at early stage in dimer search. Top view is the dimer method with the SNC method and bottom view is the original dimer method. 44
Figure 3.7. The rescaling factors and VN after applying the rescaling factors. a) The rescaling factors with four different input values; b) The snapshots of the atomic VN at early stage in dimer search after applying the rescaling factors. 46
Figure 3.8. The energy profile of the dimer search with the dynamic boundary for different rescaling inputs. a) Rescale on single input value; b) Rescale on multiple input values. 48
Figure 3.9. A schematic view of a saddle point (SP) between the initial and final states 58
Figure 4.1. First part of the Seakmc.log file 71
Figure 4.2. Second part of the Seakmc.log file 72
Figure 4.3. Third part of the Seakmc.log file 72

INTRODUCTION AND GENERAL INFORMATION

Atomistic simulations are widely used in predicting the evolution of structures and understanding the properties and performance of materials. For instance, molecular dynamics (MD) is employed to investigate the atomistic processes and thermodynamic properties of materials. However, the accessible time scale in MD is generally limited to nanoseconds due to the very small timestep, which is normally required below the 1 femtosecond. Several approaches, including kinetic Monte Carlo (KMC) are developed to extend the time scale while maintaining the framework of MD.

KMC is a stochastic method, which uses a catalog of pre-specified barriers to compute the rate of exiting the current local minimum. A move is then selected according to the Boltzmann distribution of the barrier catalog, and a dynamical trajectory associated with the selected move is constructed. Since the time steps in KMC are dominated by the lowest energy barrier of the catalog, the structure evolves along the minimum energy pathway (MEP) of its potential energy surface (PES). Depending on the barrier catalog, KMC simulations can reach the experimental time scales and beyond.

The key issue within KMC is to construct the energy barrier catalog without the knowledge of the PES and the final state would be. Over the years, a number of algorithms based on the off-lattice system have been proposed to find the saddle points on a real-time basis, among which the dimer method [1] [2], the kinetic activation-relaxation technique (k-ART) [3], and the self-evolving atomistic kinetic Monte Carlo (SEAKMC) [4] [5] are the most successfully developed techniques in the community.

SEAKMC was first developed by Xu *et al*. in 2011 [4]. Since the saddle points are often involved in localized atom rearrangement around a defect, Xu *et al* has introduced the concept of the active volume (AV). An AV is a portion of the system, which is often constructed according to the presence of the defects of interest in the system. The boundary of each AV is set to such that the increase size of the AV has negligible effects on the following saddle point search (SPS). The system is then separated into several AVs, where each is regarded as an independent sub-system. The following SPS, which is often the most time-consuming part, only performs on these AVs of a system. With the AV concept, SEAKMC has significantly increased the efficiency of SPS.

SEAKMC_py is a Python software package based on the concept of the AV and the dimer method coded in SEAKMC. It includes the recent developments of scale normal coordinates (SNC) [6], angle check in parallel SPS [7], and mean rate method (MRM) [8] [9] to handle the low energy barrier (LEB). In addition, SEAKMC_py has implemented the point group (PG) symmetry operations in saddle point sampling, prefactor calculations based on the harmonic theory [10]. Moreover, SEAKMC_py allows loading the defect bank (DB) from an external database, recycled saddle points from the previous KMC steps, and the customer defined saddle points to guide the SPS. More importantly, SEAKMC_py has first introduced a dynamic and automated method to identify the defects on the fly and the dynamic active volume (DAV) for SPS. The DAV has further reduced the dimensionality of the system by deactivating the motion of the unimportant atoms at the elevation stage of a searching process. This DAV can improve the efficiency of the SPS in both time cost and the probability of finding relevant saddle points, which is essential for sampling the potential energy landscape and building the event table as efficient as possible and as complete as possible for KMC simulations.

SEAKMC_py itself does NOT include any energy or force evaluator. The python wrapper of LAMMPS (*pyLAMMPS*) is fully implemented in SEAKMC_py and it is easy to extend to include other energy force evaluators.

SEAKMC_py has a single file named ***input.yaml*** for system inputs. The details of ***input.yaml*** are elaborated in Section INPUT.

The outputs of SEAKMC_py include ***Seakmc.log***, ***Seakmc_summary.csv***, the restart files, the information of saddle points in *SPOut*, the structure information in *DataOut*, the information of defect bank in *DefectBank*, and the structure information of active volumes in *AVOut*. The details are available at Section OUTPUT.

Flowchart of SEAKMC_py

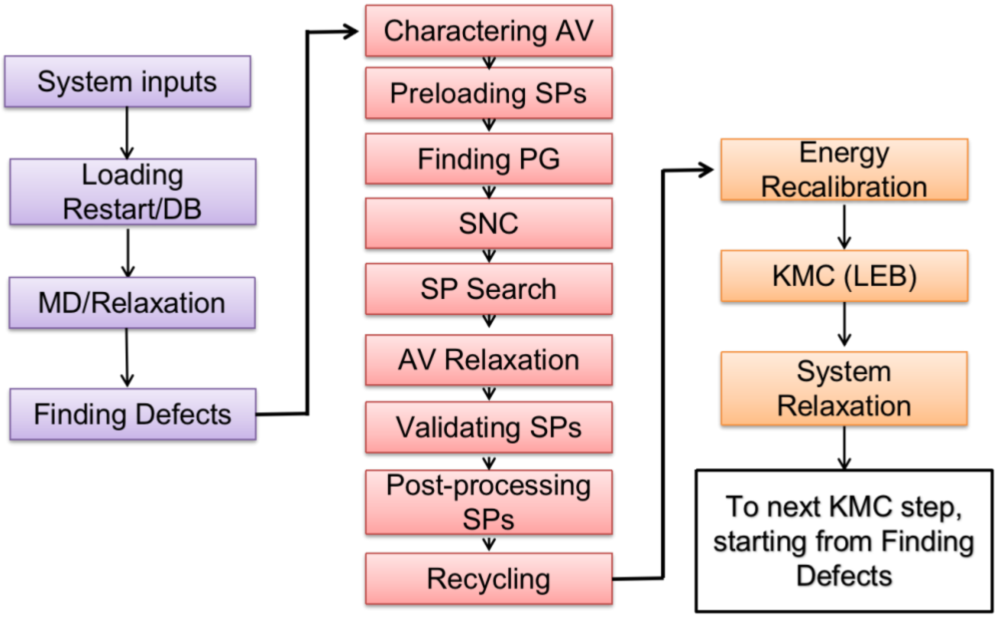

System inputs
Loading Restart/DB
MD/Relaxation
Finding Defects
Charactering AV
Preloading SPs
Finding PG
SNC
SP Search
AV Relaxation
Validating SPs
Post-processing SPs
Recycling
Energy Recalibration
KMC (LEB)
System Relaxation
To next KMC step, starting from Finding Defects

Figure 1.1. Flowchart of SEAKMC_py

A flowchart of SEAKMC_py is shown in Figure 1.1. After loading the system inputs, restart file, and defect bank, the MD simulation and relaxation is performed on the initial data structure if needed. The procedure of finding defects, charactering AV, pre-processing of AVs, SPS on each AV, post-processing before KMC, KMC, and system relaxation repeats until finished with the predefined KMC steps. The working object is the whole system (data structure) in cyan and orange boxes and is an AV (a sub-system) in pink boxes. The details of work in each box are given below.

System inputs

Loading ***input.yaml***

Loading Restart/DB

Loading restart and defect bank

MD/Relaxation

MD simulation and relaxation on initial data structure

Finding Defects

Finding defects in the data structure

Charactering AV

Charactering AV based on each found defect

Preloading SPs

Preloading saddle points from defect bank, previously recycled saddle points and user defined saddle points for the AVs

SNC

Calculating and diagonalizing the dynamic matrix of the AVs for SNC

SP Search

Saddle point search of the AVs

AV Relaxation

Local relaxation of the AVs

Validating SPs

Deleting the duplicated SPs and removing the SPs based on the customer defined criteria

Post-processing SPs

Identifying the type of SPs and calculating the prefactors of SPs

Recycling

Recycling the found SPs and adding them to DB

Energy Recalibration

Recalibrating the barriers and bias in the whole data

KMC (LEB)

KMC process with LEB - constructing energy catalog, selecting of the event, and updating the trajectory

System Relaxation

Relaxation of the whole data structure

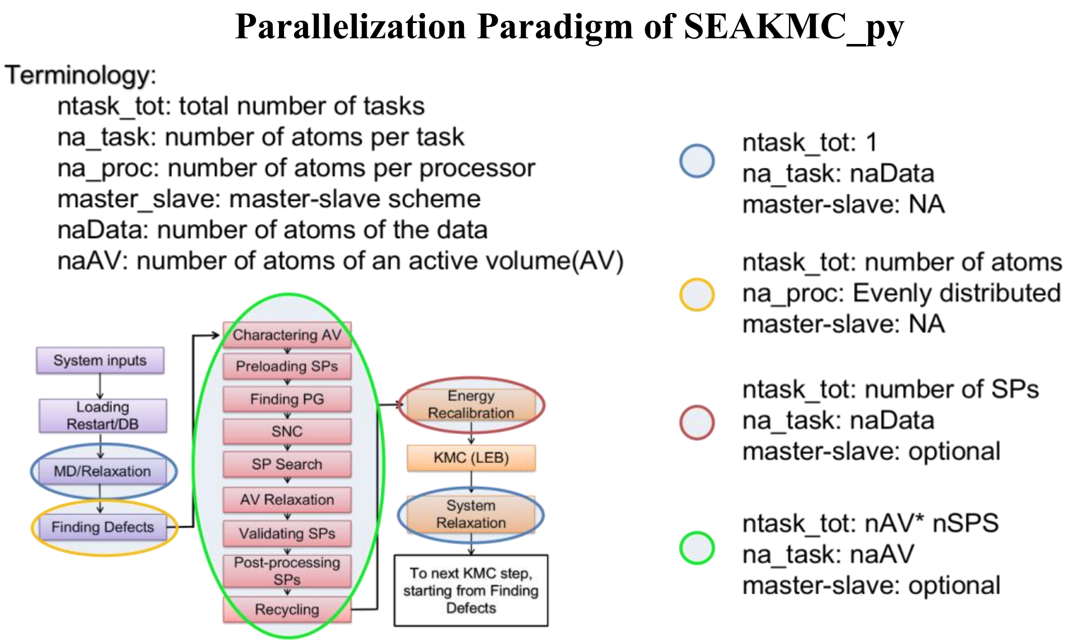

Parallelization Paradigm of SEAKMC_py
Terminology:
ntask_tot: total number of tasks
na_task: number of atoms per task
na_proc: number of atoms per processor
master_slave: master-slave scheme
naData: number of atoms of the data
naAV: number of atoms of an active volume(AV)
System inputs
Loading Restart/DB
MD/Relaxation
Finding Defects
Charactering AV
Preloading SPs
Finding PG
SNC
SP Search
AV Relaxation
Validating SPs
Post-processing SPs
Recycling
Energy Recalibration
KMC (LEB)
System Relaxation
To next KMC step, starting from Finding Defects
ntask_tot: 1
na_task: naData
master-slave: NA
ntask_tot: number of atoms
na_proc: Evenly distributed
master-slave: NA
ntask_tot: number of SPs
na_task: naData
master-slave: optional
ntask_tot: nAV* nSPS
na_task: naAV
master-slave: optional

Figure 1.2. Overview of parallelization of SEAKMC_py

SEAKMC_py is parallelized using *mpi4py* wrapper for the Message Passing Interface (MPI) standard. A parallelization paradigm for SEAKMC_py is given in Figure 1.2. Based on the nature of the workload, i.e. total number of tasks (*ntask_tot*), number of atoms per task (*na_task*), or number of atoms per processor (*na_proc*), there are four places (blue, yellow, green, and crimson ellipses) that require different ways to split the communicator. These four communicator configurations are given below.

1. In blue ellipses the working object is the whole data and only has one task. The master-slave scheme is not applicable. The number of processors per task in blue ellipses is user defined in *force_evaluator*.
2. In yellow ellipse the finding defects method is looped on each atom of the whole data. The master-slave scheme is not applicable. It uses all the available processors and has space decomposition of the data on processors, i.e. the atoms are evenly distributed on processors to perform finding defects.
3. In green ellipse the working object is (are) the AV(s). Assuming each AV has *n* SPS attempts (nSPS), the *ntask_tot* is nAV*nSPS. The master-slave scheme

is optional. The number of processors per task in green ellipse is user defined in *force_evaluator* under *spsearch*.

4. In crimson ellipse the working object is the whole data. The number of tasks is the number of saddle points. No master-slave scheme and the number of processors per task is user defined in *force_evaluator*.

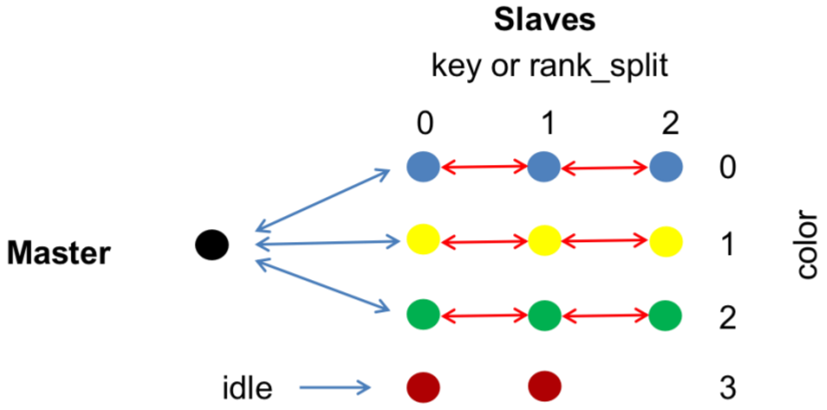


Figure 1.3. Schematic view of parallization with the master-slave scheme

The master-slave scheme for communicator splitting at Places 3 and 4 is optional. The parallization without the master-slave scheme is coded such that the program proceeds the next round of tasks only after all processors finished their working tasks. The waiting time will reduce the parallization efficiency, especially when the time costs of tasks are quite different. The master-slave scheme has a better control of task flow. An example of a master slave scheme with total 12 processors and 3 processors per task is shown in Figure 1.3. The processor which is ranked 0 at the communicator is spared as the master node. The rest of processors (slaves) are split to 4 groups. Each group is assigned a *color* and each processor is assigned a *key*, i.e. the rank at split communicator. The master node is used to distribute the tasks, communicate with *key* 0 processors, and process data on received results from slaves. Since the master node can continuously assign tasks to slaves until all tasks are completed, there is no waiting time, thus improving the efficiency. However, it may not make full use of the processors as exemplified by *color* 3 processors in Figure 1.3.

SEAKMC_py will generate *Runner_color* folder for each working task, where *color* refers to the group ID of processors (*color*=0 in the serial version).

Code Availability

The serial version: https://github.com/TaoLiang120/SEAKMC_py
The parallel version: https://github.com/TaoLiang120/SEAKMC_py_parallel

Citing

Please cite the reference below.

1. Tao Liang and Haixuan Xu, *Saddle point search with dynamic active volume*, Computational Materials Science **228**, 112354 (2023), DOI: https://doi.org/10.1016/j.commatsci.2023.112354

INSTALLATION

SEAKMC_py requires python 3.0 or above, *anaconda* 4.0 or above, *pymatgen* and *mpi4py*.

SEAKMC_py itself does NOT include any energy or force evaluator. The python wrapper of LAMMPS (*pyLAMMPS*) is fully implemented in SEAKMC_py. The software needed to compile LAMMPS are also required. Since *dynamic matrix* is involved in SEAKMC_py, installation of the *PHONON* package in LAMMPS is recommended.

For a custom installation of LAMMPS, please follow the instructions to install LAMMPS and its *Python wrapper* on the *LAMMPS* website. Make sure you can *run LAMMPS and Python*, as shown on the *link* on LAMMPS website, before you attempt to install of SEAKMC_py.

Below elaborates the installation of SEAKMC_py using *conda* in *anaconda* under a virtual environment—"seakmc_env".

1) Follow the instructions to install *anaconda*.
2) Create a virtual environment.
   ```
   > conda create -p path_to_seakmc_env/seakmc_env
   ```
3) Activate the seakmc_env.
   ```
   >conda activate path_to_seakmc_env/seakmc_env
   ```
4) Install LAMMPS using conda.
   ```
   seakmc_env>conda install lammps=*=*mpich*
   ```
   or
   ```
   seakmc_env>conda install lammps=*=*openmpi*
   ```
5) Install *OpenKIM* package:
   ```
   seakmc_env>conda install openkim-models
   ```
6) Install pymatgen.
   ```
   seakmc_env>conda install --channel conda-forge pymatgen
   ```
7) Install mpi4py (version <=3.1.6).
   ```
   seakmc_env>python -m pip install mpi4py==3.1.6
   ```
8) Install SEAKMC_py.
   Uncompress SEAKMC_py-date.tgz and change directory to SEAKMC_py to install serial version:
   ```
   seakmc_env>tar -xvf SEAKMC_py-date.tgz
   seakmc_env>cd SEAKMC_py
   seakmc_env>python setup.py install
   ```

   Uncompress SEAKMC_py_parallel-date.tgz and change directory to SEAKMC_py_parallel to install parallel version:
   ```
   seakmc_env>tar -xvf SEAKMC_py_parallel-date.tgz
   seakmc_env>cd SEAKMC_py_parallel
   seakmc_env>python setup.py install
   ```

Upon uncompression of *SEAKMC_py-date.tgz* or *SEAKMC_py_parallel-date.tgz*, the manual file is available at corresponding root directory (*SEAKMC_py* or *SEAKMC_py_parrallel*). The *run_script* folder includes three files:

1. *SPS_AV_0.json*: a sample input for *Preloading* with *Settings Method* in *spsearch*
2. *input.yaml*: a sample input file with all default values
3. *run_seakmc.py* or *run_seakmc_p.py*: A python code to execute SEAKMC_py serial or parallel versions

The *examples* are located at *examples* directory.

To execute SEAKMC_py serial version, copy run_seakmc.py to the working directory and input:

```
seakmc_env>python run_seakmc.py
```

To execute SEAKMC_py parallel version, copy run_seakmc_p.py to working directory and input:

```
seakmc_env>mpirun -np 4 python run_seakmc_p.py
```

If running on a cluster, customize your submission script based on the workload of the clusters and then submit the job.

INPUT

SEAKMC_py input is an object of Python class with 11 properties – *system*, *KMC*, *defect_bank*, *potential, force_evaluator*, *data*, *active_volume*, *dynamic_matrix, spsearch*, *saddlepoint*, and *visual*. Each property is a Python dictionary in JSON format. The filename containing SEAKMC_py settings is ***input.yaml***, a simple text file in YAML format. *PyYAML* is used to parse the ***input.yaml***.

Without specification, the unit in ***input.yaml*** is the *metal* unit in LAMMPS.

SEAKMC_py itself does NOT include any energy or force evaluator. The python wrapper of LAMMPS (*pyLAMMPS*) is the defaulted force evaluator in ***input.yaml***.

If the keywords in ***input.yaml*** are the keywords used in *LAMMPS* or *pymatgen*, they remain unchanged. Otherwise, the first letter of each sub-word in a keyword is capitalized. The options for keywords are case insensitive unless are otherwise specified.

system

TempFiles

Three temporary files are used by SEAKMC_py. They are fixed to ["tmp0.dat", "tmp1.dat", "tmp2.dat"], where "tmp0.dat" is the input data and "tmp1.dat" is the output data. If *pyLAMMPS* is the *force_evaluator*, the code will copy the structure file to *Runner_color/tmp0.dat* and write output data as *Runner_color/tmp1.dat*

Inteval4ShowProgress

Interval of tasks to show the progress, defaults to 10

For a large number of *task_tot* (nAV*nSPS), there might be no output for a long time in the parallel version. This parameter is used to show the progress of the SPS. For the master-slave scheme, the progress is output to the logfile when the remaining between task_index and *Inteval4ShowProgress**ntask_time is 0, where the task_index is the task index and the ntask_time is the number of tasks distributed at a time. Without master-slave scheme, the progress is output to the logfile when the remaining between task_index and *Inteval4ShowProgress* is 0, where the task_index is the index number of tasks on the master processor only. Another place that shows the progress is "*KMC_istep_Deleted_SPs.csv*" in *SPOut* folder.

Restart

If *WriteRestart* is True, it will write a restart file at every time finished *AVStep4Restart* AVs and *KMCStep4Restart* KMC steps. By default, it will load the latest restart file. To load a specific restart file, specify the filename (*LoadFile*) to load. You can restart the code within the same folder. The code will append the results to the previous results.

Reset_Simulation_Time

Reset simulation time to zero? Defaults to False

LoadRestart

Load restart file? Defaults to True

LoadFile

Restart file to be loaded? Defaults to False. If *LoadFile* is a string, it will load *LoadFile*. Otherwise, code which will load the latest restart file

WriteRestart

Write restart file? Defaults to True

AVStep4Restart

Interval of active volume steps for writing restart files. *AVStep* here means the number of finished active volumes, defaults to 1000

KMCStep4Restart

Interval of KMC steps for writing restart files, defaults to 1

Tolerance

General distance tolerance, defaults to 0.1 angstroms

angle_tolerance

General angle tolerance, defaults to 5 degrees

significant_figures

Significant figures for outputting atoms' coordinates and displacements, default to 6

float_precision

Float precision for outputs, defaults to 3

VerySmallNumber

A very small number that is added to division or logarithm math operations to avoid not a number (***nan***) error in code, defaults to 1.0e-20

kinetic_MC

NSteps

The number of kinetic Monte Carlo (KMC) steps, defaults to 1

Temp

Temperature in KMC, defaults to 800 K

Temp4Time

Temperature for KMC time step, defaults to *Temp* above.
If *Temp4Time* is different from *Temp*, the *Temp* is used to generate the probability distribution of the event table for selecting a given event in a KMC step. the *Temp4Time* is used in computing the KMC time step, where the prefactor is the mean value of prefactors of events in the event table.

DispStyle

Displacement of "*SP*" or "*FI*" will be updated for data relaxation between KMC steps, defaults to *FI*

Sorting

Sorting the SPs by rate, defaults to False

AccStyle

Style of acceleration method, defaults to "*NoAcc*"
The options for *AccStyle* are *NoAcc* and *MRM* [8] (only the first three letters matter).

EnCut4Transient

Max barriers (forward and backward barriers) of superbasin, defaults to 0.5

NMaxBasin

Max number of basins allowed in a superbasin, defaults to "*NA*"

Handle_no_Backward

If the backward saddle point is not found, options are "*STAYIN*" and "*OUT*", defaults to "*OUT*" (only the key words "OUT" and "IN" matter)

Tol4Disp

Tolerance of displacement between two connected basins, defaults to 0.1

Tol4Barr

Tolerance of barriers between two connected basins, defaults to 0.03

EnCut4Transient, *NMaxBasin*, *Handle_no_Backward*, *Tol4Disp*, and *Tol4Barr* are only used for mean rate method (*MRM*) [8]. *MRM* will explore basins and associated saddle points on the fly, but the local dynamics is missing. That is to say, the selection of events may come from one of the basins inside a superbasin rather than from the current basin. It will build a superbasin if both forward and backward barriers (see Figure 3.9) are smaller than *EnCut4Transient*. Since *MRM* needs to build the connection of basins between KMC steps, *LocalRelax* in *spsearch* must be True and no box relaxation is allowed between KMC steps. *Tol4Disp* and *Tol4Barr* are the tolerance to identify whether two basins of the superbasin are connected. If the maximum absolute component (MAXABSC) of the 3N (N is the number of atoms) dimensional displacement vector (coordinate difference between final state at one basin and initial state at the immediately connected basin) < *Tol4Disp* and the barrier difference between backward barrier at initial state and forward barrier at the final state < *Tol4Barr*, these two basins are connected by a saddle point. *MRM* requires to store the structure and energetic information until the stopping criteria has been met. *NMaxBasin* is used to avoid memory overflow. If *NMaxBasin* is an integer and the number of basins explored is larger than *NMaxBasin*, the code will force the current basin to leave the superbasin, release the memory, and have a fresh start of KMC step.

There are many cases that the code cannot find the saddle point that connects two basins from backward side. If *Handle_no_Backward* is "*OUT*", the code will force the current basin to move out of the superbasin. If "*IN*" is included in *Handle_no_Backward*, the code will assume the saddle point is found from the backward.

defect_bank

A defect bank is a Python object, including atom coordinates of basin structure and atomic displacements of the saddle points associated with the basin. There are two sources for a defect bank: 1) loading from previously saved files; 2) recycled from the previous KMC step. Inside code bank of defects is stored as a list of defect bank instances throughout the simulation.

LoadDB

Load defect bank (DB)? Defaults to False

Preload

Preloading displacements of saddle points, defaults to False. Please see the differences with *Preload* in *spsearch*.

LoadPath

Loading path, defaults to "DefectBank"

Ratio4DispLoad

Ratio for loading displacements, defaults to 0.8

SortDisps

Loading displacements by filename sequence? Defaults to False

Recycle

Recycling the defects and saddle points? Defaults to False

SaveDB

Save DB to files? Defaults to "DefectBank"
To save the recycled SPs to files, set *Recycle* to True.

SavePath

Saving path, defaults to "DefectBank"

UseSymm

Using the symmetry? Defaults to False
If *UseSymm* is True, only one displacement matrix from the same type of saddle points will be recycled and saved. Otherwise, displacement matrix of all saddle points in the list of defect bank will be recycled and saved.

FileHeader

The filename header for DB, defaults to "DB"
Filename for basin: *DB_basin_ibasin_sch.csv*. The "sch" is Schoenflies symbol of the detected point group
Filename for displacement: *DB_b_ibasin_disp_idisp.csv*

NMax4DB

Maximum number of atoms in DB, defaults to 100

NMin4DB

Minimum number of atoms in DB, defaults to 8

Tol4Disp

Tolerance of the MAXABSC for judging the same structures, defaults to 0.1

Scaling

Scaling factor applied on DB, defaults to 1.0
When loading from the defect bank, the coordinates will be multiplied by *Scaling*.
When saving to the defect bank, the coordinates will be divided by *Scaling*.

IgnoreType

Ignore the type of atoms, defaults to False

potential

If *OpenKIM* is installed in LAMMPS, use the *OpenKIM* below to specify the potentials.

The *bondlengths* entry default is the bond lengths in the *pymatgen* database with bond order = 1. The *coordnums* entry is defaulted to 4 of every atom type. The *bondlengths* and *coordnums* entries are two important inputs for finding defects and then constructing active volume (*FindDefects* in *active_volume*).

species

List of elements (chemical symbols) in *data*, equivalent to atom types in LAMMPS, one chemical symbol can have multiple entries of types, required, example:
- Fe
- Fe
- P

pair_style

Potential style, required for *pyLAMMPS* force evaluator and *OpenKIM* is False

FileName

Filename of the potential, required if *pyLAMMPS* is the chosen force evaluator and *pair_coeff* is False and *OpenKIM* is False. If LAMMPS force evaluator is chosen and *pair_coeff* is a string, this entry is still required, but it can be anything.

Path2Pot

Path to potential file, defaults to False

pair_coeff

A string or a list of strings, defaults to False. If False, the code will generate a string based on *pair_style*, *FileName*, *Path2Pot* and *species* for *pair_coeff.* Example "* * Fe-P.eam.fs Fe Fe P". If *pair_coeff* is a string or a list of strings, the code generated string will be overwritten. In this case, the *FileName* is can be anything.

masses

List of atomic masses, defaults to atomic mass of elements, e.g.
- 1, 55.84
- 2, 1.0
- 3, 30.97

bondlengths

List of bond lengths (BL), defaults to bond lengths in the *pymatgen* with bond order = 1, example:
- 1 1 2.47

cutneighs

List of cutoff distance for neighbors, defaults to 1.1**bondlengths*

coordnums

List of preferred coordination numbers (CN) of each atom type, defaults to 4

cutneighmax

Maximum value of *cutneighs*, defaults to max of *cutneighs*

bondlengths4LAS

List of bond lengths (BL) for local atom strain estimation, defaults to *bondlengths*

coordnums4LAS

List of preferred coordination numbers (CN) of each atom type for local atom strain estimation, defaults to *coordnums*

The *bondlengths* and *coordnums* are used in the *FindDefects* algorithm within *active_volume*. The *bondlengths4LAS* and *coordnums4LAS* are used to estimate the local atom strain (LAS) based on the preferred bond lengths of an atom. The equation for estimating the bond strain is the same. However, the code allows a user defined distance cutoff (defaults to 1.1**bondlengths*) to count coordination numbers in *FindDefects*. For example, the commonly used distance cutoff for body center cubic (BCC) crystals in the common neighbor analysis is 1.207 of lattice constants, which includes the second nearest neighbors. The LAS estimation is limited to the bonds that are paired with its nearest neighbors. The *bondlengths4LAS* and *coordnums4LAS* allows users have different settings for finding defects and LAS.

OpenKIM

Refer to *kim* command in LAMMPS for usage.

OpenKIM

Using OpenKIM? Defaults to False

kim_init

Defaults to False. Required if OpenKIM is True. Example “Tersoff_LAMMPS metal”

kim_interaction

Defaults to False. Required if OpenKIM is True. Example “Fe”

kim_param

Defaults to False.

force_evaluator

There are six places that may call the force evaluator in SEAKMC_py. 1) Input data for MD and relaxation; 2) the data relaxation between KMC steps; 3) energy recalibration with whole data; 4) dynamic matrix calculation; 5) saddle point search and 6) prefactor calculation. The structure at first three places is the whole system. The structure at last three places is the AVs. The *force_evaluator* here are parameters for the first three places. The *force_evaluator* at last three places is available at *spsearch*.

Since SEAKMC_py will generate *Runner_color* folder for each working task, it is advised to use *Runner_color/tmp0.dat* for *read_data* and *Runner_color/tmp1.dat* for *write_data* in *RinputOpt, RinputMD0*.

Bin

Script file for calling binary or python object type, currently available ‘calllammps’, ‘callvasp’, ‘pylammps’, default to ‘pylammps’. See below for details.

Path2Bin

Absolute path to binary, defaults to False

Style

Style of force evaluator, currently available ‘lammps’, ‘vasp’, and ‘pylammps’, defaults to *Bin*

If *Style* is “pylammps”, the code uses the Python wrapper to access LAMMPS results and automatically add “Runner_color/” in front of the files (such as “tmp0.dat”, “tmp1.dat”, any dump files from “dump” command and “dynmat.dat” from “dynmic_matrix” command)

If *Style* is “lammps”, the *Bin* is a callscript-‘calllammps’. The code uses Python “subprocess” module to pipe the results from LAMMPS. The energy and forces are extracting from “log.lammps” and “dump.atom.forces”, respectively. The code will automatically remove “Runner_color/” in front of the files. Please refer to the “Use_callammps” example for detail.

If *Style* is “vasp”, the *Bin* is a callscript-‘callvasp’. The code uses Python “subprocess” module to pipe the results from VASP. All the “Rinputs” in “input.yaml” (such as “RinputOpt” and “RinputMD0”) represent INCAR files for different purposes. Two sets of KPOINTS_DATA and KPOINTS_SPS must be provided, where the KPOINTS_DATA is the KPOINTS file for the whole data, and the KPOINTS_SPS is the KPOINTS file of the AV for SPS. If *FileName* in *potential* is provided, that file is the POTCAR for all purposes. Otherwise, you need to provide individual POTCAR_symbol for all possible chemical symbols in *Path2Pot* folder. The code will generate the POTCAR based on the composition of the input POSCAR.

nproc

Number of processors for data relaxation, defaults to “auto”, i.e. 1 for serial version and total number of processors (size of MPI.COMM_WORLD).

Nproc4Recal

Number of processors for energy recalibration, defaults to “auto”, i.e. 1 for serial version and total number of processors (size of MPI.COMM_WORLD).

NSteps4Relax

The *maxeval* in *minimize* command in LAMMPS, defaults to 10000. It is also the number of molecular dynamics steps.

timestep

The timestep in molecular dynamics (MD), defaults to 0.002 ps

processors

The *processors* command in LAMMPS, defaults to False

partition

The *partition* command in LAMMPS, defaults to False

Screen

Screen out or not? Defaults to False

LogFile

False or filename? Defaults to False

RinputOpt

Runner input filename for data relaxation, defaults to False

RinputMD0

Runner input filename for energy recalibration (*barrier* and *bias* recalibration), defaults to False.

ImportValue4RinputOpt

Import value from SEAKMC_py to *RinputOpt*? Defaults to False

Keys4ImportValue4RinputOpt

A list of the [*Key*, *Value*] list for *ImportValue4RinputOpt*, defaults to [].
Example

- Timestep equal, time_step

If *ImportValue4RinputOpt* is used to dynamically import results from SEAKMC_py to the custom input file for relaxation between KMC steps. So *RinputOpt* must be a filename provided by the user. The *Keys4ImportValue4RinputOpt* is a list of the [*Key*, *Value*] list, where *Key* is the string in *RinputOpt* and *Value* is the value that is included in ***Seakmc_summary.csv***. The available values are "***istep***", "***ground_energy***", "***nDefect***", "***defect_center_xs***", "***defect_center_ys***", "***defect_center_zs***", "***nBasin***", "***nSP_thisbasin***", "***nSP_superbasin***", "***iSP_selected***", "***forward_barrier***", "***ebias***", "***backward_barrier***", "***time_step***", "***simulation_time***". Please refer to ***Seakmc_summary.csv*** in *OUTPUT* for details of these keywords.
For the example provided above, “Timestep equal” is the *Key* string that is included in *RinputOpt*, and the value of the “time_step” in ***Seakmc_summary.csv*** will be imported and assigned to “Timestep equal” (the *Key*). For example, the line-

"variable Timestep equal 3.0" is in *RinputOpt*, and the value of "time_step" at this KMC step is 50.0 ps. Then the line in *RinputOpt* will be changed to "variable Timestep equal 50.0" on the fly. Please note that it will only change the value that is immediately after the *Key*. For example, if the original line is "variable Timestep equal 3.0 velocity 3.0", the changed line will be "variable Timestep equal 50.0 velocity 3.0". For example the string after the *Key* in "variable Timestep equal 3.0 velocity 3.0" will be separated to a list with three strings-["3.0", "velocity", "3.0"]. The code will only modify the first "3.0" and restore the line with the modified value.

OutFileHeaders

A list of the file header that output from the force evaluator between KMC steps, defaults to []. Example

- dump.data

If you have a custom *RinputOpt* for data relaxation between KMC steps, in which you have custom output files. Taking LAMMPS for example, the *RinputOpt* contains the line:
"dump dump1 all custom 100 Runner_0/dump.data.* id x y z" to dump files with file header "dump.data" in the Runner_0 folder, which is the folder name for data relaxation between KMC steps.
The code will add "KMC_istep+1_" to the filename of files in Runner_0 which contain the string that inside *OutFileHeaders* and then copy them to the *DataOutPath*, which is defaulted to "DataOut".

Relaxation

Additional settings for relaxing the data between KMC steps

BoxRelax

Relaxation on box? Defaults to be False

InitTemp4Opt

Initial temperature for relaxation, default to 0.0

TargetTemp4NVT

Target temperature in NVT, default to 5.0

NVTSteps4Opt

Steps for NVT, default to 10000
If *InitTemp4Opt* > *TargetTemp4NVT,* the relaxation procedure will be relax the data structure, initialize velocity with temperature of *InitTemp4Opt*, run NVT for *NVTSteps4Opt*, and then relax the data structure, in sequence. This allows the system to escape a very shallow energy well.

data

The data entry is the input data file. The data class here is an inheritance of *LammpsData* in *pymatgen*. Thus, the data format is consistent with LAMMPS.
If the tilt components (*xy*, *xz*, and *yz*) of LAMMPS *box* are NOT included in data file, the code will automatically set them to zero. If *molecular-ID* is not included in atom properties in original data, the code will automatically expand the entries of the atom properties to include *molecular-ID* tag.

This code is not designed for MD simulations. It has an embedded MD of a NVT ensemble at 300.0 K for initial data structure. For more complex MD simulations, set *RinputMD* to the filename for MD.

The input data needs to be relaxed first. If *BoxRelax* is True, the embedded box relaxation uses *fix box/relax* command in LAMMPS, where the keyword is *iso* and the three diagonal components of the pressure tensor (*Ptarget*) are all zeros. The embedded data relaxation uses LAMMPS *minimize* command, in which *maxiter* is int(0.1**NSteps4Relax*) and *maxeval* is *NSteps4Relax*. For customized data relaxation, set *RinputOpt* to the filename for data relaxation.

For both custom *RinputMD* and *RinputOpt*, it is advised to use *Runner_0/tmp0.dat* for *read_data* and *Runner_0/tmp1.dat* for *write_data*. The *atom_style* is expanded to include *molecule-ID*. Since the code will extract the potential energy from LAMMPS, you need to compute the potential energy of the data, as in “compute pe all pe” in *RinputOpt*.

FileName

The filename of input data file, required

atom_style

Style of input data file, defaults to *atomic*. After the data file is read, the *atom_style* will be expanded to include *molecular-ID* tag. This is to allow the code to have flexibility to handle preset defects.

Relaxed

Is initial data relaxed? Defaults to False

BoxRelax

Box relaxation? Defaults to False

RinputOpt

Input filename for relaxing initial structure, defaults to False

MoleDyn

Do molecular dynamics (MD) on initial data? Defaults to False

RinputMD

Input filename for running MD of initial structure, defaults to False

RinputMD0

Input filename for running 0 step to get the energy of initial structure, defaults to False *boundary*

Periodic boundary condition, defaults to “p p p”

units

Only *metal* unit of LAMMPS is available

dimension

Only 3 is available

active_volume

One of the most important advantages of SEAKMC is to introduce the concept active volume (AV) [5]. SEAKMC_py has employed the AV concept with three options: 1) overlapping; 2) no overlapping and 3) no AV. The general procedure of constructing the AC is listed below:

1. Finding defects among the atoms of interest.
2. Point defect reduction of found defects in Step 1.
3. If *Overlapping* is False, building a superdefect based on the defects in Step2. If *Overlapping* is True, each defect in Step2 is considered as a superdefect.
4. Constructing AV on each superdefect.

The *molecule-ID* tag in atom properties of LAMMPS inputs is used to control the atoms of interest that are subjected to finding defects and constructing AVs. If the *molecular-ID* of an atom is smaller than 0, it will be considered as a preset defect atom. If the *molecular-ID* of an atom is smaller than -*NPredef*, it will NOT be subjected to finding defects nor constructing AVs. If the *molecular-ID* of an atom is smaller than 0 and larger equal than -*NPredef*, it will NOT be subjected to finding defects. To set *molecular-ID* on real-time, turn on *RT_SetMolID*. The details of real-time setting *molecular-ID* are given below at *FCT4RT_SetMolID*.

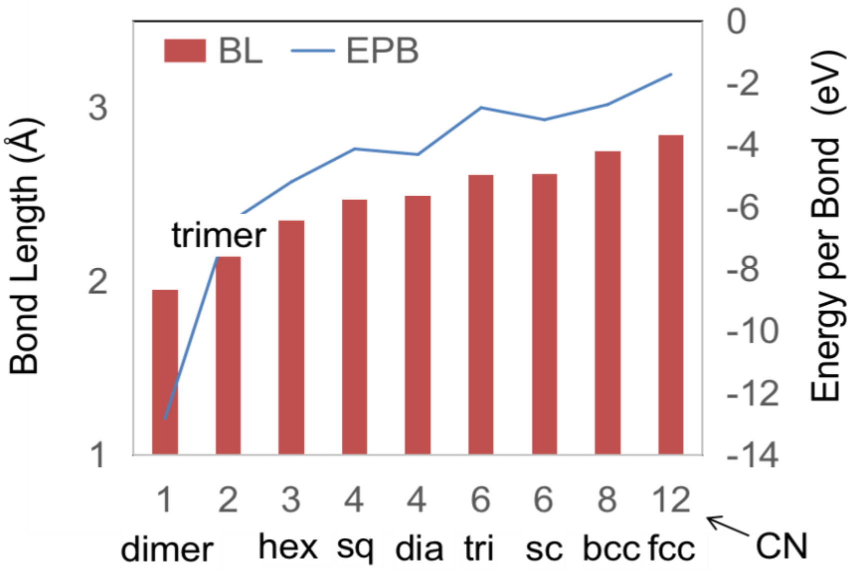


Figure 3.1. Equilibrium bond length (EBL) and energy per bond (EPB) as a function of coordination number (CN)

Whether an atom is a defect at Step 1 is based on its local environment. As shown in Figure 3.1, the equilibrium bond length (EBL) normally increases as the CN of an atom increases. The preferable CN of an atom type is given at *coordnums* and the bond length at the preferable CN is given at *bondlengths* in the *potential* entry. If the *CN*, *BL*, or *BLCN* method is used, SEAKMC_py will first compute the CN of atoms and then estimate the EBL. The atom will be considered as a defective atom based on its coordination number or/and local bond strain relative to its EBL. The details are given below in *FindDefects*.

At Step 2, the defective atoms including preset defects will be subjected to point defect reduction if *PDReduction* is True. If the distance of defective atoms to the selected defect atom is smaller than *DCut4PDR,* they will be reduced to one defect. If *RecursiveRed* is True, a chain of defective atoms with inter-defect distance < DCut4PDR will be reduced to one single defect.

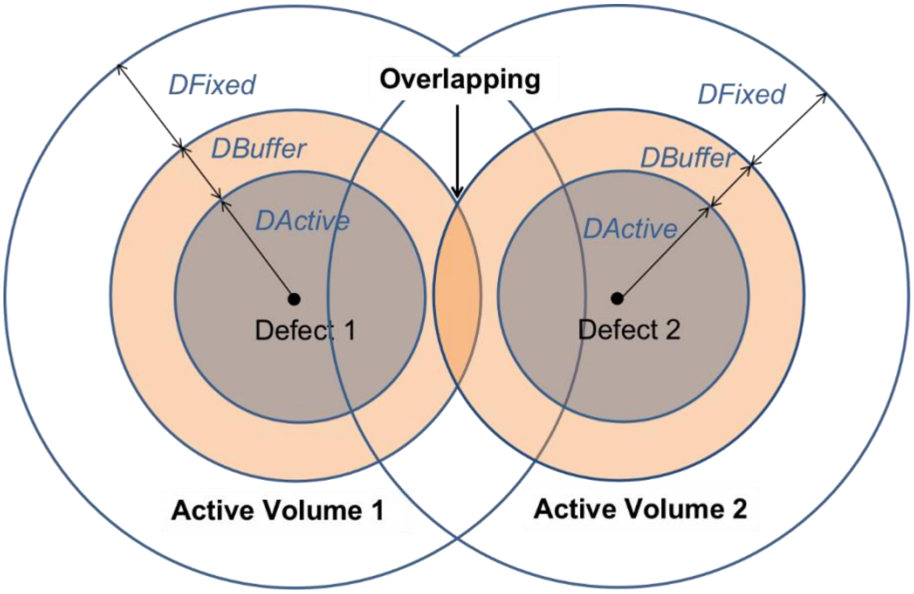


Figure 3.2. Schematic view of constructing active volume

As shown Figure 3.2, an active volume (AV) is a sphere centered with a reduced defect from Step 2. The AV normally consists of active, buffer and fixed layers. The corresponding radius/thickness are *DActive*, *DBuffer* and *DFixed*. There is no constrain on atoms in the active layer. Atoms in the fixed layer are fixed throughout saddle point search. Atoms in the buffer layer are fixed at saddle point search on active atoms and can switch to active atoms if *SearchBuffer* in *spsearch* is True.

If the distance between two defects is smaller than 2.0*(*DActive*+*DBuffer*), there will be overlapping atoms (Figure 3.2), which are assigned to different active volumes. If *Overlapping* is False, the reduced defects with the distance between at least one defect pair is smaller than 2.0*(*DActive*+*DBuffer*) will form a superdefect. In other words, a superdefect is a chain of reduced defects with the distance cut of 2.0*(*DActive*+*DBuffer*). As shown in Figure 3.2, Defects 1 and 2 will form a superdefect with Defects 1 and 2 as constituent defects, AV 1 and 2 will merge into one AV, in which the gray area is the active layer, orange area is the buffer layer and blank area is the fixed layer. The defect center of a superdefect is the center of its constituent reduced defects. If *Overlapping* is True, no action is taken at Step 3, which is equivalent to treating every reduced defect as a superdefect.

The atoms in AV are centered with the defect/superdefect. The active atoms are sorted by modified values of distance to the defect. The details of sorting atoms in AV are given below after *SortingFixed*.

Style

Style of characterizing active volume, default to defects

Available options are *all*, *custom* and *defects*. The *all* means all atoms are one active volume and all the atoms are active atoms, which equivalent to no AV concept. The *custom* option has *NActive*, *NBuffer*, and *NFixed* to define the number of active, buffer, and fixed atoms. The *defects* option means the active volume is built around defects.

NActive

Number of active atoms, only used and required for *custom* *Style*

NBuffer

Number of buffer atoms, only used and optional for *custom* *Style*

NFixed

Number of fixed atoms, only used and optional for *custom* *Style*

NPredef

Number of types of preset defects, defaults to 0
Preset defects uses *molecule-ID* tag in atom properties of LAMMPS. The atom's *molecular-ID* for preset defects must be smaller than 0. They will NOT subject to finding defects. If *molecular-ID* < -*NPredef*, they will NOT subject to finding defects nor characterizing active volume.

PredefOnly

Only consider preset defects? Defaults to False

RT_SetMolID

Real-time set *molecule-ID*, defaults to False

DefectCenter4RT_SetMolID

Manually input defect center in fractional coordinates, defaults to *Auto*

R4RT_SetMolID

Radius for RT_SetMolID, defaults to 30.0

FCT4RT_SetMolID

Fractional coordinate thresholds for *RT_SetMolID*, defaults to ["NA", "NA", "NA", "NA", "NA", "NA"]

RT_SetMolID, *DefectCenter4RT_SetMolID*, *R4RT_SetMolID*, and *FCT4RT_SetMolID* are used to real-time set the regions to find defects.
If *RT_SetMolID* is True, the atoms near the *DefectCenter4RT_SetMolID* of the whole system will be subjected to finding defects and the rest of atoms will be preset defects with *molecule-ID* = -*NPredef*-1 (*NPredef* >=1). If *DefectCenter4RT_SetMolID* is *Auto*, the fractional coordinates are the defect center of the selected basin. If *MRM* is used in *AccStyle*, it is possible that the selected basin is the data from the previously KMC step. Please note that the center of all defects is the mean coordinates of all defects; be careful since since may be misleading with PBC. It is advised to move the defects close to center in data file. If *DefectCenter4RT_SetMolID* are a list of three fractional coordinates, the defect center will use these user defined values.

If the distance to the defect center > *R4RT_SetMolID*, its *molecule-ID* will be set to -*NPredef*-1, i.e. it will not be subjected to finding defects nor constructing active volume.

FCT4RT_SetMolID contains 6 fractional coordinate threshold values of [xmin, xmax, ymin, ymax, zmin, ymax]. Only the atoms within this block region will be subjected to finding defects. The range of fractional coordinates is [0.0, 1.0). To activate *FCT4RT_SetMolID*, at least one item of *FCT4RT_SetMolID* is a float number.

cutdefectmax

Maximum cutoff distance of defects, defaults to maximum of *cutneighs* in *potential*

FindDefects

Method

Method of finding defects, defaults to *BLCN*

Available keywords are *CN*, *BL*, *BLCN*, *WS*, *all* and *custom*.

CN: based on CN difference from the CN in *coordnums* (CNC)

BL: based on BL difference between the BL in *bondlengths* (BLB).

BLCN: based on both CN and BL methods.

WS: Wigner-Seite method, i.e. compare with a reference data

all: all atoms are considered as defects

custom: Manually input defects

Defects

If *Method* in *FindDefects* is *custom*, it requires manually input the Cartesian coordinates of defects, example

- 10.0, 10.0, 10.0
- 15.0, 15.0, 15.0

ReferenceData

The filename of reference data, only used and required for *WS* *Method*

atom_style4Ref

Atom_style of reference data, defaults to atom_style of input data

DCut4Def

Distance cutoff for finding defects

If it is *WS* method, the ratio to *cutdefectmax*, defaults to 0.2. If it is *BL* or *BLCN*, the ratio to BLB, defaults to 0.1

The *CN*, *BL*, and *BLCN* methods allow identification of defects in the data on a real-time basis. First, the code counts the coordination number (CN) of each atom based on *cutneighs* in *potential*. Then, the equilibrium bond length (EBL) is estimated by the equation:

$$EBL = \begin{cases} 0.8 * BLB,\ if\ EBL < 0.8 * BLB \\ BLB * \left(1.0 - \frac{0.2*(CNC-CN)}{CNC}\right) \\ 1.2 * BLB,\ if\ EBL > 1.2 * BLB \end{cases} .$$

If the bond length of any bonds of an atom is smaller than (1.0-*DCut4Def*)*EBL or larger than (1.0+*DCut4Def*)*EBL, this atom is considered as an defective atom.
As the defaulted bond length in *pymatgen* is close to the lattice constant for BCC structures, which is the second nearest neighbors' distance, it will arise problems to find defects using *BL* or *BLCN* method. It is advised to set the BLB to 0.866 of lattice constant and CNC to 8 for BCC structures. If *CN* method is used for this method, you can use *bondlengths* and *coordnums* suggested by *common neighbor analysis* to differ BCC, FCC, HCP structures.

In the case where local atomic strain (LAS) is used, the *bondlengths4LAS* and *coordnums4LAS* are used to estimate EBL. The LAS of an atom is estimated by

$$LAS = \sum_{i=0}^{CN} \frac{100*(BL_i - EBL)}{EBL*CN},$$

where CN is the coordinate number of the atom. The applied LAS (LAS_app) is the difference between the LAS of an atom and the mean value of LAS of all atoms:

$$LAS\_app = LAS - \overline{LAS},$$

where $\overline{LAS}$ is the mean value of LAS, which is assumed to be the LAS of the unstrained atoms. If the CN of an atom is zero, the LAS of the atom will set to the maximum value of LAS_app.

As described in *potential*, the *bondlengths4LAS* and *coordnums4LAS* are defaulted to the *bondlengths* and *coordnums*, respectively. But they are allowed to have different values for estimating LAS_app.

PDReduction

Reduce defects to point defects? Defaults to True

SortD4PDR

Sorting defects by their distance to defect center before point defect reduction? Defaults to False

DCut4PDR

Distance cut for *PDReduction*, defaults to *cutdefectmax**1.4

RecursiveRed

Recursive *PDReduction*? Defaults to False
The CN, BL, and BLCN methods normally consider a group of atoms around the defect as defective atoms. *PDReduction* will reduce the defective atoms with distance smaller than *DCut4PDR* into one single defect. If *SortD4PDR* is True, defects will be sorted by their distance from the center of all defects before the point defect reduction. This will cause *PDReduction* to start from the defects near the center of all defects, which may result in a smaller number of defects after *PDReduction*. The *RecursiveRed* will recursively perform point defect reduction on defects. As a result, the *RecursiveRed* will reduce a chain of defects with inter-defect distance < *DCut4PDR* to one single defect.

Overlapping

Allow overlapping? Defaults to True

Stack4noOverlap

Stack the superdefects from no *Overlapping* to reduced defects? Defaults to False

DCut4noOverlap

Distance cut for Overlapping, default to 2.0*(*DActive*+*DBuffer*)

When *Overlapping* is False, the active and buffer atoms can only be assigned to one unique AV. To guarantee no overlapping, the default distance cutoff of constituent reduced defects in a superdefect is 2.0*(*DActive*+*DBuffer*). This distance cut can be overwritten by *DCut4noOverlap*. If *Stack4noOverlap* is True, the superdefects from no *Overlapping* will add in front of the reduced defects from Step 2. That is to say, the AVs from superdefects always appear earlier than those from reduced defects.

Order4Recursive4PDR

The recursive order for recursive point defect reduction, default to “NA”, only available for Pro-version.

Order4Recursive4AV

The recursive order for building superdefects for constructing AV, default to “NA”, only available for Pro-version.

Overlap4OrderRecursive

Allow overlapping for *Order4Recursive4AV*, default to True, only available for Pro-version.

Both *RecursiveRed* and building superdefect find the neighbors of single defects recursively and build a chain of single defects starting from a given single defect. The difference is that *RecursiveRed* reduces the chain of single defects to a single defect. For building superdefect, the chain of single defects is reduced to a superdefect. The recursive order means how far the neighbors go in the chain of single defects. If the recursive order is 1, the chain of single defects includes the first neighbors of the starting single defect. If the recursive order is 2, the chain of single defects includes the neighbors of the neighbors of the starting single defect and so on. If the recursive order is not an integer (default values), the recursive order is an infinity number.

The *Order4Recursive4AV* is only used for building superdefect (*Overlapping* must be False). Set *Order4Recursive4AV* to a finite integer, it will reduce the size of AV. This will improve the probability of finding saddle points and reduce the time cost for SPS. The risk is some collective motion of saddle points might be missed. Like the idea of the *Overlapping*, the as-build superdefects allow overlapping if *Overlap4OrderRecursive* is True. In another words, a single defect can be included in multiply superdefects. If *Overlap4OrderRecursive* is False, a single defect is exclusively belonging to a superdefect.

NMax4Def

Max number of defects, defaults to False (no limit)

DActive

Distance of active atoms in AV, defaults to 4.1**cutdefectmax*

DBuffer

Distance between active and buffer atoms, defaults to 0.4**cutdefectmax*

DFixed

Distance between buffer and fixed atoms, defaults to 4.1**cutdefectmax*

NMax4AV

Max number of active atoms contained in an AV, defaults to False (no limit)

NMin4AV

Min number of active atoms contained in an AV, default to 40

Sorting

Sorting the active atoms in AV, defaults to True

Sort_by

Sort values, defaults to *Auto*

SortingSpacer

Spacer for sorting atoms in active volume, defaults to [0.3, 0.3, 0.3]

SortingShift

Coordinates shift for sorting atoms, defaults to [0.0, 0.0, 0.0]

SortingBuffer

Sorting buffer atoms? Defaults to False

SortingFixed

Sorting fixed atoms? Defaults to False

Sorting, *Sort_by*, *SortingSpacer*, *SortingShift*, *SortingBuffer*, and *SortingFixed* are settings for sorting the atoms in AVs, which are centered around its defect. If *Sorting* is True, it will sort active atoms. To sort atoms in buffer or fixed layers, set *SortingBuffer* or *SortingFixed* to True. *Sort_by* is a list of sorting values. If *Sort_by* is Auto means the sorting values are ["*D*", "*X*", "*Y*", "*Z*"], which is equivalent to

Sort_by:

- *D*
- *X*
- *Y*
- *Z*

The capitalized letters *X*, *Y*, and *Z* represent the corresponding vague values (the nearest integer of the rescaled and shifted original values). Taking *X* as an example, *X* = round((*x*-*SortingShift*[0]) / *SortingSpacer*[0], 0). *D* is the square root of $X^2+Y^2+Z^2$. The other available options has two sets: 1) *DXY*, *DXZ*, and *DYZ*, which are the square root of X^2+Y^2, X^2+Z^2, Y^2+Z^2, respectively; and 2) *DX*, *DY*, and *DZ*, which are the absolute value of *X*, *Y*, and *Z*, respectively. The selections of *SortingSpacer* and *SortingShift* should be such that the resulting in *X*, *Y*, and *Z* before rounding are close to an integer. The suggested *SortingSpacer* is the inter-layer distance in *x*, *y*, and *z* directions. The *SortingShift* is either 0.0 or half of *SortingSpacer* to make most atoms' *X*, *Y*, and *Z* are near to an integer. When you use external database such as *DefectBank* or preloading displacements, the code will assume the sequence of atoms are sorted by ["*D*", "*X*", "*Y*", "*Z*"].

PointGroupSymm

Applied point group symmetry operations, default to False

NMax4PG

Max number of atoms for finding point group symmetry, default to 1000
SEAKMC_py uses the symmetry analysis in *pymatgen* to find the point group symmetry of AVs. If *PointGroupSymm* is True, a set of saddle points with the same type will be generated by all point group symmetry operations.

TurnoffPBC

Turn-off the periodic boundary condition (PBC) of the AVs? Defaults to [False, False, False]
In the case of using multiple processors for SPS, it may result in zero or very few atoms on some processors since the *box* of AVs is inherited from the *box* of the data. In this case, it is advised to turn off PBC on the selected directions by setting *TurnoffPBC* to True on the selected directions.

dynamic_matrix

To compute *dynamic matrix* in LAMMPS, the phonon package is required. The dynamic matrix is used in the scaled normal coordinates (SNC) [6] and computing the prefactor.

Please note that the dynamic matrix is a 3N*3N matrix, where N is the number of atoms. The time cost scales with 3N for computing the dynamic matrix and with N^3 with diagonalizing the dynamic matrix. The memory usage scales $9N^2$. As a result, it will be a very demanding process if the number of active atoms in an AV is large.

SNC

Scaled normal coordinates, defaults to False

CalPrefactor

Calculate prefactors? Defaults to False

NMax4SNC

Maximum number of atoms for SNC, defaults to 1000

FileName

The filename for dynamic matrix, fixed to "*dynmat.dat*"

displacement

Displacement for computing dynamic matrix, defaults to 0.000001

delimiter

Delimiter in dynamic matrix file, defaults to " "

LowerHalfMat

Is the dynamic matrix a lower half matrix? Defaults to False

Method4Prefactor

Method of prefactor calculations, only “harmonic” [10] is available

VibCut

Vibrational frequency cutoff for prefactor calculation, below *Vibcut* will be neglected for prefactor calculations, defaults to 1e-8

spsearch

Within transition state theory the key problem in KMC becomes that of finding the free energy barrier of saddle points on the potential energy surface (PES) associated with the initial state. Instead of using the whole data as the initial state, each AV is considered to be an initial state in SEAKMC_py. The saddle point search (SPS) is performed on the potential energy surface of every AV. At present SEAKMC_py has only implemented the dimer method [1] [2] for SPS.

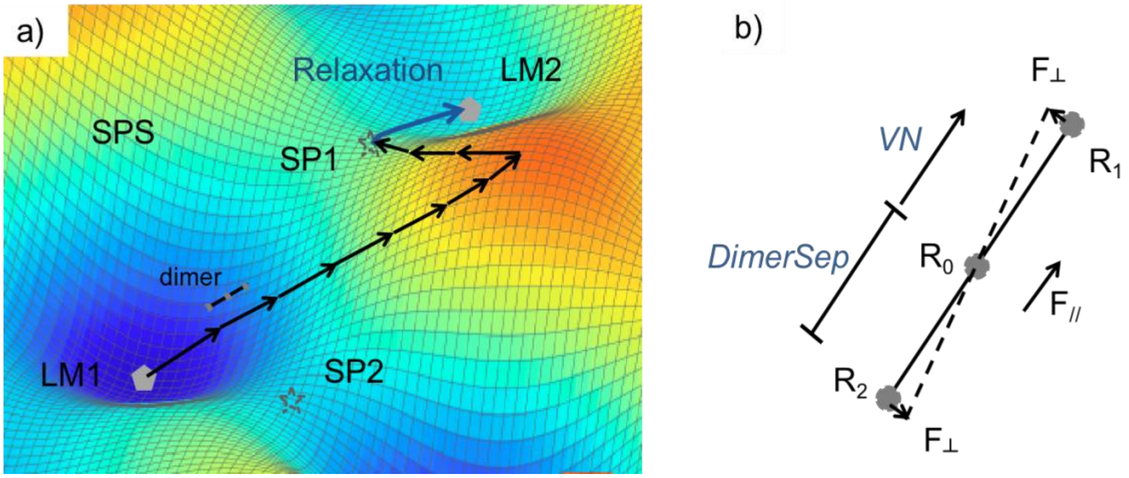


Figure 3.3. a) Saddle point search on the potential energy surface; b) Schematic view of a dimer

Figure 3.3a illustrates a potential energy surface of a N-atom AV, where local minimum one (LM1) is the starting state of the AV. Without knowledge of the final state, SPS starts with creating a dimer as shown in Figure 3.3b. As defined in Ref. [1], the dimer is a pair of images ($\mathbf{R}_1$ and $\mathbf{R}_2$ in Figure 3.3b) separated from their middle point $\mathbf{R}_0$ by a distance *DimerSep*. The vector *VN* is defined as a unit vector pointing from $\mathbf{R}_2$ to $\mathbf{R}_1$, which is referred to the dimer orientation in Ref. [1] and the dimer axis in Ref. [2].

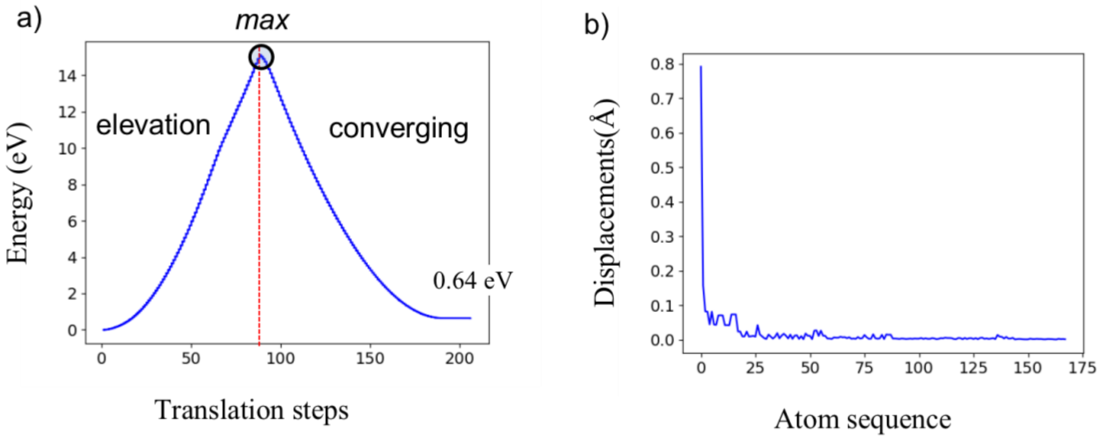


Figure 3.4. The energy and displacements during a dimer search of a 168-atom AV with a vacancy at the center of an Fe BCC system. a) The energy profile as a function of dimer translation steps; b) Atomic displacements vs atom sequence of the saddle point.

Figure 3.4 shows the energy profile during the dimer search and the atomic displacements of the saddle point, where the system is Fe BCC structure with a vacancy. The AV has 168 active atoms, sorted by [“*D*”, “*X*”, “*Y*”, “Z”], as indicated by *Sorting* in *active_volume*. The Fe-P interatomic potential [11] is used for this study. The settings that involved in the dimer search are all the defaulted values, as shown below. Without prior information of the saddle point configuration nor the final state, a dimer search attempt launches with a random unit vector (*VN*). The typical dimer search process involves two stages:

1) Leaving the initial harmonic well (referred to the “elevation” stage). The dimer curvature (***C***) in this stage is larger than zero. The search process translates the dimer “uphill” on the potential energy surface until the dimer curvature reaches zero where the energy reaches a maximum (Point “*max*” in Figure 3.4).
2) Converging to the saddle point (referred to the “converging” stage). The dimer curvature in this stage is normally smaller than zero. The dimer moves “downhill” to the saddle point, which is a maximum along the lowest curvature mode and the minima along all other modes.

Along the dimer searching process, the dimer will first rotate to find the lowest curvature mode and then translate the dimer with the orientation of the dimer unchanged. Thus, the unit vector *VN* can be regarded as the partition of the displacement vector associated with the lowest curvature mode. The rotation forces ($\mathbf{F}_{\perp}$ in Figure 3.3b) are computed by the difference of forces on the dimer acting normal to the *VN*. When the dimer curvature is positive as it is in the “uphill” region (the dimer curvature may be slightly less than 0.0 near the peak of the “uphill” region), the translation forces equal to the negative value of the forces that parallel to *VN* of the dimer ($-\mathbf{F}_{//}$). When the dimer curvature is negative, the translation forces are $\mathbf{F}_{R}-2\mathbf{F}_{//}$, where $\mathbf{F}_{R}$ are true forces on dimer.

For the given structure and settings, the dimer search takes 88 translation steps to reach the max with the energy of 15.1 eV and total 206 translation steps to find the saddle point. The energy barrier of the saddle point is about 0.64 eV, and the atomic displacements of the saddle point are shown in Figure 3.4b, in which only one atom that is close to the defect center moves 0.8 Å and the rest atoms are barely moved.

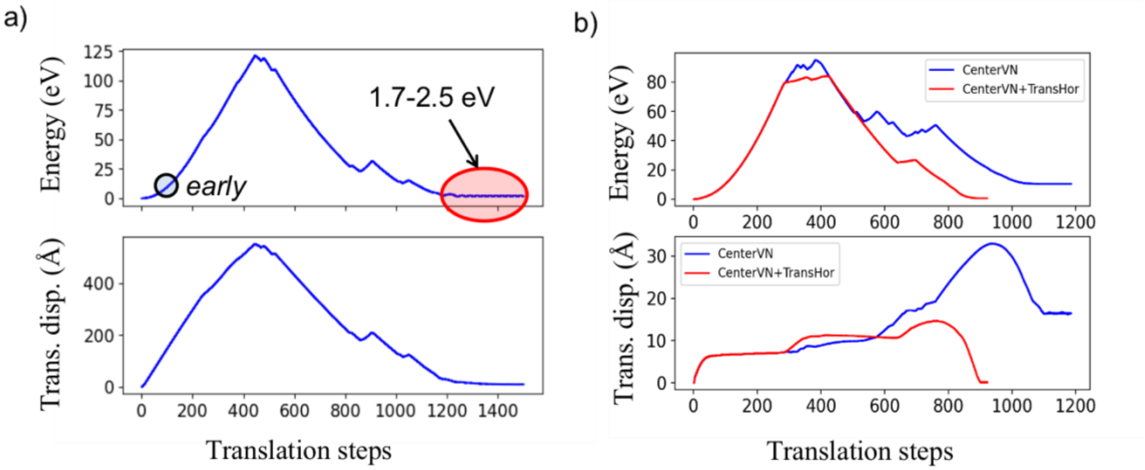


Figure 3.5. The energy and translation displacements (*Trans. disp.*) during a dimer search of a 964-atom AV with a vacancy at the center of an Fe BCC system. a) The energy profile and translation displacement as a function of dimer translation steps; b) The energy profile and translation displacement as a function of dimer translation steps with centering *VN* (*CenterVN*) and centering *VN*+translating dimer horizontally (*CenterVN* + *TransHor*) center.

The dimer method is an efficient technique to find the saddle point on the fly in many applications. However, it is found to be problematic in some relatively large systems, as shown in Figure 3.5a, where the AV is a 964-atom Fe BCC sphere with a vacancy at the center. The translation displacement at the *iter*th translation step (*dsum_iter*) in x, y, and z directions are the summation of individual displacements at the *iter*th translation step in x, y, and z directions ($D_{i,n;iter}$), respectively:

$$dsum\_iter = \sum_{n=1}^{N} D_{i,n;\, iter}$$

where i indicates x, y, and z directions. The value plotted in Figure 3.5 is the magnitude of *dsum_iter*.

As shown in Figure 3.5a, the maximum energy is about 121 eV and the magnitude of the translation displacements is about 544 Å at 460 translation steps. In the converging stage of a dimer search, the energy drops to 2.5 eV and the magnitude of the translation displacements drops to 30 Å at about 1190 translation steps. Then the dimer was trapped on the PES and the energy oscillated from 1.7 eV to 2.5 eV.

With a such high peak energy, the high-lying modes can be activated at the elevation stage of the dimer search. Thus, it takes more steps to converge to the saddle point for a larger system, and may even be trapped there forever. The modifications below are primarily focused on scaling down the undesirable components of the unit vector *VN* during the dimer search, and therefore accelerate the dimer search. These modifications include:

1) removing the net translation displacements (*CenterVN*),

2) translating the dimer horizontally (*TransHorizon*),
3) using the scaled normal coordinates (*SNC* in *dynamic_matrix*),
4) applying the rescaling factors (*RescaleVN* in *HandleVN*).

Please note that applying 3 and 4 simultaneously will lead to overcorrection of the low-lying modes and thus fail to find the right location of saddle points.

The net translation displacements inside the *VN* are removed by deducting its net translation displacements per atom (see the equation below) if *CenterVN* is True. After removing the net translation displacements, the values of $\sum_{n=1}^{N} VN_{i,n;\,iter}$ equal zero in *x*, *y*, and *z* directions. Please note, the *CenterVN* procedure is applied every *NSteps4CenterVN* translation steps after the first *IgnoreSteps* translation steps. The net translation displacements are largely originated from the translation mode, in which the displacement eigenvectors have the same direction (in-phase). This *CenterVN* has changed the displacement pattern of the translation mode. On the other hand, the rigid shift of *VN* in this *CenterVN* is very small. It is safe to set *CenterVN* to True in a wide range of applications.

$$VN_{i,n;\,iter} = VN_{i,n;\,iter} - \frac{1}{N}\sum_{n=1}^{N} VN_{i,n;\,iter}$$

The *TransHorizon* is used to translate the dimer horizontally on the PES. In some cases, instead of directly converging to the saddle point, the dimer converges to a contour location on the PES (somewhere between the saddle point and the potential energy basin). Then the dimer curvature turns into positive again, and it starts to climb on the PES as it does in the elevation stage. The idea of the translating the dimer horizontally (TDH) is to set the translation force to the rotation force and thus translate the dimer horizontally on the PES if its curvature turns into positive again. The change of dimer translation direction may help the dimer leave the energy basin at the "converging" stage and thus lead the dimer to saddle point rather than oscillating at the shallow convex region of the PES as shown in red ellipse region in Figure 3.5a. The *TransHorizon* is activated only after the dimer curvature has turned into positive again. If the dimer curvature changes to positive second time or after and the energy is larger than *En4TransHorizon*, the dimer will translate horizontally.

$$\boldsymbol{F}^{\dagger} = \boldsymbol{F}_1^{\perp} - \boldsymbol{F}_2^{\perp}, \qquad if\ \boldsymbol{C} > 0\ at\ second\ time\ or\ after$$

The energy profile and translation displacements during dimer search with *CenterVN* and *CenterVN*+*TransHorizon* are shown in Figure 3.5b. The maximum energies have dropped to about 80 eV with these two modification methods. The energy barrier of the found saddle point is about 10 eV at 1200 translation steps for modification with *CenterVN* and 0.64 eV at 930 translation steps for modification with *CenterVN*+*TransHorizon*.

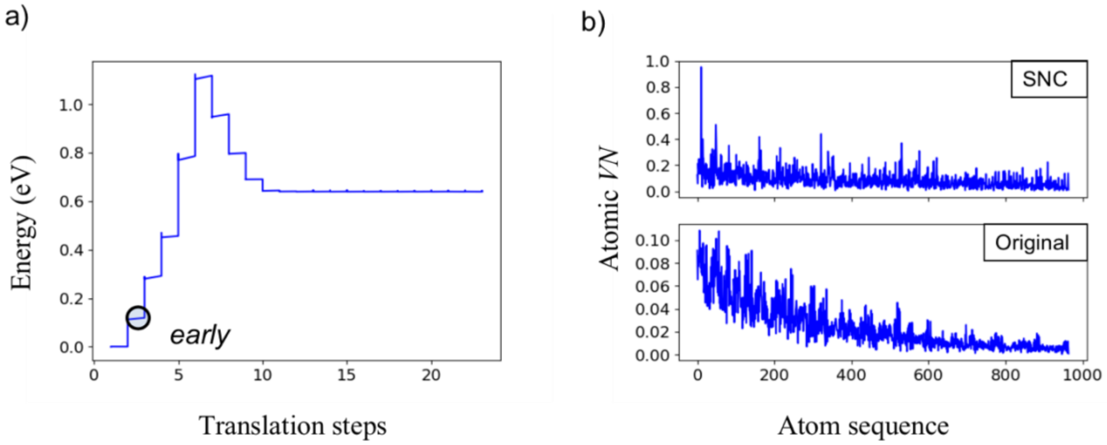


Figure 3.6. The energy and *VN* during dimer search of a 964-atom AV with a vacancy at the center of an Fe BCC system. a) The energy profile with SNC [6]; b) The atomic *VN* at *early* stage in dimer search. Top view is the dimer method with the SNC method and bottom view is the original dimer method.

Figure 3.6 shows the energy profile and *VN* during dimer search of a 964-atom AV using the SNC method [6]. The maximum energy and number of translation steps to find the saddle point was dramatically dropped to 1.1 eV and 23 steps, respectively. The snapshots of atomic *VN* at an early stage of the "elevation" during dimer search using the SNC (*SNC*) method and the original dimer method are compared in Figure 3.6b. The SNC has converted the true forces to forces on the dimer based on the phonon modes (eigenvectors of the dynamic matrix) and phonon energies (square root of the eigenvalues of the dynamic matrix). The resulting *VN* (top view in Figure 3.6b) reflects the superimposition of phonon modes that scaled with their phonon energies.

Compared to the *VN* when using the SNC method (top view in Figure 3.6b), the partition of the *VN* vector using the original dimer search (bottom view in Figure 3.6b) is exponentially decayed with the atom sequence or the distance to the defect. In addition, the *VN* can largely reflect the characteristic displacements of the phonon modes (the spikes in bottom pane in Figure 3.6b). Since the *VN* is a unit vector, the largest atomic *VN* in the bottom pane is about 9 times smaller than the atomic *VN* in the top pane in Figure 3.6b. If the characteristic reaction coordinates to reach the *max* point on the PES are about the same, it requires many more iterations for the dimer search without the SNC method.

While the SNC can significantly improve the efficiency in the dimer search, the time cost for computing dynamic matrix (scaling with 3N) and diagonalizing the dynamic matrix (scaling with N^3) are very expensive. In addition, the memory required to store the eigenvectors and inversion of eigenvectors in SNC scales with $9N^2$. When the number of atoms in an AV is large, it is inefficient, even impractical, to use the SNC method.

The AV has significantly reduced the dimensionality of the system during the SPS. As shown in Figure 3.4, the AV with 168 active atoms is sufficiently large for the Fe-vacancy system. The further analysis shows that only 11 out of 168 atoms have been displaced more than 0.05 Å in Figure 3.4b. As only a few of atoms are deformed in the saddle point configuration, we have introduced the dynamic active volume (DAV) to further reduce the dimensionality of the PES by deactivating the motion of the selected atoms that are not critical for the saddle point configuration. The DAV method is only applied at the elevation stage of a search process. At the subsequential converging stage, the dynamic boundary in DAV is lifted to accommodate the displacements that required to find the right location of the saddle point.

The dynamic boundary in DAV is realized by introducing a rescaling factor ranging from 0 to 1 on each active atom inside an AV. This N-dimensional rescaling factor array is first broadcasted to a 3N dimensional array, where the rescaling factors on x, y, and z directions of each atom are identical with its rescaling factor of the atom. The broadcasted rescaling factor array is imposed on the *VN* to rescale the components inside the *VN* and the rescaled *VN* is then converted to a unit vector. This procedure is denoted by "*RescaleVN*". The rescaling factor of 0.5 is consider as the dynamic boundary. The atoms with the rescaling factor below 0.5 are deactivated atoms. With the DAV method, the components of the *VN* for atoms with the rescaling factor below 0.5 are deactivated, which leads the dimer search more focuses on the activated atoms. From rescaling factor point of the view, the AV concept introduced by SEAKMC is equivalent to impose a rescaling factor of 1.0 on active atoms and 0.0 on the fixed atoms in the AV throughout the dimer search. Please note that *RescaleVN* only applies at the elevation stage of a dimer search. At the converging stage, it is lifted to allow the dimer finding the right location of the saddle point.

The rescaling factors of active atoms are generated by the rescaling function. The available rescaling functions include the *Sigmoid* and Heaviside *step* functions, in which the *Sigmoid* function is default selection. The Heaviside *step* function is 0.0 if the input value is smaller and equal than the user defined zero point, and otherwise, it is 1.0. The Sigmoid function:

$$\tilde{S}(\hat{x}^*) = \frac{1.0}{1.0 + e^{-\hat{x}^*}}$$

is exponentially decayed from 1.0 to 0.0 with decreasing $\hat{x}^*$ and equals to 0.5 when $\hat{x}^*$ is 0.0. The range of $\hat{x}^*$ (*xrange*) is a user-defined parameter that influences the rescaling efficiency. The *xrange* needs to be reasonably large but the rescaling efficiency of the Sigmoid function varies insignificantly when the *xrange* is larger than 20 due to its exponential term. The $\hat{x}^*$ is the mapping of the input values ($\hat{x}$) through the following equations.

$$\begin{cases} \hat{x}_{incr} = {(\hat{x}_{max} - \hat{x}_{min})}/{xrange} \\ \hat{x}_{zero} = \hat{x}_{min} + (\hat{x}_{max} - \hat{x}_{min})\,\hat{x}_{r0} \\ \hat{x}^* = {(\hat{x} - \hat{x}_{zero})}/{\hat{x}_{incr}} \end{cases}$$

The $\hat{x}_{max}$ and $\hat{x}_{min}$ are the maximum and minimum value in the input values, respectively. The $\hat{x}_{r0}$, which is the ratio between $\hat{x}_{zero} - \hat{x}_{min}$ and $\hat{x}_{max} - \hat{x}_{min}$, is another user-defined parameter to define the location of the zero point of $\hat{x}^*$.

The available inputs (keyword in parentheses) for the rescaling function include the reverse atom sequence (*RAS*), logarithm of the modified relative atomic displacement (*LOGN*), and a combination of *LOGN* and RAS inputs (*LOGNRAS*). The relative atomic displacement $\widetilde{N}$ is defined as the magnitude of *x*, *y,* and *z* components of an atom in the 3N-dimensionlal unit vector *VN*:

$$\widetilde{N} = \sqrt{\sum_{i=0}^{2} VN_{i,n}^{2}}$$

where *n* is the atom index number and *i* = 0, 1, and 2, which stands for *x*, *y*, and *z* axis, respectively.

Below we have presented the details of DAV using the Fe-vacancy system. The r_{active} (the radius of the AV) is set to 7.7 Å and the resulting AV consists of 168 active atoms. The RAS, LOGN and LOGN+RAS rescaling factors in Figure 3.7a are computed based on the same relative atomic displacement ($\widetilde{N}$) of a given translation step, which is presented as "Original" in the top panel of Figure 3.7b. Together with the original $\widetilde{N}$, the resulting $\widetilde{N}$ after applying DAV are plotted in Figure 3.7b.

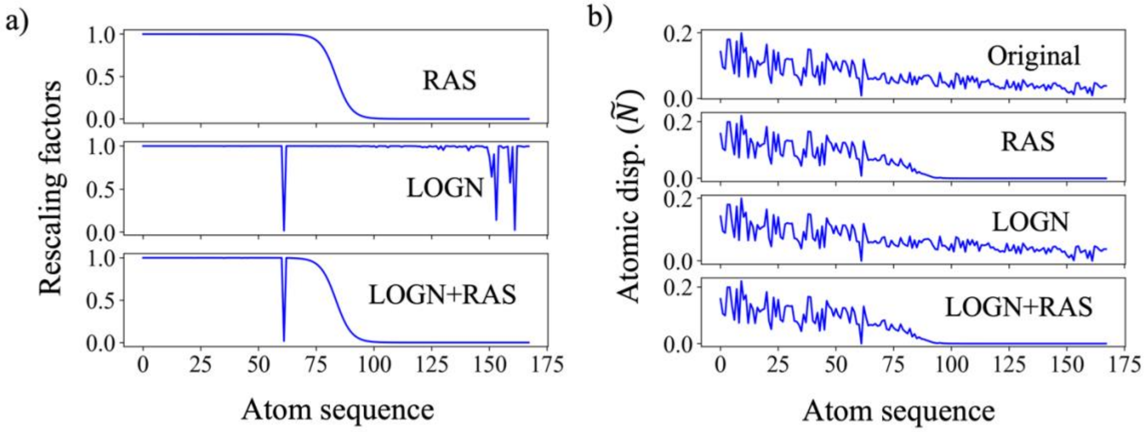


Figure 3.7. The rescaling factors and relative atomic displacement ($\widetilde{N}$) after applying the rescaling factors as a function of the atom sequence of the 168 active atoms in the Fe-vacancy AV: a) The rescaling factors with three different input values; b) The $\widetilde{N}$ after applying the three rescaling factors.

The default values of the dynamic boundary are listed in *HandleVN* below. Each input value for rescaling function has similar parameters, such as *XRange4RAS* (the *xrange* for *RAS* rescaling) and *Ratio4Zero4RAS* (The $\hat{x}_{r0}$ for *RAS* rescaling), with the corresponding appendix. Below are the detail explanations of these parameters using *RAS* as an example.

The atom sequence is ascendingly sorted by ["*D*", "*X*", "*Y*", "Z"] by default. Thus, the larger index number of an atom, the closer it is to the boundary of the AV, which is mandatorily remained fixed throughout the dimer search. The reverse atom sequence (RAS) rescaling is to deactivate the atoms near the boundary of the AV. The input value for the RAS rescaling is a list from N-1 to 0. The *XRange4RAS* default to 40.0. If *Ratio4Zero4RAS* is 0.5, the last 50% of atoms (84 atoms) are deactivated as shown in Figure 3.7. If the AV is a sphere, the 50% in volume is equal to 21% of the distance to the edge of the sphere.

That is to say, the atoms within 0.21**DActive* distance to the frozen atoms are deactivated at the elevation stage of dimer search. For different defect configurations or AV shapes, such as the dislocations or interfaces, the sorting sequence (*Sort_by* in *active_volume*) as well as *Ratio4Zero4RAS* needs to be adjusted to avoid the possible over-deactivation. For some cases, such as a dislocation junction, you should disable the RAS rescaling since the atom sequence cannot reflect the distance to the AV boundary. The default value of *Ratio4Zero4RAS* is 0.3.

For LOGN rescaling, the input values are the modified relative atomic displacement, $\widetilde{N}$, through the equation below.

$$\hat{x} = ln\left(MA(\widetilde{N})^{PowerOnV} + e^{MinValue4LOGV}\right)$$

The *MinValue4LOGV* (default to -20) is used to avoid logarithm on zero. *MA* stands for moving average with the period of *Period4MA* (defaults to 1). The *PowerOnV* (defaults to 4.0) is the power on the relative atomic displacement. With this modification on the relative atomic displacement, the input values have a more spread distribution in a range from -20 to 0. The *XRange4LOGV* in for *LOGN* rescaling defaults to 20.

The selection of *Ratio4Zero4LOGV* should be as efficient as possible to deactivate the unimportant atoms without the possible over-deactivation for a wide range of applications. For most applications, the $\widetilde{N}$ typically has a long-tail distribution, in which the minimum value in $\widetilde{N}$ is almost zero and the maximum value in $\widetilde{N}$ often varies from 0.1 to 0.3. Assuming the minimum value in $\widetilde{N}$ is zero ($\hat{x}_{min}$is -20), *Ratio4Zero4LOGV* reflects a threshold ratio to the maximum value in $\widetilde{N}$, below which the atoms are deactivated. The default value of *Ratio4Zero4LOGV* is 0.2. With this setting, the threshold ratio is 4.8%, if the maximum value in $\widetilde{N}$ is 0.3, which is equivalent to a relative atomic displacement of 0.0144. That is to say, the *LOGN* rescaling would deactivate the atoms if their relative atomic displacements are below 0.0144, which is a reasonable selection in most applications. Since the displacements during a saddle point search are often involved in a localized region, the maximum value of $\widetilde{N}$ barely changes with increasing size of an AV in different applications. More importantly, this *LOGN* rescaling can deactivate the unimportant atoms more efficiently as the size of an AV increases. For example, it deactivates 3% of atoms for the Fe-vacancy case with 168 active atoms as shown in Figure 3.7. For a <110> dumbbell interstitial in Fe system which consists of 610 active atoms, the *LOGN* rescaling deactivates 25% to 73% of atoms depending on dimer search steps. It deactivates 70% to 85% of atoms for a 2006-atom AV of a self-interstitial loop in the Fe system.

Since the atom sequence is unchanged during a dimer search, the RAS rescaling factor of each atom, which reflects its distance to the AV boundary, remains the same throughout the elevation stage of a SPS. On the other hand, the *LOGN* rescaling factor, which reflects the relative magnitude in $\widetilde{N}$ at a given dimer search step, is dynamically changing. If *TakeMin4MixedRescales* is True, the minimum rescaling value between the *LOGN* and *RAS* rescaling methods is applied (label as "LOGN+RAS" in Figure 3.7), which is the default selection for the DAV method in SEAKMC_py.

Figure 3.8 compares the energy profiles of a dimer search without ("Original") and with three different DAVs for the simple point defects in Fe systems. The simple point defects include the Fe-vacancy case and a <110> dumbbell interstitial (Fe-dumbbell), where The AV radii are set to 7.7 and 12 Å for Fe-vacancy and Fe-dumbbell cases, respectively. The resulting AV contains 168 active atoms for the Fe-vacancy case and 610 active atoms for the Fe-dumbbell case. Only one dimer search attempt is launched using an identical initial unit vector for both cases.

For the Fe-vacancy case, the peak energy is 11.7 eV and the number of translation steps required to converge to a saddle point is 177 using the original method. As shown in Figure 3.8a, the peak energies are reduced to 5.9, 10.2, and 5.6 eV, and the number of translation steps are reduced to 118, 165, and 115 using RAS, LOGN, and LOGN+RAS DAVs, respectively. For the Fe-dumbbell case in Figure 3.8b, the peak energies are reduced to 11.2, 6.4, and 6.0 eV from 33.4 eV for RAS, LOGN, and LOGN+RAS DAVs, respectively. The resulting number of translation steps are 244, 190, and 180 for these three DAVs, whereas the original method requires 445 translation steps to find the saddle point. With the DAVs, the displacements of the unimportant atoms are significantly suppressed at the elevation stage, which leads the dimer exploring the PES on a reduced dimensionality and thus accelerates the SPS.

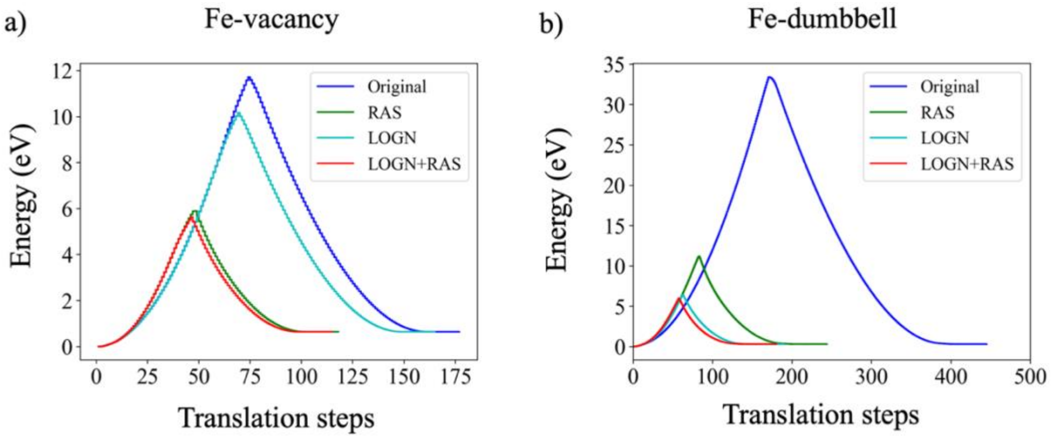


Figure 3.8. A comparison of the energy profiles of a dimer search with the different DAVs: a) the 168-atom Fe-vacancy case, and b) the 610-atom Fe-dumbbell case.

With the modified dimer method including *CenterVN*, *TransHorizon*, *AdaptiveDimerSep* and the dynamic boundary, SEAKMC_py can finish the dimer search of a 20000-atom system within several minutes.

If a saddle point is found, $\mathbf{R}_2$ to $\mathbf{R}_1$ will be subjected to local relaxation (*LocalRelax*). If one of images can restore the initial configuration, and the other image leads to the final state as exemplified LM2 in Figure 3.3a, the connectivity of this saddle point is True. Otherwise, it is not a saddle point that connects two neighboring states, i.e. the connectivity of the saddle point is False.

Method

Method of SPS, only "*dimer*" is available

NSearch

Number of SPS attempts, defaults to 10.

If *NSearch* is large, the memory usage might be demanding. For a 1000-atom AV, 1GB memory can store ~2500 saddle points and their connected final states. To save memory usage, set *RealtimeDelete* in *saddle_point*->*ValidSPs* to True. If needed, you may have to request high memory clusters.

SearchBuffer

SPS including buffer atoms, defaults to False

NMax4Trans

Maximum number of dimer translation steps, defaults to 1000

TrialStepsize

Trial step size for dimer translation, defaults to 0.015

MaxStepsize

Maximum step size for dimer translation, defaults to 0.05. The *MaxStepsize* must be larger than the *TrialStepsize*. Otherwise, the dimer search will fail to find the saddle point

RatioStepsize

Defined as the ratio between *MaxStepsize* and *TrialStepsize*

DecayStyle

Decay style of adaptive translation step size in SPS, defaults to *fixed*. Three styles are available *fixed*, *staircase,* or *continuous* (only the first three letters matter).

DecayRate

Base number (< 1.0) in a power function for getting the decay rate at a given translation step, defaults to 0.71

DecaySteps

Translation steps for getting the power number at a given translation step in the power function (see below), defaults to 20

MinStepsize

Min translation step size in SPS, defaults to 0.003
The dimer translation step size is adaptive. If *DecayStyle* is *fixed*, two translation stepsizes, *TrialStepsize* and *MaxStepsize*, are remained unchanged throughout SPS. If *DecayStyle* is *continuous*, $Decayrate\_i = DecayRate^{(i/DecaySteps)}$, where $Decayrate\_i$ is the decay rate at i^{th} translation step. Then the trial step size at i^{th} translation step, $TrialStepsize\_i$, is computed by

$$TrialStepsize\_i = TrialStepsize * Decayrate\_i + MinStepsize.$$

The maximum step size at i^{th} translation step is computed by

$$MaxStepsize\_i = MaxStepsize * Decayrate\_i + MinStepsize * RatioStepsize.$$

If *DecayStyle* is *staircase*, $Decayrate\_i = DecayRate^{(\mathrm{int}(i/DecaySteps))}$.

DimerSep

Separation distance between the dimer images, defaults to 0.005

TransHorizon

Translate dimer horizontally? Defaults to True

En4TransHorizon

Energy threshold for *TransHorizon*, defaults to 0.1

CheckAng

Check angle for earlier termination? Defaults to True

CheckAngSteps

Translation steps for *CheckAng*, defaults to 50

AngCut

Angle cut for *CheckAng*, defaults to 2.0 degrees.
If *CheckAng* is True, the angle check will be performed every *CheckAngSteps*. If the angle between the displacements of a given ongoing SPS and any found saddle point is smaller than *AngCut*, that SPS will be terminated.

IgnoreSteps

Number of ignored translation steps for a SPS, defaults to 4

NMax4Rot

Maximum number of dimer rotation steps, defaults to 3

FThres4Rot

Force threshold to rotate dimer forward, defaults to 0.1

FMin4Rot

Rotation force convergence for stopping dimer rotation, defaults to 0.001

FConv

Force convergence for stopping SPS, defaults to 1e-6

EnConv

Energy convergence for stopping SPS, defaults to 1e-5

DRatio4Relax

Ratio to *DimerSep* of the dimer at saddle point, Defaults to 2.0
The dimer images at the last step of a SPS are used for local relaxation. This is to tweak the two images before local relaxation. The modified $R_1 = R_0 +$ *DimerSep***DRatio4Relax* and $R_2 = R_0 -$ *DimerSep***DRatio4Relax*.

Tol4Connect

Tolerance of MAXABSC for checking the connectivity, defaults to 0.1

R2Dmax4SPAtom

Ratio to the maximum atomic displacement of a SP (see below), defaults to 0.04

DCut4SPAtom

Distance cutoff of the atomic displacement to consider an atom is displaced in the saddle point configuration, defaults to 0.01

DynCut4SPAtom

Use a dynamic distance cutoff of the atomic displacement to consider an atom is displaced in the saddle point configuration? Defaults to False

If *DynCut4SPAtom* is True, the cutoff distance (*dcut*) is selected such that

$\int_{dcut}^{\infty} p(\boldsymbol{r})d\boldsymbol{r} = 1/N$, where N is the number of active atoms. The $p(\boldsymbol{r})$ is a distribution function that is exponentially decayed with the standard deviation (s) of the atomic displacement array (D):

$$p(r) = \frac{1}{s}e^{-\frac{r}{s}}$$

This is a way to remove the background fluctuations and find the important data point within the database with a long-tail distribution, which is the case of atomic displacement distribution of the saddle point in most applications. The chosen *dcut* is to make sure at least one atom is involved in the saddle point configuration.

If *DynCut4SPAtom* is False, the *dcut* is the maximum value between *R2Dtot4SPAtom***dmax* and *DCut4SPAtom*, where *dmax* is the maximum atomic displacement.

If the absolute values of three displacements of an atom is larger than *dcut*, this atom is considered as an atom which is involved in saddle point. This information is to count the number of atoms that are displaced (*nad*) in a saddle point (*nad*, *nadfs*, *nadf* in *KMC_istep_SPs.csv* and *KMC_istep_Deleted_SPs.csv*).

ActiveOnly4SPConfig

Only store active atoms for SPs, defaults to True.

If *ActiveOnly4SPConfig* is False, it will store displacements of *active* and *buffer* atoms if *SearchBuffer* is True.

ShowIterationResults

Show the iteration results during a SPS attempt? Defaults to False

Inteval4ShowIterationResults

The translation step inteval for *ShowIterationResults*, defaults to 1

ShowVN4ShowIterationResults

Show the *VN* at every *Inteval4ShowIterationResults* translation step for *ShowIterationResults*? Defaults to False

ShowCoords4ShowIterationResults

Show the dimer coordinates at every *Inteval4ShowIterationResults* translation step for *ShowIterationResults*? Defaults to False

The above four parameters are used to output the information at every *Inteval4ShowIterationResults* translation step during a SPS attempt. If *ShowIterationResults* is True, a folder named "*ITERATION_RESULTS*" will be created at working directory and a series of folders named "*idav_idsps*" will be created inside the "*ITERATION_RESULTS*" folder. A summary file named "summary_array.csv" will be output inside the corresponding "*idav_idsps*" folder. If *ShowVN4ShowIterationResults* is True, a series of files named "*VectorN_TranslationStep_Iteration.csv*", which contains the information of the *VN* at the *Iteration* (number of force calls) of the *TranslationStep*, will be output inside the corresponding "*idav_idsps*" folder. If *ShowCoords4ShowIterationResults* is True, a series of files named "*Coords_TranslationStep_Iteration.csv*", which contains the reaction coordinates at the *Iteration* (number of force calls) of the *TranslationStep*, will be output inside the corresponding "*idav_idsps*" folder. Please see ITERATION_RESULTS in OUTPUT for details. They are normally used to adjust the values in *spsearch*.

insituGuidedSPS

LocalRelax

LocalRelax

Local relaxation of the SP? Defaults to True

InitTemp4Opt

Initialize velocity with temperature of *InitTemp4Opt*, defaults to 0.0

TargetTemp4NVT

Target temperature for NVT ensemble, defaults to 5.0

NVTSteps4Opt

Steps for NVT ensemble, defaults to 1000

Similar to *Relaxation* in *force_evaluator*, if *InitTemp4Opt* >*TargetTemp4NVT*, it will apply an initial velocity on active atoms for local relaxation to help the initial/final state moving out of the local minimum.

Preloading

Preload the displacements of SPs for a given active volume. The difference of *Preload* here from *Preload* in *defect_bank* is listed below:

1. The associated basin structure is the active atoms of the AV, if the basin file is not given. If a basin file is given and *CheckSequence* is True, this basin file is assumed to be slightly different from the AV and the sequence of atoms might change. This *CheckSequence* is to change the sequence of preloading displacements of SPs.
2. To load the displacements from *defect bank*, which is a list of defect bank instances, the code will judge whether the AV is the same basin structure with the basin structures in the list of *defect bank* instances. If the same basin structure is found, then load the displacements of SPs associated with that basin structure of the *defect bank* instance.
3. The file format used here is same with saved *defect bank* instances.
4. Preloading displacements are read from *defect bank* instances first.

Preload

Preloading the displacements for SP search? Defaults to False

LoadPath

Path to preload displacements, defaults to False

Method

Files or *Settings*, defaults to *Files*

Ratio4DispLoad

Ratio for loading displacements, defaults to 0.8

Scaling

Scaling factor of the coordinates of atoms in basin and displacement files, defaults to 1.0

SortDisps

Loading displacements by filename sequence, defaults to False

FileHeader

File header of preloading displacements, default to "SPS_AV_".
These preloading displacements will add to preloading displacements from defect bank.

CheckSequence

Manually check the atom sequence in this active volume and data for preloading? defaults to False

FileHeader4Data

File header for preloading data, defaults to "SPS_basin_"

The defaulted atom sequence of preloading displacements is the same as this active volume. If *Method* is *Files*, defaulted filenames for reading displacements are *SPS_AV_idav_disp_idisp.csv*. The *idav* means ID of the active volume. The *idisp* means ID number of preloading displacements for the given AV. The format of files is *csv* with keywords of "*index*, *dx*, *dy*, *dz*".
If the atom sequence is different, you need to include the data file that is used to generate the preloading displacements in *LoadPath* and set *CheckSequence* to True. The file name of the data file must contain *FileHeader4Data*+str(idav). As an example, a valid data file is *SPS_basin_0_Oh.csv*. the file is in csv format with "*index*, *type*, *x*, y, *z*".
If *Method* is *Settings*, defaulted filename for settings are *FileHeader* + "idav.json". The code will load the settings and generate the preloading displacements on the fly. A sample input file for preloading with *Settings* on a first AV, *SPS_AV_0.json*, is provided below.

SPS_AV_0.json:

[{"Selection": "All", "Operation": "translate", "Values": [0.0, 0.0, 0.2]}, {"Selection": ["type", "x"], "Range": [[1, 1],[0.0, 5.0]], "Operation": "rotation", "Values": [5.0, 0.0, 0.0]}]

The file is in JSON format (a list of dictionaries). The available keys and options are list below:

Key/Options:

Selection: "all", ["type", "id", "x", "y", "z", "dxy", "dxz", "dyz", "dxyz"]
Range: [[min, max], [min, max] ...]
Operation: "translation" or "rotation"

Values: [x, y, z] in *angstrom or [alpha, beta, gamma] in degree*

Please note that the defaulted coordinates for AV are centered with the defect and sorted by [*D*, *X*, *Y*, *Z*], "id" is ranged from 1 to number the of atoms.

HandleVN

CheckAng4Init

Check angle for *VN* initialization? Defaults to True

AngTol4Init

Angle tolerance for initializing *VN*, defaults to 5.

MaxIter4Init

Maximum iteration for *VN* initialization, defaults to 20

NMaxRandVN

Maximum number of random *VN*s in database, defaults to 20

CheckAng4Init, *AngTol4Init*, *MaxIter4Init* and *NMaxRandVN* are used for *VN* initialization, which normally is a random unit vector. If *CheckAng4Init* is True and the angle between this *VN* and any previously initiated *VN*s in database is smaller than *AngTol4Init*, the program will regenerate a random unit vector for this SPS until the angle is larger than *AngTol4Init*. If the iteration of the re-initialization process is larger than *MaxIter4Init*, the last *VN* will be assigned to a dimer and the SPS will launch anyway. If the number of *VN*s in database is smaller than *NMaxRandVN*, this *VN* will be saved to the database.

ResetVN04Preload

Reset VN to the initial *VN* (VN_0)? Defaults to True

RatioVN04Preload

Ratio of initial *VN* to preloaded displacements, defaults to 0.2

ResetVN04Preload and *RatioVN04Preload* are used only if the SPS has a preloading displacement array. With the preloading displacement, the dimer might be very close to saddle point. The random unit vector might be far away from the real orientation of the dimer at the saddle point. The real orientation of the dimer at the saddle point should be largely in line with the preloading displacements. If *ResetVN04Preload* is True, *VN*_0 = *VN*_0**RatioVN04Preload* + preloading displacements, and it is normalized to a unit vector.

IgnoreSteps

Ignore translation steps for modifying *VN*, defaults to 4

CenterVN

Center VN? Defaults to False

NSteps4CenterVN

Number of translation steps to Center *VN*, defaults to 5

RescaleVN

Rescale *VN*? Defaults to True

RescaleValue

Value for Rescaling *VN*, defaults to *LOGN*

The available values for rescaling include the reverse atom sequence (*RAS*), natural logarithm value of the modified relative atomic displacement (*LOGN*), and a combination of *RAS* and *LOGN* (*LOGNRAS*). If the shape of AVs is sphere-like. *LOGNRAS* is recommended.

Int4ComputeScale

Iteration interval for computing rescale factors, defaults to 1

TakeMin4MixedRescales

Take minimum value of mixed rescaling factors? Defaults to True

RescaleStyle4LOGV

Rescaling style for *LOGN*, defaults to *SIGMOID*, available options are *SIGMOID* or STEP

Period4MA

Period for moving average, defaults to 1

XRange4LOGV

Input value range of rescaling function using *LOGN,* defaults to 20.0

Ratio4Zero4LOGV

Ratio to the range of zero point of rescaling function using *LOGN*, default to 0.2. Beside a float number between 0 and 1.0, the “median” and “mean” values are also available with the keywords “median” and “mean”, respectively. This number is to adjust the zero point of the input values for the rescaling function:

$VX_{\mathrm{zero}} = VX_{\mathrm{min}} + (VX_{\mathrm{max}}-VX_{\mathrm{min}})$**Ratio4Zero4LOGV*, where VX_{min} and VX_{max} are the minimum and maximum values of the input values, respectively.

MinValue4LOGV

The power value on the natural number that is added to *LOGN*, defaults to -20

MinSpan4LOGV

Minimum span of *LOGN* to use a rescaling function, defaults to 4.0. If the difference between maximum and minimum input values is smaller than *MinSpan4LOGV*, no rescaling factor will be applied.

PowerOnV

Power on *VN* or summed *VN*, defaults to 4.0

RescaleStyle4RAS

Rescaling style for *RAS*, defaults to *SIGMOID*. Available options are *SIGMOID* or *STEP*

XRange4RAS

Input value range of the rescaling function using *RAS,* defaults to 40.0

Ratio4Zero4RAS

Ratio to the range of zero point of rescaling function using *RAS*, defaults to 0.3. Beside a float number between 0 and 1.0, the "median" and "mean" values are also available with the keywords "median" and "mean", respectively. Like to the *Ratio4Zero4LOGV* above, it is used to adjust the zero point of the input values for the rescaling function.

MinSpan4RAS

Minimum span of *RAS* to use rescaling function, defaults to 40.0. If the difference between maximum and minimum *RAS* is smaller than *MinSpan4RAS*, no rescaling factor will be applied.

force_evaluator

Force evaluator for SPS, default to "*force_evaluator*; but *nproc* default to 1; *processors* defaults to False; and *partition* is defaults to False.

nproc

Number of processors per SPS, defaults to 1

processors

Command *processors* in LAMMPS, defaults to False

partition

Command *partition* in LAMMPS, defaults to False

Rinput

Input file for SPS, defaults to False

RinputOpt

Input file for local AV relaxation, defaults to False

RinputDM

Input file for dynamic matrix, defaults to False

For custom *Rinput*, *RinputOpt*, and *RinputDM*, please note that the *read_data* is always *tmp0.dat*, and *write_data* is always *tmp1.dat*. The *atom_style* is expanded to include "*molecule-ID*". You need to specify the number of active atoms and the number of fixed atoms in the input file. For example, you need to include the following lines at proper place.

- "group gactive id <= %d" % (nactive)
- "group gfixed subtract all active"
- "fix freeze gfixed setforce 0.0 0.0 0.0"

The *nactive* is the number of active atoms in the AV. If *SearchBuffer* is True, SPS will first perform on active atoms and then perform on active and buffer atoms in the AV. Thus, the *nactive* is the number of active and buffer atoms in the AV at the second stage of SPS.

RinputDM is used to compute dynamic matrix. Please follow the LAMMPS command "dynamic_matrix" to create your own input file for *RinputDM*. Please note that this code will compute the dynamic matrix on active atoms only.

FixTypes

The types of atoms that are fixed during a SPS, default to False

FixAxesStr

The axes of the fixed atoms by *FixTypes*, default to "ALL", which means [0, 1, 2] for x, y, and z axes.

Besides the fixed atoms in the fixed layer, there are additional fixed atoms inside the active layer of an AV in some cases. For example, the AV consists of a dislocation and a rigid ball (stay fixed throughout SPS). In this case, these fixed atoms need to assign to different types in the data file. The *FixTypes* should be a list of types staying fixed in SPS and *FixAxesStr* should be a list of strings, which has the same length of *FixTypes* and the string inside *FixAxesStr* should be the axis index numbers among 0, 1, and 2 separating with a comma. For instances, "0, 2" and "0, 1, 2" are valid strings for *FixAxesStr*. If *FixAxesStr* is "ALL", all three axes will be fixed, which is equivalent to "0, 1, 2" input for each type in *FixTypes*. Below we show the details of codes do for a sample settings as:

FixTypes:
  - 3
  - 4

FixAxesStr:
  - 0, 2
  - 1

Inside LAMMPS input script,

```
…
group gactive id <= nactive # nactive is the number of active atoms of an AV
group gfixed subtract all active

group gfixt1 type 3 #set type 3 to be fixed
group gfixt1 intersect gfixt1 gactive #find the intersect between gfixt1 and active
group gactive subtract gactive gfixt1 #update group active
fix ffix1 gfixt1 setforce 0.0 NULL 0.0 #set force of the corresponding axes to 0.0

group gfixt2 type 4
group gfixt2 intersect gfixt2 gactive
group gactive subtract gactive gfixt2
fix ffix2 gfixt2 setforce NULL 0.0 NULL

fix freeze gfixed setforce 0.0 0.0 0.0
…
```

For more complex setup of using LAMMPS force evaluator for an AV, you can provide custom input scripts (*Rinput*, *RinputOpt*, and *RinputDM*) at *force_evaluator* of *spsearch*. Please note that the "gactive" and "gfixed" are two used groups in the code and they must be defined as above. The codes can automatically update the folder to *Runner_color* and nactive to nactive of the AV that is performing the simulation.

TaskDist

How to distribute the tasks? Options are "*AV*" or "*SPS*", defaults to "*AV*".
Only used in parallel version. The total number of tasks (*ntask_tot*) for SPS is *nAV*nSPS*, where *nAV* is the number of AVs and *nSPS* is the number of SPS attempts (*NSearch*) per AV. If *TaskDist* is *AV*, the task will distribute on AVs prior the *NSearch*. Otherwise, the task will distribute on *NSearch* first, then on AVs. In general, the *AV* style of task distribution has more time efficiency but consumes more memory than *SPS* style.

Master_Slave

Master-slave scheme? Defaults to True.
As the time cost for an SPS varies a lot, the default for using master-slave scheme is True.

saddle_point

A saddle point (SP) lays between the initial and final states as illustrated in Figure 3.9, where the *barrier* (*barr*) is the energy difference between the SP and initial state, and *ebias* is the energy difference between final and initial states. As a result, the *backward barrier* (*bbarr*) from final state to initial state is *barrier - ebias*.

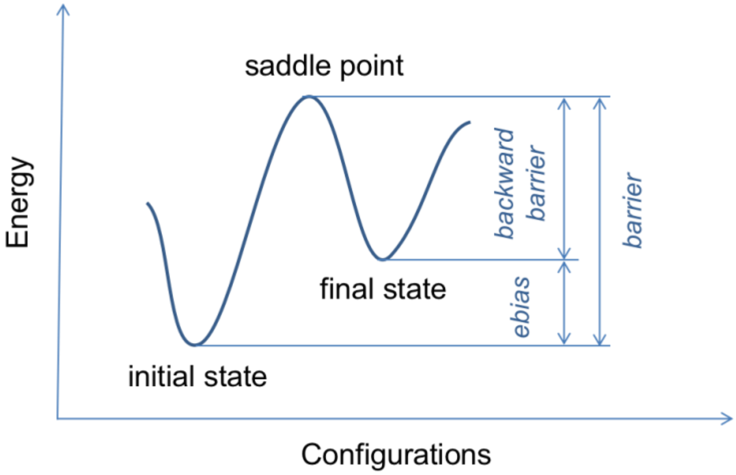


Figure 3.9. A schematic view of a saddle point (SP) between the initial and final states

In addition to energies, the displacement arrays of the SP (*Disp_SP*) and final state (*Disp_FI*) are stored. The displacement array between the SP and final state (*Disp_FS*) is computed by *Disp_FS* = *Disp_FI* – *Disp_SP*. The displacement array is a 3N matrix, where N is the number of active or (active+buffer) atoms in an AV depending on *ActiveOnly4SPConfig* in *spsearch*. At the first level of *saddle_point*, the values for validation are scalars, where "*_FI*" and "*_FS*" represent the displacement arrays of initial to final states and SP to final state, respectively. Below is the list of terms on displacement array (D).

Magnitude of D:

$$dmag = \sqrt{\sum_{i=0}^{2}\sum_{n=1}^{N} D_{i,n}^2}$$

Atomic displacement array (1xN):

$$atomic\ disp = \sqrt{\sum_{i=0}^{2} D_{i,n}^2}$$

Summation of D:

$$dsum = \sum_{i=0}^{2}\sum_{n=1}^{N} D_{i,n}$$

Summation of absolute D:

$$adsum = \sum_{i=0}^{2}\sum_{n=1}^{N} abs(D_{i,n})$$

Ratio between summation and absolute values of D:

$$dsumr = {}^{dsum}\!/_{adsum}$$

The ratio between dsum/adsum is used to measure the translation portion of displacement.

The validation of SPs on these scalar values of energies or displacements is such:
If the corresponding value > string end with "*Cut*", this SP is not valid.
If the corresponding value < string end with "*Min*", this SP is not valid.
The additional validation of SPs is given in *ValidSPs*.
The validation of SPs will be performed after all SPS attempts for a given AV are finished. The information of valid SPs is available in *KMC_istep_SPs.csv* and deleted (not valid) SPs is available in *KMC_istep_Deleted_SPs.csv* in *SPOut* directory.

BarrierCut

Barrier cut, defaults to 100.0

BarrierMin

Minimum barrier, defaults to 0.0

EbiasCut

Ebias cut, defaults to 100.0

EbiasMin

Minimum *ebias*, defaults to NA (no limit)

BackBarrierMin

Minimum backward barrier, defaults to 0.0. In some cases, the backward barrier might be negative after relaxation between KMC step, where the energetic is computed with the whole data. This will result in the structure return to the initial state after relaxation, which is not physical. You can set a threshold value to avoid this situation. But it may delete the true SP with a very small backward barrier. The safest way is to set *CalBarrsInData* and *CalEbiasInData* to True and then check the backward barrier after recalibrated with the whole data.

Prefactor

Vibrational prefactor for KMC, defaults to 1.0, unit THz. To calculate the prefactor, set *CalPrefactor* in *dynamic_matrix* to True.

CalBarrsInData

Recalibrate barriers in data? Defaults to False

CalEbiasInData

Recalibrate *ebias* in data? Defaults to False

Thres4Recalib

Barrier threshold for recalibrating the energies in data? Defaults to None

The barriers and *ebias* are calculated in the active volume. If *Thres4Recalib* is a float and the barrier of this SP – the minimum barrier of all SPs, the energies of this SP will be recalibrated with the whole data. If *Thres4Recalib* is NOT a float, the energies of all SPs will be recalibrated with the whole data.

DAtomCut

Maximum allowed atomic displacement, defaults to *cutneighmax*

DMagCut

Maximum magnitude of *Disp_SP*, defaults to nactive**DAtomCut*, where nactive is the number of active atoms

DMagMin

Minimum magnitude of *Disp_SP*, defaults to 0.0

DtotCut

Maximum value of the summed atomic displacement of *Disp_SP*, defaults to NA

DtotMin

Minimum value of the summed atomic displacement of *Disp_SP*, defaults to 0.0

DmaxCut

Maximum displacement value of the atom with the largest atomic displacement of *Disp_SP*, defaults to NA

DmaxMin

Minimum displacement value of the atom with the largest atomic displacement of *Disp_SP*, defaults to 0.0

DsumCut

Maximum value of the summation of *Disp_SP*, defaults to NA

DsumMin

Minimum value of the summation of *Disp_SP*, defaults to 0.0

DsumrCut

Maximum ratio between the summation and its absolute values of *Disp_SP*, defaults to NA

DsumrMin

Minimum magnitude ratio between the summation and its absolute values of *Disp_SP*, defaults to 0.0

DMagCut_FI

Maximu, magnitude of *Disp_FI*, defaults to NA

DmagMin_FI

Minimum magnitude of *Disp_FI*, defaults to 0.0

DtotCut_FI

Maximum value of the summed atomic displacement of *Disp_FI*, defaults to NA

DtotMin_FI

Minimum value of the summed atomic displacement of *Disp_FI*, defaults to 0.0

DmaxCut_FI

Max displacement value of the atom with the largest displacement of *Disp_FI*, defaults to NA

DmaxMin_FI

Min displacement value of the atom with the largest displacement of *Disp_FI*, defaults to 0.0

DsumCut_FI

Maximum value of the magnitude of a summation over *Disp_FI*, defaults to NA

DsumMin_FI

Minimum value of the magnitude of a summation over *Disp_FI*, defaults to 0.0

DsumrCut_FI

Maximum ratio between the summation and its absolute values of *Disp_FI*, defaults to NA

DsumrMin_FI

Minimum ratio between the summation and its absolute values of *Disp_FI*, defaults to 0.0

DMagCut_FS

Maximum magnitude of *Disp_FS*, defaults to NA

DmagMin_FS

Minimum magnitude of *Disp_FS*, defaults to 0.0

DtotCut_FS

Maximum value of the summed atomic displacement of *Disp_FS* defaults to NA

DtotMin_FS

Minimum value of the summed atomic displacement of *Disp_FS*, defaults to 0.0

DmaxCut_FS

Maximum displacement value of the atom with the largest displacement of *Disp_FS*, defaults to NA

DmaxMin_FS

Minimum displacement value of the atom with the largest displacement of *Disp_FS*, defaults to 0.0

DsumCut_FS

Maximum value of the summation of *Disp_FS*, defaults to NA

DsumMin_FS

Minimum value of the summation of Disp_FS, defaults to 0.0

DsumrCut_FS

Maximum ratio between the summation and its absolute values of *Disp_FS*, defaults to NA

DsumrMin_FS

Minimum ratio between the summation and its absolute values of *Disp_FS*, defaults to 0.0

ValidSPs

RealtimeValid

Realtime validating SPs? Defaults to False

If *RealtimeValid* is True, the SP will be validated immediately and the information of currently found SPs will be updated. Whether or not the SP is valid, it will be kept in the found SPs database for other usage, for example *CheckAng* in *spsearch*.

RealtimeDelete

Realtime deleting not valid SPs? Defaults to False

If *RealtimeDelete* is True, the SP will be validated immediately. If it is not valid, it will be deleted immediately. Otherwise, it will be kept in the found SPs database. Obviously, it is not wise to set both *RealtimeValid* and *RealtimeDelete* to be True. Set *RealtimeDelete* to True can save the memory usage. For large number of search attempts (*NSearch* in *spsearch*), it is recommended to set *RealtimeDelete* to be True.

CheckConnectivity

Check connectivity? Defaults to True

NScreenDisp

Number of performed Screening SPs based on the displacement matrix, defaults to 0

NScreenEng

Number of performed Screening SPs based on energies, defaults to 0 *toScreenDisp*

Choose which to apply *ScreenDisp*, *ALL*, *NotConn*, or *Conn*, defaults to *NotConn*

toScreenEng

Choose which to apply *ScreenEng*, *ALL*, *NotConn*, or *Conn*, defaults to *NotConn*

AND4ScreenDE

Place *AND* condition for *ScreenDisp* and *ScreenEng*, defaults to True. If *AND4ScreenDE* is True, a SP is not valid if both *ScreenDisp* and *ScreenEng* are True. If *AND4ScreenDE* is False, a SP is not valid if either *ScreenDisp* or *ScreenEng* is True.

ScreenDisp

A list of *ScreenDisp* conditions

Str4ScreenD

Structures for this *ScreenDisp*, *SP*, *FI*, or *FS*, defaults to *SP.* *SP*, *FI*, and *FS* are initial to a SP, initial to final state and SP to final state

Type4ScreenD

Displacement type for this ScreenDisp, defaults to DMAG

AbsVal4ScreenD

Is an absolute value? Defaults to True

MinVal4ScreenD

Minimum value required for this *ScreenDisp*, defaults to NA

MaxVal4ScreenD

Maximum value allowed for this *ScreenDisp*, defaults to NA

AND4ScreenD

Place *AND* condition for this *ScreenDisp*, defaults to True

ScreenEng

A list of *ScreenEng* conditions

Type4ScreenE

Energy type for *ScreenEng*, defaults to barrier.

Available options are *BARRier*, *BBARrier*, and *EBIA*s, which means barrier, backforward barrier, and energy difference between final and initial state, respectively. (Only the first four letters matter).

MinVal4ScreeE

Minimum value required for this *ScreenEng*, defaults to NA

MaxVal4ScreenE

Maximum value allowed for this *ScreenEng*, defaults to NA

AND4ScreenE

Place *AND* condition for this *ScreenEng*, defaults to True

ScreenDisp conditions are to screen SPs based on its displacement array, which is a 3N dimensional array. The *toScreenDisp* defines which SPs (connected or not) will have the *ScreenDisp* conditions applied. *nScreenDisp* refers to the number of screen conditions. If *nScreenDisp* > 0, *ScreenDisp* is a list of screen conditions.
The *SP* (initial->SP), *FI* (initial->final) and *FS* (SP->final) in *Str4ScreenDisp* means the displacement array is used.

The keywords *dmag*, *dmax, vmax*, and *dsum* inside *ScreenDisp* are the 3-component vectors with "x", "y", and "z" directions. Below is the list of the definitions of these 3-component vectors, where *i* = 0, 1 or 2 standing for "x", "y", or "z" directions.

Partition of the magnitude of D:

$$dmag\_i = \sqrt{\sum_{n=1}^{N} D_{i,n}^{2}}$$

Partition of the displacements of the atom with the largest atomic displacement:

$$dmax\_i = \max\,(atomic\ displacement)$$

Partition of the maximum absolute values of the displacements in each direction:

$$vmax\_i = \max\,(absolute(D_{i,n}), axis = 1)$$

Partition of the Summation of D:

$$dsum\_i = \sum_{n=1}^{N} D_{i,n}$$

To measure the translation portion of the displacement arrays, the partition vector of the summation of the absolute values (*adsum_i*) is introduced, and thus the ratio between *dsum_i* and *adsum_i* is computed as below:

Partition of Summation of absolute D:

$$adsum\_i = \sum_{n=1}^{N} absolute(D_{i,n})$$

Ratio between summation and absolute values of D:

$$dsum\_i\_rabs = {}^{dsum\_i}\!/_{adsum\_i}$$

The available options for *Type4ScreenD* are *dmag*, *dmagx*, *dmagy*, *dmagz*, *dmag_dmin*, *dmag_dmax*, *dmag_drx*, *dmag_dry*, *dmag_drz*, *dmag_drxy*, *dmag_drxz*, *dmag_dryz*, *dmax*, *dmaxx*, *dmaxy*, *dmaxz*, *dmax_dmin*, *dmax_dmax*, *dmax_drx*, *dmax_dry*, *dmax_drz*, *dmax_drxy*, *dmax_drxz*, *dmax_dryz*, *vmax*, *vmaxx*, *vmaxy*, *vmaxz*, *vmax_dmin*, *vmax_dmax*, *vmax_drx*, *vmax_dry*, *vmax_drz*, *vmax_dyxy*, *vmax_drxz*, *vmax_dryz*, *dsum*, *dsumx*, *dsumy*, *dsumz*, *dsum_dmin*, *dsum_dmax,* *dsum_drx*, *dsum_dry*, *dsum_drz*, *dsum_drxy*, *dsum_drxz*, *dsum_dryz*, *dsum_rabs*, *dsumx_rabs*, *dsumy_rabs*, and *dsumz_rabs*.

There are 12 entries for each of four partition vectors (*dmag_i*, *dmax_i*, *vmax_i*, and *dsum_i*). Taking *dmag_i* as an example, the meaning of each entry is given below:

dmag: magnitude of *dmag_i*

dmagx, *dmagy*, *dmagz*: values for each direction

dmag_dmax, *dmag_dmin:* direction of maximum or minimum value, in which 0-'x', 1-'y' and 2-'z'

dmag_drx, *dmag_dry*, *dmag_drz*: ratio between each direction to its magnitude

dmag_drxy, *dmag_drxz*, *dmag_dryz*: ratio between 'x' and 'y', between 'x' and 'z', and between 'y' and 'z' directions.

The *dsum_rabs*, *dsumx_rabs*, *dsumy_rabs*, and *dsumz_rabs* are the ratio between summation and absolute values of a displacement array ($dsum\_i\_rabs$).

If *AbsVal4ScreenD* is True, the value will be the absolute value.

If the value >= *MinVal4ScreenD* and the value <= *MaxVal4ScreenD*, this screen condition is satisfied. If either *MinVal4ScreenD* or *MaxVal4ScreenD* is "NA", the corresponding side in the "if" statement above is deactivated. If both of them are "NA", this screen condition is unsatisfied.

If *AND4ScreenD* is True and *ScreenDisp* from previous screen conditions is False, this screen condition in the list will be applied.

If *AND4ScreenD* is False and *ScreenDisp* from previous screen conditions is True, this screen condition in the list is to revoke the previous screen conditions. That is to say, *ScreenDisp* will change to False from True, if this screen condition is satisfied.

If the final *ScreenDisp* is True, the SP is not valid from the *ScreenDisp* conditions.

Likewise, *ScreenEng* conditions are to screen SPs based on their energies. *toScreenEng* means which SPs (connected or not) will apply the *ScreenEng* conditions. *nScreenEng* refers to the number of screen conditions. If *nScreenEng* > 0, *ScreenEng* is a list of screen conditions. Available options for *ScreenEng* are barrier (*BARR*), backward barrirer (*BBAR*) or ebias (*EBIA*). If *AND4ScreenE* is False and *ScreenEng* is True, this screen condition revokes previous screen conditions in the list.

If the final *ScreenEng* is True, the SP is not valid from the *ScreenEng* conditions.

Please refer to *AND4ScreenDE* above after *ScreenDisp* and *ScreenEng* are obtained.

MaxRatio4Dmag

Maximum ratio between the magnitude and the smallest magnitude, defaults to NA

MaxRatio4Barr

Maximum ratio between the barrier and the smallest barrier, defaults to NA

If *CheckConnectivity* is True, this SP is not valid if it is not connected. If *CheckConnectivity* is False, all SPs will be regarded as connected.

Minimum values of barriers and *dmag* of *Disp_SP* within an AV are computed on the fly. If a SP > *MaxRatio4Dmag* or >*MaxRatio4Barr*, this SP is not valid.

Additional screening on SPs is available at *ScreenDisp* and *ScreenEng* in *ValidSPs*.

All criteria will not be applied if the corresponding threshold values are a string ("NA")

GroupSP

Group saddle points (SP) in an active volume, defaults to False

AngCut4GSP

Angle cutoff for *GroupSP*, defaults to 10.0 degrees

MagCut4GSP

Magnitude cutoff for *GroupSP*, defaults to 0.1

EnCut4GSP

Barrier cutoff for *GroupSP*, defaults to 0.1

The group SP within an active volume is to group SPs within the same AV. If two SPs have satisfied GSP criteria (*AngCut4GSP*, *MagCut4GSP*, and *EnCut4GSP*), they will be merged into one SP with longest displacement, summation of prefactors, and highest barrier.

EnTol4AVSP

Energy tolerance for checking the duplicated SPs within the same active volume, defaults to 0.1

Tol4AVSP

Displacement tolerance for checking the duplicated SPs within the same active volume, defaults to 0.1

Within the same AV, if the energy difference of SPs is smaller than *EnTol4AVSP* and the maximum difference between displacements of SPs is smaller than *Tol4AVSP*, they will be considered as the same saddle point.

FindSPType

Finding the type of SPs? Defaults to False

AngCut4Type

Angle cutoff for SP type, defaults to 5

MagCut4Type

Magnitude cutoff for SP type, defaults to 0.05

LenCut4Type

Length cutoff for SP type, defaults to 0.05

EnCut4Type

Energy cutoff for SP type, defaults to 0.05

The type of saddle points is determined by the saddle point lattice, which is a 3x3 array formed by the dot product between displacement array (D) of the SP and the transformed coordinates ($\boldsymbol{R}$) of the initial structure ($D \cdot \boldsymbol{R}^T$). Analogizing to the Bravais lattice, if the saddle point lattices from two SPs have satisfied group type criteria (*AngCut4type*, *MagCut4type*, *LenCut4Type*, and *EnCut4type*), they will be considered one type.

NMax4Dup

Maximum number of atoms for checking duplicated SPs from different active volumes, defaults to 600

NCommonMin

Minimum number of common atoms used in checking duplicated SPs from different active volumes, defaults to 10.

R2Dmax4Tol

Ratio of maximum atomic displacement for tolerance of displacement difference between two SPs, defaults to 0.1

Tol4Disp

Tolerance of the displacement difference between two SP, defaults to 0.1 angstroms

Checking duplicated SPs from different active volumes has 4 steps: 1) whether these two AVs are neighbors? If not, all SPs inside these two AVs are considered as different; 2) Whether the number of common atoms is larger than a threshold value? If not, all SPs inside these two AVs are considered as different; 3) Whether the MAXABSC of the displacement vector of the uncommon atoms are smaller than a threshold value? If not, these two SPs are different. Otherwise, 4) Whether the MAXABSC of the displacement vector of the common atoms are smaller than a threshold value? If not, these two SPs are considered as different. Otherwise, they are duplicated SPs.

The threshold value in Query 2 is max(*NCommonMin*, 0.5*min(N1, N2)), where N1 and N2 are the number of active atoms in two AVs. If N1 or N2 is larger than *NMax4Dup*, then only the first *NMax4Dup* atoms are under consideration for the duplication check from two AVs. The threshold value in Query 3 is min(*R2Dmax4Tol***dmax*, *Tol4Disp*), where *dmax* is the maximum atomic displacement of the SP. The threshold value in Query 4 is min(*R2Dmax4Tol***dmax1*, *R2Dmax4Tol***dmax2*, *Tol4Disp*), where *dmax1* and *dmax2* are the maximum atomic displacements of two SPs.

visual

Parameters involved in log file and output files of SEAKMC_py. The code will always output the summary file "***Seakmc_summary.csv***". The detail of information of output files is given in *OUTPUT*.

Screen

Screen output? Defaults to True

Log

Output "***Seakmc.log***" log file? Defaults to True

Write_SP_Summary

Output SEAKMC_py SP summary file? Defaults to True.
The filenames are *KMC_istep_SPs.csv* and *KMC_istep_Deleted_SPs.csv* inside "SPOut" folder.

Write_Data_SPs

Write_Data_SPs

Write SPs in data? Defaults to True

Write_KMC_Data

Write data after KMC step? Defaults to True, filename *KMC_istep.dat*

Write_Data_AVs

Write active volumes in data? Defaults to True, filename *KMC_istep_Data_AVs.dat*

Write_Prob

Write probabilities of all SPs? Defaults to True, filename *KMC_istep_Prob.csv*.

DetailOut

Write detailed information of SPs? Defaults to True

SPs4Detail

SPs for detailed information, defaults to *Auto*
Available options of *SPs4Detail* are *Auto* and *ALL*. *Auto* means the selected SPs in current basin of this KMC. The filename of these SPs is *KMC_istep_DetailSPs.csv*.

DispStyle4DataSP

Options are "SP" or "FI" or "Both", defaults to "Both". Using displacements of the saddle point or final state or "Both" when writing SPs of the data

OutputStyle

Stack or *Separate*, defaults to *Separate*

Sel_iSPs

Selected SPs to be written to Write_Data_Sel_SPs, defaults to *Auto*
All: write all SPs
Auto: write the selected events

“0, 1, 3”: user input list of SP local ID

Offset

An integer number to offset isp, defaults to 0

DataOutPath

Path to output structure files, fixes to “DataOut”

SPOutPath

Path to output “csv” files, fixes to “SPOut”

If *MRM* is used, it is common for the selected event is from one of basins of the superbasin rather than the current basin. In this case, the *Offset* will be **10000** if *Sel_iSPs* is *Auto*. There is no output for *DetailOut*.
If *OutputStyle* is *Stack*, the visible atoms will append to the original structure. The filename is *KMC_istep_Data_Selected_SPs.dat* and *KMC_istep_Data_Selected_FIs.dat*.
If *OutputStyle* is *Separate*, every saddle point will output a file. The filename is *KMC_istep_Data_SP_(offset+isp).dat* and *KMC_istep_Data_FI_(offset+isp).dat*.

Write_AV_SPs

Write_AV_SPs

Write SPs of an active volume to active volume? Defaults to False, filename *KMC_istep_AV_idav_SPs.dat*. This tag writes stacking SPs of an active volume using local information of the active volume

Write_Local_AV

Write active volume? Defaults to False, filename *KMC_istep_Data_AV_idav.dat*.

Write_Data_AV_SPs

Write the SPs of an active volume to active volume? Defaults to False, filename *KMC_istep_Data_AV_idav_SPs.dat*. Writing stacking SPs of an active volume using the information contained in the current data.

AVOutPath

Path to output files, defaults to “AVOut”

DispStyle4AVSP

Choices are *SP* or *FI* or *Both*, default to *SP*. Using displacements of the saddle point or final state or *Both* when writing the SPs of an AV

RCut4Vis

Ratio of total SP’s displacement for atom visibility, defaults to 0.04

DCut4Vis

Displacement cutoff for visibility, defaults to 0.01
If the maximum absolute value of three displacement components of an atom < max(*RCut4Vis***dmax*, *DCut4Vis*), where *dmax* is the maximum atomic displacement, this atom is invisible in output structures.

This visibility based on atom displacement is only used if *OutputStyle* is *Stack*. If *OutputStyle* is *Separate*, all the active atoms will be visible.

Invisible

Are irrelevant atoms invisible? Defaults to True

Reset_Index

Reset index of output atoms? Defaults to False

ShowBuffer

Show buffer atoms? Defaults to False

ShowFixed

Show fixed atoms? Defaults to False

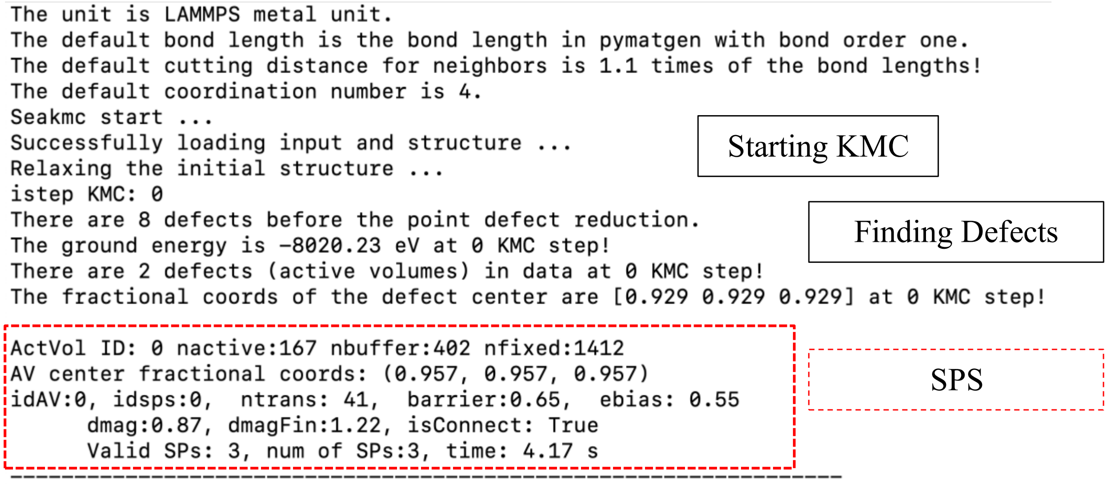
The *ShowBuffer* and *ShowFixed* are only used in writing the AV (*Write_Data_AVs* and *Write_Local_AV*).

OUTPUT

The folders and filenames use the default values. The output includes *Seakmc.log* -- SEAKMC_py log file, *Seakmc_summary.csv*, the restart files, the information of saddle points in *SPOut*, the structure information in *DataOut*, the defect bank information in *DefectBank*, the structure information of active volumes in *AVOut*, and the iteration information at every *Inteval4ShowIterationResults* translation step during a SPS attempt in *ITERATION_RESULTS*.

Throughout the document, *istep* is the ID number for the KMC step, *idav* or *idAV* is the ID number for the active volume, *idsps* is the ID number of SPS and *isp* is local ID number of SPs.

Seakmc.log



```
The unit is LAMMPS metal unit.
The default bond length is the bond length in pymatgen with bond order one.
The default cutting distance for neighbors is 1.1 times of the bond lengths!
The default coordination number is 4.
Seakmc start ...
Successfully loading input and structure ...
Relaxing the initial structure ...
istep KMC: 0
There are 8 defects before the point defect reduction.
The ground energy is -8020.23 eV at 0 KMC step!
There are 2 defects (active volumes) in data at 0 KMC step!
The fractional coords of the defect center are [0.929 0.929 0.929] at 0 KMC step!

ActVol ID: 0 nactive:167 nbuffer:402 nfixed:1412
AV center fractional coords: (0.957, 0.957, 0.957)
idAV:0, idsps:0,  ntrans: 41,  barrier:0.65,  ebias: 0.55
      dmag:0.87, dmagFin:1.22, isConnect: True
      Valid SPs: 3, num of SPs:3, time: 4.17 s
-----------------------------------------------------------------
```

Figure 4.1. First part of the Seakmc.log file

As shown in Figure 4.1, after relaxing the initial structure, SEAKMC_py starts with finding defects. The information in log file includes: 1) the number of defects before the point defect reduction; 2) the ground energy of the data; 3) the number of defects or active volumes; and 4) the fractional coordinates of the defect center (simple mean values of all defects without PBC consideration).

SPS starts with the first active volume. The logfile outputs: 1) numbers of active, buffer and fixed atoms; 2) the defect center of the active volume (mean values with PBC consideration) of the AV. After an SPS attempt is done, the output includes:

idAV: ID numbers of active volumes

idsps: ID numbers of SPS attempts

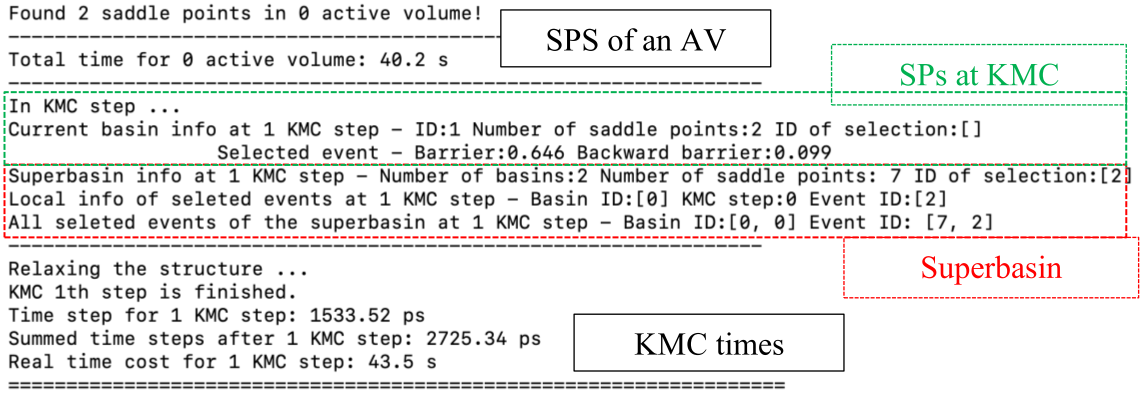
ntrans: number of dimer translation steps

barrier: energy barrier of the found SP. If the SP is not found or not valid, barrier will be *BarrierCut* +10.0.

ebias: energy bias between initial and final states. If the SP is not found or not valid, barrier will be *BarrierCut* +10.0. If *LocalRelax* is False, *ebias* = 0.

dmag: the magnitude of displacement from the initial state to SP

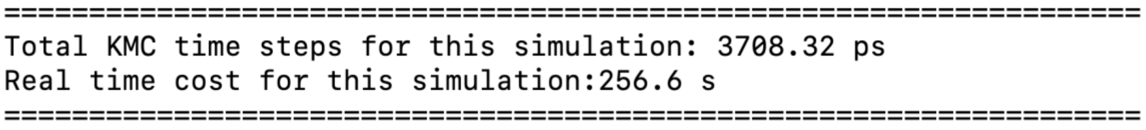
dmagFin: the magnitude of displacement from initial to finial states
isConnect: connectivity of the found SP
Valid SPs: number of valid SPs in this AV
num of SPs: number of SPs in this AV. If *PointGroupSymm* is True, the equivalent SPs will be generated based on the found SP and symmetry operations.
time: time cost for the SPS.



```
Found 2 saddle points in 0 active volume!
------------------------------------------------------------------
Total time for 0 active volume: 40.2 s
------------------------------------------------------------------
In KMC step ...
Current basin info at 1 KMC step - ID:1 Number of saddle points:2 ID of selection:[]
                  Selected event - Barrier:0.646 Backward barrier:0.099
Superbasin info at 1 KMC step - Number of basins:2 Number of saddle points: 7 ID of selection:[2]
Local info of seleted events at 1 KMC step - Basin ID:[0] KMC step:0 Event ID:[2]
All seleted events of the superbasin at 1 KMC step - Basin ID:[0, 0] Event ID: [7, 2]
------------------------------------------------------------------
Relaxing the structure ...
KMC 1th step is finished.
Time step for 1 KMC step: 1533.52 ps
Summed time steps after 1 KMC step: 2725.34 ps
Real time cost for 1 KMC step: 43.5 s
===================================================================
```

Figure 4.2. Second part of the Seakmc.log file

After an SPS is finished within an AV, the number of found SPs and time for SPS of this AV will be contained in the output. The information inside green box in Figure 4.2 shows the screen-printed information of SPs for a given KMC step. The superbasin data will be printed to the screen only if *MRM* is used, as shown red box in Figure 4.2. The final piece of information in Figure 4.2gives the simulation time of this KMC step, summed time over the previous KMC steps and time cost for this KMC step.



```
======================================================================
Total KMC time steps for this simulation: 3708.32 ps
Real time cost for this simulation:256.6 s
======================================================================
```

Figure 4.3. Third part of the Seakmc.log file

After all the KMC steps have finished, the total simulation time and time cost for this simulation are output as shown Figure 4.3.

Seakmc_summary.csv

Summary of **SEAKMC_py** output in ".csv" format. The list of keywords in the summary file is given below.

istep: The ID number of the current KMC step
ground_energy: The free energy of the current system
nDefect: Number of superdefects (active volumes) in the system

defect_center_xs: Fractional coordinate of the defect center in *x*
defect_center_ys: Fractional coordinate of the defect center in *y*
defect_center_zs: Fractional coordinate of the defect center in *z*
nBasin: Number of basins within the current superbasin
nSP_thisbasin: Number of saddle points within the current basin (data)
nSP_superbasin: Number of saddle points within the current superbasin
iSP_selected: ID number of the selected saddle point
forward_barrier: The barrier of the selected saddle point
ebias: The energy bias of the selected saddle point
backward_barrier: The backward barrier of the selected saddle point
time_step: KMC time step of the current KMC step
simulation_time: Summed KMC time steps until the current KMC step

Please note that the coordinates of the defect center within the current basin (data structure) are the mean values of all reduced defects. It might be misleading if the defects are at corner of a system with PBC.

If you have a custom input file for data relaxation between KMC steps, i.e. *RinputOpt* in *force_evaluator* is the input filename, you can import these values from SEAKMC_py to your *RinputOpt* file by following the instructions after *Keys4ImportValue4RinputOpt*.

RESTART_istep_idav.restart

Restart file written at the *istep* KMC step after the SPS finishes with *idav* AV. The restart is a Python *pickle* file, which includes istep, list of *defect bank* instances, superbasin (information of SPs and basins), *istep* and simulation time. If running from a restart file, the log file will append to *Seakmc.log* if it exists. Besides inherent information in the restart file, all the other parameters are allowed to change in ***input.yaml***.

SPOut

The information of SPs at *istep* KMC step. All are *Pandas dataframes* written in *csv* format, in which the first column is the index of the dataframe.

KMC_istep_SPs.csv

File containing valid SPs. The keywords are "*idav*", "*idsps*", "*type*", "*iters*", "*ntrans*", "*emax*", "*barrier*", "*prefactor*", "*ebias*", "*dtot*", "*dmag*", "*dmax*", "*dsum*", "*adsum*", "*nad*", "*dtotfs*", "*dmagfs*", "*dmaxfs*", "*dsumfs*", "*adsumfs*", "*nadfs*", "*dtotfi*", "*dmagfi*", "*dmaxfi*", "*dsumfi*", "*adsumfi*", "*nadfi*", and "*isConnect*".
The "*type*" is the type ID of SPs. The "*iters*" is the number of iterations (force calls) of the SPS attempt. The "*ntrans*" is the number of translation steps and the "*emax*" is the maximum potential energy of the SPS attempt (Figure 3.4). The "*barrier*" and "*ebias*" are energies of SPs (Figure 3.9). The "*prefactor*" is the vibrational prefactor in transition state theory.

The "*dtot*", "*dmag*", "*dmax*", "*dsum*", and "*adsum*" are scalar values of the displacement array between initial state to SP. Please refer to *saddle_point* for more information. The appendix of “fs” or “fi” indicates the displacement array from the SP to final state and from the initial to final states, respectively.

The “*nad*”, “*nadfs*”, and “*nadfi*” are the number of displaced atoms in displacement arrays from the initial state to SP, from the SP to final state and from initial to final states, respectively. The criteria of displaced atoms are given in *R2Dmax4SPAtom* and *DCut4SPAtom* in *spsearch*.

The “*isConnect*” is the connectivity of SPs.

KMC_istep_Prob.csv

Probability information of valid SPs. The sequence is identical with that of *KMC_istep_SPs.csv*. The keywords are "*idav*", "*idsps*", "*barrier*", "*prefactor*", "*ebias*", "*dmag*", "*dmax*", "*dsum*", "*rdsum*", "*dmagfi*", "*dmaxfi*", "*dsumfi*", "*dsumrfi*", "*isConnect*", "*probability*", and "*isSel*".

probability: probability distribution (%) of SPs

isSel: the selected event in KMC step

KMC_istep_Deleted_SPs.csv

Information of deleted SPs. The keywords are "*idav*", "*idsps*", "*type*", "*barrier*", "*prefactor*", "*ebias*", "*dtot*", "*dmag*", "*dmax*", "*dsum*", "*adsum*", "*nad*", "*dtotfs*", "*dmagfs*", "*dmaxfs*", "*dsumfs*", "*adsumfs*", "*nadfs*", "*dtotfi*", "*dmagfi*", "*dmaxfi*", "*dsumfi*", "*adsumfi*", "*nadfi*", "*isConnect*", and “*reason*”. The keywords are the same with that of KMC_istep_SPs.csv + “*reason*”. The list of “reason” is given below.

Conn: deleted due to connectivity

SE: deleted due to *ScreenEng*

SD: deleted due to *ScreenDisp*

SDSE: deleted due to *ScreenEng* and *ScreenDisp*

SDorSE: deleted due to *ScreenEng* or *ScreenDisp*

BorD: deleted due to the barrier or scalar values of displacement arrays

DupAV: deleted due to duplication of an SP within the same AV

Dup: deleted due to duplication of an SP within different AV

minB_re: deleted due to barrier value being below the minimum barrier in recalibration

minBB_re: deleted due to backward barrier value being below the minimum backward barrier in recalibration

KMC_istep_DetailSPs.csv

Detail information of selected SPs. The keywords are "*isp*", "*barrier*", "*prefactor*", "*ebias*", "*dtot*", "*dmag*", "*dmax*", "*dsum*", "*adsum*", "*dmagx*", "*dmagy*", "*dmagz*", "*dmaxx*", "*dmaxy*", "*dmaxz*", "*vmaxx*", "*vmaxy*", "*vmaxz*", "*dsumx*", "*dsumy*", "*dsumz*", "*adsumx*", "*adsumy*", "*adsumz*", "*dtotfs*", "*dmagfs*", "*dmaxfs*", "*dsumfs*", "*adsumfs*", "*dmagxfs*", "*dmagyfs*", "*dmagzfs*", "*dmaxxfs*", "*dmaxyfs*", "*dmaxzfs*",

"*vmaxxfs*", "*vmaxyfs*", "*vmaxzfs*", "*dsumxfs*", "*dsumyfs*", "*dsumzfs*", "*adsumxfs*", "*adsumyfs*", "*adsumzfs*", "*dtotfi*", "*dmagfi*", "*dmaxfi*", "*dsumfi*", "*adsumfi*", "*dmagxfi*", "*dmagyfi*", "*dmagzfi*", "*dmaxxfi*", "*dmaxyfi*", "*dmaxzfi*", "*vmaxxfi*", "*vmaxyfi*", "*vmaxzfi*", "*dsumxfi*", "*dsumyfi*", "*dsumzfi*", "*adsumxfi*", "*adsumyfi*", and "*adsumzfi*".
KMC_istep_DetailSPs.csv contains the partition values of displacement arrays as introduced in *ValidSPs* in *saddle_point*. Likewise, the appendices of "fs" and "fi" are indicated the displacement arrays between the SP to final state and from the initial to final states, respectively. If *MRM* is used, it is common that the selected event is from one of the basins contained in the superbasin rather than the current basin. In this case, no information is included in the *KMC_istep_DetailSPs.csv* file.

DataOut

Information of structures including the data structure, saddle point structures, and structures of final states at *istep* KMC step. All given in the *LAMMPS* data format.

KMC_istep.dat

The data file at the *istep* KMC step.

KMC_istep_Data_AVs.dat

The active volumes of the current system, in which "*molecular-ID*" of an atom indicates the degree of overlapping, i.e. the number of AVs that include the atom.

KMC_istep_Data_SP_offset+isp.dat

The structure of the *isp* saddle point. The *offset* defaults to 0, if the saddle point is associated with the current system or basin at the *istep* KMC step.
If *MRM* is used, it is common that the selected event is from one of the basins contained in the superbasin rather than the current basin. In this case, the offset will be 10000 if *Sel_iSPs* in *visual* is Auto.

KMC_istep_Data_FI_offset+isp.dat

The structure of final state that associated with the *isp* saddle point. The *offset* defaults to 0, if the saddle point is associated with the current data or basin at the *istep* KMC step.
If *MRM* is used, it is common that the selected event is from one of the basins contained in the superbasin rather than the current basin. In this case, the offset will be 10000 if *Sel_iSPs* in *visual* is Auto.

KMC_istep_Data_Selected_SPs.dat

The structure of stacking saddle points if *OutputStyle* in *visual* is *stack*. All saddle points will be stacked together and appended to the data file. The atom ID of saddle points starts from the maximum atom ID of the system + 1. The output file contains three extra columns "*tag*", "*isp*", and "*idv*", where "*tag*" is the original atom ID, "*isp*" is ID of the saddle point, and "i*dv*" is the id used for the *visibility*.

KMC_istep_Data_Selected_FIs.dat

The structure of stacking final states if *OutputStyle* in *visual* is *stack*. All final states will be stacked together and appended to the data file. The atom ID of the final states starts from the maximum atom ID of the system + 1. The output file contains three extra columns "tag", "isp", and "idv", where "tag" is the original atom ID, "isp" is ID of the saddle point, and "idv" is the id used for the visibility.

DefectBank

The information of defect bank instances. All are *Pandas dataframes* written in *csv* format, in which the first column is the index of the dataframe. The *ibasin* is the ID of the basin or the *defect bank* instance and *idisp* is the ID of the displacement.
The file format is the same with the files in *Preload* in *defect_bank*.

DB_basin_ibasin_sch.csv

The structure of the basin. The "*sch*" is Schoenflies symbol of the detected point group. The keywords are "*type*", "*x*", "*y*", and "*z*". The "*type*" is atom type. The "*x*", "*y*", and "*z*" are the atom coordinates.

DB_b_ibasin_idisp.csv

The displacements of the saddle point that associated with *ibasin* basin. The keywords are "*dx*", "*dy*", and "*dz*". The "*dx*", "*dy*" and "*dz*" are displacements in the *X*, *Y*, and *Z* directions.

AVOut

Information of structures including active volume and saddle point structures at the *istep* KMC step. All files are in the *LAMMPS* data format. All structures are written relative to the active volume, which is a part of the data and centered on the defect. The atom ID of an active volume is reset from 1 to N, where N is the number of atoms in the active volume.

KMC_istep_Data_AV_idav.dat

The structure information of the *idav* AV using the atom ID and coordinates of the system at the *istep* KMC step.

KMC_istep_AV_idav_SPs.dat

The structure of stacking saddle points of *idav* AV using the atom ID and coordinates of the AV at the *istep* KMC step. The atom ID of the final states starts from the maximum atom ID of the AV + 1. The output file contains three extra columns, "*tag*", "*isp*", "*idv*", where "*tag*" is the original atom ID, "*isp*" is ID of the saddle point, and "*idv"* is the id used for the *visibility*, respectively.

KMC_istep_AV_idav_FIs.dat

The structure of stacking final states of the *idav* AV using the atom ID and coordinates of the AV at the *istep* KMC step. The atom ID of final states starts from the maximum atom ID of the AV + 1. The output file contains three extra columns "*tag*", "*isp*", and "*idv*", where "*tag*" is the original atom ID, "*isp*" is ID of the saddle point, and "*idv*" is the id used for the *visibility*, respectively.

KMC_istep_Data_AV_idav_SPs.dat

The structure of stacking saddle points of the *idav* AV using the atom ID and coordinates of the system at the *istep* KMC step. The atom ID of the final states starts from the maximum atom ID of the data + 1. The output file contains three extra columns "*tag*", "*isp*", "*idv*", where "*tag*" is the original atom ID, "*isp*" is ID of the saddle point, and "*idv*" is the id used for the *visibility*, respectively.

KMC_istep_Data_AV_idav_FIs.dat

The structure of stacking final states of the *idav* AV using the atom ID and coordinates of the system at the *istep* KMC step. The atom ID of final states starts from the maximum atom ID of the data + 1. The output file contains three extra columns "*tag*", "*isp*", "*idv*", where "*tag*" is the original atom ID, "*isp*" is ID of the saddle point, and "*idv*" is the id used for the *visibility*, respectively.

ITERATION_RESULTS

If *ShowIterationResults* is True, this folder will be created. Inside this folder, a series of folders named "***idav_idsps***" will be created to store the iteration results of the *idsps* search attempt of the *idav* AV. The iteration is the number of force calls during a SPS attempt. Since a dimer contains two images, every rotation or translation step requires two force calls. If *ShowIterationResults* is True, the *summary_array.csv* will be output. To adjust of the interval of the output files, change the *Inteval4ShowIterationResults*. To output *VN* or reaction coordinates during a SPS attempt, set the corresponding *ShowVN4ShowIterationResults* or *ShowCoords4ShowIterationResults* to True.

summary_array.csv

A summary of the iteration results during a SPS attempt. The keywords are "*iter*", "*ntrans*", "*curvature*", "*ediff*", and "*ediffmax*".

iter: the iteration
ntrans: the translation step
curvature: the curvature of the dimer
ediff: the energy difference with respected to the initial structure
ediff: the maximum of the energy difference with respected to the initial structure

VectorN_TranslationStep_Iteration.csv

The list of the atomic *VN* in “x”, “y”, and “z” directions at the *Iteration* step of the *TranslationStep* translation step.

Coords_TranslationStep_Iteration.csv

The list of the reaction coordinates (coordinates of the dimer) in “x”, “y”, and “z” directions at the *Iteration* step of the *TranslationStep* translation step.

EXAMPLES

A vacancy in Fe bcc structure (*Fe_vacancy*)

It is an example for quick start. Besides the custom parameters for this case (*potential*, *data,* and *active_volume* entries in **input.yaml**), **input.yaml** contains the most commonly used parameters to run SEAKMC_py. With such settings, there are 2 defects (or two AVs) after the point defect reduction. SEAKMC_py found 6 saddle points in first AV and 1 saddle point in second AV (see Seakmc.log). After checking for duplication, it found 7 saddle points at the first KMC step and one saddle point is missed (there are 8 low lying saddle points known to exist in Fe-vacancy system). It is not uncommon that SPS cannot find all the low-lying saddle points within a limited number of search attempts (*NSearch* in *spsearch*). You can increase *NSearch* or turn on *PointGroupSymm* for a symmetric AV to ensure all low-lying saddle points are found.

Due to the randomness in the code, for example the initial *VN* vector, you may not regenerate the exact same results from the same example.

For custom input file for "pylammps" style, see "in.sample.pylammps" at the "examples" directory.

Guided SPS (*GuidedSPS*)

The guided SPS refers to the SPS launch from the loaded saddle points rather than the bottom of the initial well of the PES. There are three places from which to load the saddle points: 1) from the defect bank (see *defect_bank*); 2) from recycled saddle points from previous KMC steps (see *defect_bank*); and 3) from *Preloading* in *spsearch* for the given AV (see *Preloading* in *spsearch*).

The example here involves two steps. The first step is to generate the database of the eight saddle points in the Fe-vacancy system. The second step is performing the SPS with the locations of the saddle points that are generated in the first step.

Generating the database of the saddle points (GenerateSPs)

To generate the database of the saddle points, we need to turn on *Recycle* and *SaveDB* in *defect_bank*. The *SavePath* is set to "GeneratedSPs" in this example. To ensure we can find 8 saddle points, we use a larger cutoff distance (*DCut4PDR*) for the point defect reduction and turn on *PointGroupSymm*. The resulting saddle points are saved inside "GeneratedSPs", where DB_basin_0_Oh.csv contains the coordinates of the AV and DB_b_0_disp_0.csv to DB_b_0_disp_7.csv contains the displacement array of the 8 saddle points.

SPS with the database of the saddle points (GuidedSPS)

The "GeneratedSPs" folder is copied from "GenerateSPs" from the previous step. This example loads the saddle points from *Preloading* in *spsearch*. The *NSearch* is set to 10. The first 8 attempts will load the eight saddle points, one at a time. The last two attempts will perform SPSs without any guidance. You may need to use smaller *TrialStepsize* and *MaxStepsize* for guided SPS, since the starting point of SPS is already close to the saddle point.

SPS on selected defects (*SelectedDefects*)

The data contains a <110> dumbbell interstitial at center and a vacancy at the corner. The *NPredef* is set to 1. The molecule-ID of atoms around the vacancy is set to -2, i.e. they are NOT subjected to finding defects nor to constructing the AV. The *Overlapping* is set to False. The defective atoms around the dumbbell will form a superdefect and the AV is constructed on the constituent reduced defects of the superdefect.

RT_SetMolID is set to True; *DefectCenter4RT_SetMolID* is set to "auto"; and *R4RT_SetMolID* is set to 15.0. These settings will reset the molecule-ID of atoms whose distance to the defect center of the last KMC step is larger than 15.0 Å to -2. *FCT4RT_SetMolID* is set to [0.1, 0.9] for *x*, *y*, and *z* directions. That is to say that if the fractional coordinates of atoms outside these ranges will be set to -2. The input data at every KMC step (KMC_istep.dat) are included in the *DataOut* folder.

To visualize the AV at each KMC step, please open the KMC_istep_Data_AVs.dat files in *DataOut* folder. Please note that the molecule-ID entry in these files indicates the degree of overlapping.

Import outputs from SEAKMC_py (*ImportOutput*)

This example shows how to import the KMC timestep (*time_step*) in SEAKMC_py to a LAMMPS input script and copy the custom output files ("Runner_0/stress_top.dump" and "Runner_0/stress_bot.dump") to *DataOut* folder ("KMC_(istep+1)_stress_top.dump" and "KMC_(istep+1)_stress_bot.dump"). This example simulates the shear test with strain rate of 1 per second, which is several orders of the magnitude slower than the MD simulations. The shearing direction is "*yz*" tilt component on the box. The data is a vacancy in the Fe system with 15 angstroms vacuum on each side of the "*z*" direction. The thickness of the bottom and top fixed layers is 10 angstroms. The stress is computed on atoms that are one atomic layer distance to each of these two fixed layers. The thickness of the slab that is allowed for finding defects is 20 angstroms (molecule-ID is 0) and the molecule-ID of the rest of atoms is -2. The LAMMPS script for relaxing the data between KMC steps is "in.relax". Below summarizes the "in.relax":

a) Import the KMC time_step;
b) Compute the displacement of "*yz*" on the box based on imported KMC timestep with a fixed strain rate of 1e-12 per picosecond (ps);
c) Compute the stress of atoms that are one atomic layer distance to the fixed atoms;
d) Energy minimization of the system;
e) Dump the stress calculations on the top and bottom layers to "Runner_0/stress_top.dump" and "Runner_0/stress_top.dump";
f) Apply the shear strain with "change_box" command in LAMMPS; The commented lines in "in.relax" gives two alternative ways to apply the strain: 1) Deforming the box with the "fix deform" command; and 2) displacing atoms with the "fix move" command;
g) Run the NVT ensamble of the deformed system with the top and bottom atoms fixed for 100 steps;
h) Energy minimization of the system and writing the data for the next KMC step.

Depending on the barriers, the KMC timestep can be very large. For example, the timestep for a single barrier of 0.6 eV with the prefactor of 10 THz at 300.0 K is about 2.6 ms, which makes it possible to match the experimental strain rate (1e-16 to 1e-18 per ps). The strain rate in "in.relax" is set to 1e-12 per ps in order to have the noticeable shear strain in very few steps.

Changes in **input.yaml** are listed below:

a) *RinputOpt* in *data* is set to False; If you have a custom input script to relax the initial data, set it to the filename of the input script;
b) *RinputOpt* in *force_evaluator* is set to "in.relax"; This is the input script for relaxing the system between KMC steps;
c) *ImportValue4RinputOpt* is set to True to import outputs from SEAKMC_py;
d) *Keys4ImportValue4RinputOpt* is set to [Timestep equal, time_step] to import time_step and replace the value after the string of "Timestep equal" in "in.relax";
e) *OutFileHeaders* is set to ["stress"] to copy custom output files in the *Runner_0* folder to the *DataOut* folder;
f) *Temp* in *kinetic_MC* is set to 300.0 K;
g) *Prefactor* is set to 10 THz; You can turn on *CalPrefactor* in *dynamic_matrix* to compute the prefactors.

Use Script File "calllammps" (*Use_calllammps*)

This example shows how to use "calllammps" in SEAKMC_py. If the force_evaluator does NOT have a Python wrapper to update coordinates and get the energies and forces on the fly. The code extracts the needed information from the output files from the force_evaluator. For example, "CONTCAR" and "OUTCAR" for VASP. For LAMMPS, the code will get energy from "log.lammps", get coordinates from "tmp1.dat", and atom forces from "dump.atom.forces". If not custom input file provided, the code will generate LAMMPS input, which contains lines "dump myDump all custom 100000000000000 dump.atom.forces id type x y z fx fy fz" and "dump_modify myDump sort id" to extract atom forces from "dump.atom.forces" and a line "print etotal=$(pe+ke)" to extract the energy from "log.lammps". For custom input files, please include these lines.

Using script file will use PYTHON subprocess module to execute the force_evaluator at the working directory. For custom input files, see "in.sample.lammps" for details.

Changes in **input.yaml** are listed below:

a) *Path2Pot* in 'potential' is set to False. The code will add the working directory as the absolute path to the potential file or you can manually input the absolute path to the potential file.
b) *Style* in 'force_evaluator' set to lammps;
c) *Bin* in 'force_evaluator' set to calllamps;
d) *Path2Bin* in 'force_evaluator' set to False. The code will add the working directory as the absolute path to the call script file or you can manually input the absolute path to the call script file.;

Before running the example, make the script file “calllammps” executable and have permission to run. It is highly recommended to copy the “calllammps” and lammps binary to the system defaulted binary folder.

Since it requires to read the output files and then extract information from the output files at every iteration of saddle point search, it is way slower (about 20 times slower) than the ‘pylammps’ style.

Since VASP takes NULL arguments, the code only passes working directory in ‘callvasp’ script file. The extracting atom forces is not finished in VASP.

REFERENCES

[1] G. Henkelman and H. Jonsson, "A dimer method for finding saddle points on high dimensional potential surfaces using only first derivatives," *The Journal of Chemical Physics,* vol. 111, no. 15, p. 7010, 1999.

[2] A. Heyden, A. T. Bell and F. J. Keil, "Efficient methods for finding transition states in chemical reactions: Comparison of improved dimer method and partitioned rational function optimization method," *The Journal of Chemical Physics,* vol. 123, p. 224101, 2005.

[3] F. El-Mellouhi, N. Mousseau and L. J. Lewis, "Kinetic activation-relaxation technique: An off-lattice self-learning kinetic Monte Carlo algorithm," *Physical Review B,* vol. 78, p. 153202, 2008.

[4] H. Xu, Y. N. Osetsky and R. E. Stoller, "Simulating complex atomistic processes: On-the-fly kinetic Monte Carlo scheme with selective active volumes," *Physical Review B,* vol. 84, p. 132103, 2011.

[5] H. Xu, Y. N. Osetsky and R. E. Stoller, "Self-evolving atomistic kinetic Monte Carlo: fundamentals and applications," *Journal of Physics: Condensed Matter,* vol. 24, p. 375402, 2012.

[6] S. Hayakawa and H. Xu, "Saddle point sampling using scaled normal coordinates," *Computational Materials Science,* vol. 200, p. 110785, 2021.

[7] S. Hayakawa, J. Isaacs, H. R. Medal and H. Xu, "Atomistic modeling of meso-timescale processes with SEAKMC: A perspective and recent developments," *Computational Materials Science,* vol. 194, p. 110390, 2021.

[8] B. Puchala, M. L. Falk and K. Garikipati, "An energy basin finding algorithm for kinetic Monte Carlo acceleration," *The Journal of Chemical Physics,* vol. 132, p. 134104, 2010.

[9] L. K. Beland, P. Brommer, F. El-Mellouhi, J.-F. Joly and N. Mousseau, "Kinetic activation-relaxation technique," *Physical Review E,* vol. 84, p. 046704, 2011.

[10] G. H. Vineyard, "Frequency factors and isotope effects in solid state rate processes," *Journal of Physics and Chemistry of Solids,* vol. 3, p. 121, 1957.

[11] G. Ackland, M. Mendelev, D. Srolovitz, S. Han and A. Barashev, "Development of an interatomic potential for phosphorus impurities in α-iron," *Journal of Physics: Condensed Matter,* vol. 16, no. 27, p. S2629, 2004.

INDEX

A

AbsVal4ScreenD, 63
AccStyle, 23
active_volume, 26, 31
ActiveOnly4SPConfig, 51
AND4ScreenD, 63
AND4ScreenDE, 63
AND4ScreenE, 63
AngCut, 50
AngCut4GSP, 66
AngCut4Type, 66
angle_tolerance, 22
angle4init, 54
AngTol4Init, 54
atom_style, 30
atom_style4Ref, 35
AV Relaxation, 15
AVOut, 76
AVOutPath, 69
AVstep4Restart, 22

B

BackBarrierMin, 60
BarrierCut, 59
BarrierMin, 59
Bin, 27
bondlengths, 26, 32
bondlengths4LAS, 26
boundary, 31
BoxRelax, 29, 30

C

CalBarrsInData, 60
CalEbiasInData, 60
CalPrefactor, 39
CenterVN, 54
Charactering AV, 15
CheckAng, 50
CheckAngSteps, 50
CheckConnectivity, 62
CheckSequence, 53
coordnums, 26, 32
coordnums4LAS, 26
Coords_TranslationStep_Iteration.csv, 78
cutdefectmax, 35
cutneighmax, 26
cutneighs, 26

D

DActive, 38
data, 30
DataOut, 75
DataOutPath, 69
DAtomCut, 60
DB_b_ibasin_disp_idisp.csv, 24
DB_b_ibasin_idisp.csv, 76
DB_basin_ibasin_sch.csv, 24, 76
DBuffer, 38
DCut4Def, 35
DCut4noOverlap, 37
DCut4PDR, 36
DCut4SPAtom, 50
DCut4Vis, 70
DecayRate, 49
DecaySteps, 49
DecayStyle, 49
defect_bank, 24
DefectBank, 76
DefectCenter4RT_SetMolID, 34
Defects, 35
delimiter, 39
DetailOut, 68
DFixed, 38
dimension, 31
DimerSep, 49
displacement, 39
DispStyle, 22
DispStyle4AVSP, 69
DispStyle4DataSP, 68
DMagCut, 60
DMagCut_FI, 61
DMagCut_FS, 61
DMagMin, 60
DmagMin_FI, 61
DmagMin_FS, 61
DmaxCut, 60
DmaxCut_FI, 61
DmaxCut_FS, 62
DmaxMin, 60
DmaxMin_FI, 61
DmaxMin_FS, 62
DRatio4Relax, 50
DsumCut, 60

DsumCut_FI, 61
DsumCut_FS, 62
DsumMin, 61
DsumMin_FI, 61
DsumMin_FS, 62
DsumrCut, 61
DsumrCut_FI, 61
DsumrCut_FS, 62
DsumrMin, 61
DsumrMin_FI, 61
DsumrMin_FS, 62
DtotCut, 60
DtotCut_FI, 61
DtotCut_FS, 61
DtotMin, 60
DtotMin_FI, 61
DtotMin_FS, 61
dynamic_matrix, 39
DynCut4SPAtom, 50

E

EbiasCut, 59
EbiasMin, 59
EnConv, 50
EnCut4GSP, 66
EnCut4Transient, 23
EnCut4Type, 67
Energy Recalibration, 15
EnTol4AVSP, 66
examples, 20

F

FConv, 50
FCT4RT_SetMolID, 34
FileHeader, 24, 53
FileHeader4Data, 53
FileName, 25, 30, 39
FindDefects, 25, 26, 35
Finding Defects, 15
FindSPType, 66
FixAxesStr, 57
FixTypes, 57
float_precision, 22
FMin4Rot, 50
force_evaluator, 16, 17, 21, 27, 56
FThres4Rot, 50

G

GroupSP, 66

H

Handle_no_Backward, 23
HandleVN, 54

I

IgnoreSteps, 50, 54
IgnoreType, 25
ImportValue4RinputOpt, 28
INDEX, 84
InitTemp4Opt, 29, 52
insituGuidedSPS, 52
Int4ComputeScale, 55
Inteval4ShowIterationResults, 51
Inteval4ShowProgress, 21
Invisible, 70
ITERATION_RESULTS, 77

K

Keys4ImportValue4RinputOpt, 28
kim_init, 26
kim_interaction, 26
kim_param, 26
kinetic_MC, 22
KMC (LEB), 15
KMC_istep_AV_idav_FIs.dat, 77
KMC_istep_AV_idav_SPs.dat, 69, 76
KMC_istep_Data_AV_idav_FIs.dat, 77
KMC_istep_Data_AV_idav_SPs.dat, 69, 77
KMC_istep_Data_AV_idav.dat, 69, 76
KMC_istep_Data_AVs.dat, 68, 75
KMC_istep_Data_FI_(offset+isp).dat, 69
KMC_istep_Data_FI_offset+isp.dat, 75
KMC_istep_Data_Selected_FIs.dat, 69, 76
KMC_istep_Data_Selected_SPs.dat, 69, 75
KMC_istep_Data_SP_(offset+isp).dat, 69
KMC_istep_Data_SP_offset+isp.dat, 75
KMC_istep_Deleted_SPs.csv, 74
KMC_istep_DetailSPs.csv, 68, 74
KMC_istep_Prob.csv, 68, 74
KMC_istep_SPs.csv, 73
KMC_istep.dat, 68, 75
KMCstep4Restart, 22

L

LenCut4Type, 67
LoadDB, 24
LoadFile, 22
Loading Restart/DB, 15
LoadPath, 24, 52
LoadRestart, 22
LocalRelax, 23, 52
Log, 68
LogFile, 28
LowerHalfMat, 39

M

MagCut4GSP, 66
MagCut4Type, 67
masses, 25
Master_Slave, 58
MaxIter4Init, 54
MaxRatio4Barr, 66
MaxRatio4Dmag, 66
MaxStepsize, 49
MaxVal4ScreenD, 63
MaxVal4ScreenE, 63
MD/Relaxation, 15
Method, 35, 48, 52
Method4Prefactor, 40
MinSpan4LOGV, 55
MinSpan4RAS, 56
MinStepsize, 49
MinVal4ScreenD, 63
MinVal4ScreenE, 63
MinValue4LOGV, 55
MoleDyn, 30

N

NActive, 34
NBuffer, 34
NCommonMin, 67
NFixed, 34
NMax4AV, 38
NMax4DB, 24
NMax4Def, 37
NMax4Dup, 67
NMax4PG, 39
NMax4Rot, 50
NMax4SNC, 39
NMax4Trans, 49
NMaxBasin, 23
NMaxRandVN, 54
NMin4AV, 38
NMin4DB, 25
NPredef, 34
nproc, 27, 56
Nproc4Recal, 28
NScreenDisp, 62
NScreenEng, 62
NSearch, 48
NSteps, 22
NSteps4CenterVN, 54
NSteps4Relax, 28, 30
NVTSteps4Opt, 29, 52

O

Offset, 69
OpenKIM, 26
Order4Recursive4AV, 37
Order4Recursive4PDR, 37
OutFileHeaders, 29
OUTPUT, 68
OutputStyle, 68
Overlap4OrderRecursive, 37
Overlapping, 37

P

pair_coeff, 25
pair_style, 25
partition, 28, 56
Path2Bin, 27
Path2Pot, 25
PDReduction, 36
Period4MA, 55
PointGroupSymm, 39
Post-processing SPs, 15
potential, 25, 32
PowerOnV, 55
PredefOnly, 34
prefactor, 60
Preload, 24, 52
Preloading, 20, 52
Preloading SPs, 15
processors, 28, 56
pymatgen, 25

R

R2Dmax4SPAtom, 50
R2Dmax4Tol, 67
R4RT_SetMolID, 34
Ratio4DispLoad, 24, 53
Ratio4Zero4LOGV, 55
Ratio4Zero4RAS, 56
RatioStepsize, 49
RatioVN04Preload, 54
RCut4Vis, 70
RealtimeDelete, 62
RealtimeValid, 62
RecursiveRed, 36
Recycle, 24
Recycling, 15
ReferenceData, 35
Relaxation, 29
Relaxed, 30
RescaleStyle4LOGV, 55
RescaleStyle4RAS, 55
RescaleValue, 55
RescaleVN, 55
Reset_Index, 70
Reset_Simulation_Time, 21
ResetVN04Preload, 54
Restart, 21

RESTART_istep_idav.restart, 73
Rinput, 56
RinputDM, 56
RinputMD, 30
RinputMD0, 28, 30
RinputOpt, 28, 30, 56
RT_SetMolID, 34

S

saddle_point, 58
SaveDB, 24
SavePath, 24
Scaling, 25, 53
Screen, 28, 68
ScreenDisp, 63
ScreenEng, 63
Seakmc_summary.csv, 72
Seakmc.log, 71
SearchBuffer, 49
Sel_iSPs, 68
ShowBuffer, 70
ShowDimer4ShowIterationResults, 51
ShowFixed, 70
ShowIterationResults, 51
ShowVN4ShowIterationResults, 51
significant_figures, 22
SNC, 15, 39
Sort_by, 38
SortD4PDR, 36
SortDisps, 24, 53
Sorting, 23, 38
SortingBuffer, 38
SortingFixed, 33, 38
SortingShift, 38
SortingSpacer, 38
SP Search, 15
species, 25
SPOut, 73
SPOutPath, 69
SPs4Detail, 68
spsearch, 17, 20, 23, 24, 27, 40, 58
Stack4noOverlap, 37
Str4ScreenD, 63
Style, 27, 33
summary_array.csv, 77
system, 21
System inputs, 15
System Relaxation, 15

T

TakeMin4MixedRescales, 55
TargetTemp4NVT, 29, 52
TaskDist, 58
Temp, 22
Temp4Time, 22
TempFiles, 21
Thres4Recalib, 60
timestep, 28
Tol4AVSP, 66
Tol4Barr, 23
Tol4Connect, 50
Tol4Disp, 23, 25, 67
Tolerance, 22
toScreenDisp, 63
toScreenEng, 63
TrialStepsize, 49
TurnoffPBC, 39
Type4ScreenD, 63
Type4ScreenE, 63

U

units, 31
UseSymm, 24

V

Validating SPs, 15
ValidSPs, 62
VectorN_TranslationStep_Iteration.csv, 78
VerySmallNumber, 22
VibCut, 40
visual, 68

W

Write_AV_SPs, 69
Write_Data_AV_SPs, 69
Write_Data_AVs, 68
Write_Data_SPs, 68
Write_KMC_Data, 68
Write_Local_AV, 69
Write_Prob, 68
Write_SP_Summary, 68
WriteRestart, 22

X

XRange4LOGV, 55
XRange4RAS, 56