GPUMD documentation

The GPUMD developer team

Oct 31, 2025

MAIN

GPUMD stands for *Graphics Processing Units Molecular Dynamics*. It is a general-purpose molecular dynamics (*MD*) package fully implemented on graphics processing units (*GPU*). In addition to *several empirical interatomic potentials*, it supports *neuroevolution potential* (*NEP*) models. **GPUMD** also allows one to construct the latter type of models using the *nep executable*.

CHAPTER

ONE

INTRODUCTION

GPUMD stands for *Graphics Processing Units Molecular Dynamics*. It is a general-purpose molecular dynamics (*MD*) package fully implemented on graphics processing units (*GPU*). In addition to *several empirical interatomic potentials*, it supports *neuroevolution potential* (*NEP*) models. **GPUMD** also allows one to construct the latter type of models using the *nep executable*.

GPUMD is also highly efficient for conducting *MD* simulations with many-body potentials such as the Tersoff potential and is particularly good for heat transport applications. It is written in CUDA C++ and requires a CUDA-enabled Nvidia GPU of compute capability no less than 3.5.

1.1 Python interfaces

There are several packages that provide Python interfaces to **GPUMD** functionality, including calorine, gpyumd, and pyNEP.

1.2 GitHub discussions

- For general discussions: https://github.com/brucefan1983/GPUMD/discussions
- For issue reporting: https://github.com/brucefan1983/GPUMD/issues

1.3 Discussion groups

There is a Chinese discussion group based on the following QQ number: 778083814

CHAPTER

TWO

INSTALLATION

2.1 Download

The source code is hosted on github.

2.2 Prerequisites

To compile (and run) **GPUMD** one requires an Nvidia GPU card with compute capability no less than 3.5 and CUDA toolkit 9.0 or newer. On Linux systems, one also needs a C++ compiler supporting at least the C++11 standard. On Windows systems, one also needs the `cl.exe` compiler from Microsoft Visual Studio and a 64-bit version of make.exe.

2.3 Compilation

In the `src` directory run `make`, which generates two executables, `nep` and `gpumd`. Please check the comments in the beginning of the makefile for some compiling options.

2.4 Examples

You can find several examples for how to use both the `gpumd` and `nep` executables in the examples directory of the **GPUMD** repository.

2.5 NetCDF setup

To use NetCDF (see *dump_netcdf keyword*) with **GPUMD**, a few extra steps must be taken before building **GPUMD**. First, you must download and install the correct version of NetCDF. Currently, **GPUMD** is coded to work with netCDF-C 4.6.3 and it is recommended that this version is used (not newer versions).

The setup instructions are below:

- Download netCDF-C 4.6.3
- Configure and build NetCDF. It is best to follow the instructions included with the software but, for the configuration, please use the following flags seen in our example line

```
./configure --prefix=<path> --disable-netcdf-4 --disable-dap
```

Here, the `--prefix` determines the output directory of the build. Then make and install NetCDF:

```
make -j && make install
```

- Enable the NetCDF functionality. To do this, one must enable the `USE_NETCDF` flag. In the makefile, this will look as follows:

```
CFLAGS = -std=c++14 -O3 $(CUDA_ARCH) -DUSE_NETCDF
```

  In addition to that line the makefile must also be updated to the following:

```
INC = -I<path>/include -I./
LDFLAGS = -L<path>/lib
LIBS = -lcublas -lcusolver -l:libnetcdf.a
```

  where `<path>` should be replaced with the installation path for NetCDF (defined in `--prefix` of the `./configure` command).

- Follow the remaining **GPUMD** installation instructions

Following these steps will enable the *dump_netcdf keyword*.

2.6 PLUMED setup

To use PLUMED (see *plumed keyword*) with **GPUMD**, a few extra steps must be taken before building **GPUMD**. First, you must download and install PLUMED.

The setup instructions are below:

- Download the latest version of PLUMED, e.g. the plumed-src-2.8.2.tgz tarball.
- Configure and build PLUMED. It is best to follow the instructions, but for a quick installation, you may use the following setup:

```
./configure --prefix=<path> --disable-mpi --enable-openmp --enable-modules=all
```

  Here, the `--prefix` determines the output directory of the build. Then make and install PLUMED:

```
make -j6 && make install
```

  Then update your environment variables (e.g., add the following lines to your bashrc file):

```
export PLUMED_KERNEL=<path>/lib/libplumedKernel.so
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path>/lib
export PATH=$PATH:<path>/bin
```

  where `<path>` should be replaced with the installation path for PLUMED (defined in `--prefix` of the `./configure` command). Finally, reopen your shell to apply changes.

- Enable the PLUMED functionality. To do this, one must enable the `USE_PLUMED` flag. In the makefile, this will look as follows:

```
CFLAGS = -std=c++14 -O3 $(CUDA_ARCH) -DUSE_PLUMED
```

  In addition to that line the makefile must also be updated to the following:

```
INC = -I<path>/include -I./
LDFLAGS = -L<path>/lib -lplumed -lplumedKernel
```

  where `<path>` should be replaced with the installation path for PLUMED (defined in `--prefix` of the `./configure` command).

- Follow the remaining **GPUMD** installation instructions

Following these steps will enable the *plumed keyword*.

2.7 DP potential support

2.7.1 Program introduction

This is the beginning of **GPUMD** support for other machine learned interatomic potentials.

Necessary instructions

- This is a test version.
- Only potential function files ending with `.pb` in deepmd are supported, that is, the potential function files of the tensorflow version generated using `dp --tf` freeze.

Installation dependencies

- You must ensure that the new version of DP is installed and can run normally. This program contains DP-related dependencies.
- The installation environment requirements of GPUMD itself must be met.

2.7.2 Installation details

Use the instance in AutoDL for testing. If one need testing use AutoDL, please contact Ke Xu (twtdq@qq.com).

And we have created an image in AutoDL that can run GPUMD-DP directly, which can be shared with the account that provides the user ID. Then, you will not require the following process and can be used directly.

2.7.3 GPUMD-DP installation (Offline version)

DP installation (Offline version)

Use the latest version of DP installation steps:

```
>> $ # Copy data and unzip files.
>> $ cd /root/autodl-tmp/
>> $ wget https://mirror.nju.edu.cn/github-release/deepmodeling/deepmd-kit/v3.0.0/deepmd-
↪kit-3.0.0-cuda126-Linux-x86_64.sh.0 -O deepmd-kit-3.0.0-cuda126-Linux-x86_64.sh.0
>> $ wget https://mirror.nju.edu.cn/github-release/deepmodeling/deepmd-kit/v3.0.0/deepmd-
↪kit-3.0.0-cuda126-Linux-x86_64.sh.1 -O deepmd-kit-3.0.0-cuda126-Linux-x86_64.sh.1
>> $ cat deepmd-kit-3.0.0-cuda126-Linux-x86_64.sh.0 deepmd-kit-3.0.0-cuda126-Linux-x86_
↪64.sh.1 > deepmd-kit-3.0.0-cuda126-Linux-x86_64.sh
>> $ # rm deepmd-kit-3.0.0-cuda126-Linux-x86_64.sh.0 deepmd-kit-3.0.0-cuda126-Linux-x86_
↪64.sh.1 # Please use with caution "rm"
>> $ sh deepmd-kit-3.0.0-cuda126-Linux-x86_64.sh -p /root/autodl-tmp/deepmd-kit -u #␣
↪Just keep pressing Enter/yes.
>> $ source /root/autodl-tmp/deepmd-kit/bin/activate /root/autodl-tmp/deepmd-kit
>> $ dp -h
```

After running according to the above steps, using `dp -h` can successfully display no errors.

GPUMD-DP installation

The GitHub link is Here.

```
>> $ wget https://codeload.github.com/Kick-H/GPUMD/zip/
↪7af5267f4d8ba720830c154f11634a1942b66b08
>> $ cd ${GPUMD}/src-v0.1
```

Modify `makefile` as follows:

- Line 19 is changed from `CUDA_ARCH=-arch=sm_60` to `CUDA_ARCH=-arch=sm_89` (for RTX 4090). Modify according to the corresponding graphics card model.
- Line 25 is changed from `INC = -I./` to `INC = -I./ -I/root/miniconda3/deepmd-kit/source/build/path_to_install/include/deepmd`
- Line 27 is changed from `LIBS = -lcublas -lcusolver` to `LIBS = -lcublas -lcusolver -L/root/miniconda3/deepmd-kit/source/build/path_to_install/lib -ldeepmd_cc`

Then run the following installation command:

```
>> $ sudo echo "export LD_LIBRARY_PATH=/root/miniconda3/deepmd-kit/source/build/path_to_
↪install/lib:$LD_LIBRARY_PATH" >> /root/.bashrc
>> $ source /root/.bashrc
>> $ make gpumd -j
```

Running tests

```
>> $ cd /root/miniconda3/GPUMD-bu0/tests/dp
>> $ ../../src/gpumd
```

2.7.4 GPUMD-DP installation (Online version)

Introduction

This is to use the online method to install the *GPUMD-DP* version, you need to connect the machine to the Internet and use github and other websites.

Conda environment

Create a new conda environment with Python and activate it:

```
>> $ conda create -n tf-gpu2  python=3.9
>> $ conda install -c conda-forge cudatoolkit=11.8
>> $ pip install --upgrade tensorflow
```

Download deep-kit and install

Download DP source code and compile the source files following DP docs. Here are the cmake commands:

```
>> $ git clone https://github.com/deepmodeling/deepmd-kit.git
>> $ cd deepmd-kit/source
>> $ mkdir build && cd build
>> $ cmake -DENABLE_TENSORFLOW=TRUE -DUSE_CUDA_TOOLKIT=TRUE -DCMAKE_INSTALL_PREFIX=`path_
↪to_install` -DUSE_TF_PYTHON_LIBS=TRUE ../
>> $ make -j && make install
```

We just need the DP C++ interface, so we don't source all DP environment. The libraries will be installed in `path_to_install`.

Configure the GPUMD makefile

The GitHub link is Here.

```
>> $ wget https://codeload.github.com/Kick-H/GPUMD/zip/
↪7af5267f4d8ba720830c154f11634a1942b66b08
>> $ cd ${GPUMD}/src
>> $ vi makefile
```

Configure the makefile of GPUMD. The DP code is included by macro definition `USE_TENSORFLOW`. So add it to `CFLAGS`:

```
CFLAGS = -std=c++14 -O3 $(CUDA_ARCH) -DUSE_TENSORFLOW
```

Then link the DP C++ libraries. Add the following two lines to update the include and link paths and compile GPUMD:

```
INC += -Ipath_to_install/include/deepmd
```

```
LDFLAGS += -Lpath_to_install/lib -ldeepmd_cc
```

Then, you can install it using the following command:

```
>> $ make gpumd -j
```

Run GPUMD

When running GPUMD, if an error occurs stating that the DP libraries could not be found, add the library path temporarily with:

```
LD_LIBRARY_PATH=path_to_install/lib:$LD_LIBRARY_PATH
```

Or add the environment permanently to the `~/.bashrc`:

```
>> $ sudo echo "export LD_LIBRARY_PATH=/root/miniconda3/deepmd-kit/source/build/path_to_
↪install/lib:$LD_LIBRARY_PATH" >> ~/.bashrc
>> $ source ~/.bashrc
```

Run Test

This DP interface requires two files: a setting file and a DP potential file. The first file is very simple and is used to inform GPUMD of the atom number and types. For example, the `dp.txt` is shown in here for use the `potential dp.txt DP_POTENTIAL_FILE.pb` command in the `run.in` file:

```
dp 2 O H
```

Notice

The type list in the setting file and the potential file must be the same.

Example

- Some water simulations using the `DP` model in `GPUMD`: https://github.com/brucefan1983/GPUMD/discussions

References

- DeePMD-kit: https://github.com/deepmodeling/deepmd-kit

CHAPTER

THREE

THEORETICAL BACKGROUND

3.1 Forces and stresses

In classical molecular dynamics, the total potential energy U of a system can be written as the sum of site potentials U_i:

$$U = \sum_{i=1}^{N} U_i(\{\boldsymbol{r}_{ij}\}_{j \neq i}).$$

Here, $\boldsymbol{r}_{ij} \equiv \boldsymbol{r}_j - \boldsymbol{r}_i$ is the position difference vector used throughout this manual.

3.1.1 Interatomic forces

A well-defined force expression for general many-body potentials that explicitly respects (the weak version of) Newton's third law has been derived in [Fan2015]:

$$\boldsymbol{F}_i = \sum_{j \neq i} \boldsymbol{F}_{ij}$$

where

$$\boldsymbol{F}_{ij} = -\boldsymbol{F}_{ji} = \frac{\partial U_i}{\partial \boldsymbol{r}_{ij}} - \frac{\partial U_j}{\partial \boldsymbol{r}_{ji}} = \frac{\partial \left(U_i + U_j\right)}{\partial \boldsymbol{r}_{ij}}.$$

Here, $\partial U_i / \partial \boldsymbol{r}_{ij}$ is a shorthand notation for a vector with Cartesian components $\partial U_i / \partial x_{ij}$, $\partial U_i / \partial y_{ij}$, and $\partial U_i / \partial z_{ij}$.

Starting from the above force expression, one can derive expressions for the stress tensor and the heat current.

3.1.2 Stress tensor

The stress tensor is an important quantity in MD simulations. It consists of two parts: a virial part which is related to the force and an ideal gas part which is related to the temperature. The virial part must be calculated along with force evaluation.

The validity of Newton's third law is crucial to derive the following expression of the virial tensor [Fan2015]:

$$\mathbf{W} = \sum_i \mathbf{W}_i$$

where

$$\mathbf{W}_i = -\frac{1}{2} \sum_{j \neq i} \boldsymbol{r}_{ij} \otimes \boldsymbol{F}_{ij},$$

where only relative positions $\boldsymbol{r}_{ij}$ are involved.

After some algebra, we can also express the virial as [Gabourie2021]

$$\mathbf{W} = \sum_i \mathbf{W}_i$$

where

$$\mathbf{W}_i = \sum_{j \neq i} \boldsymbol{r}_{ij} \otimes \frac{\partial U_j}{\partial \boldsymbol{r}_{ji}}.$$

The ideal gas part of the stress is:

$$\frac{\sum_i m_i v_i^\alpha v_i^\beta}{V},$$

where m_i is the mass of particle \$i\$, v_i^α is the velocity of particle \$i\$, and V is the volume of the system.

The total stress tensor is thus

$$\sigma^{\alpha\beta} = \frac{W^{\alpha\beta} + \sum_i m_i v_i^\alpha v_i^\beta}{V}.$$

3.2 Ensembles

The aim of the time evolution in *MD* simulations is to find the phase trajectory

$$\{\boldsymbol{r}_i(t_1),\ \boldsymbol{v}_i(t_1)\}_{i=1}^N,\ \{\boldsymbol{r}_i(t_2),\ \boldsymbol{v}_i(t_2)\}_{i=1}^N,\ \cdots$$

starting from the initial phase point

$$\{\boldsymbol{r}_i(t_0),\ \boldsymbol{v}_i(t_0)\}_{i=1}^N.$$

The time interval between two time points $\Delta t = t_1 - t_0 = t_2 - t_1 = \cdots$ is called the time step.

The algorithm for integrating by one step depends on the ensemble type and other external conditions. There are many ensembles used in MD simulations, but we only consider the following 3 in the current version:

3.2.1 NVE ensemble

The NVE ensmeble is also called the micro-canonical ensemble. We use the velocity-Verlet integration method with the following equations:

$$\boldsymbol{v}_i(t_{m+1}) \approx \boldsymbol{v}_i(t_m) + \frac{\boldsymbol{F}_i(t_m) + \boldsymbol{F}_i(t_{m+1})}{2m_i}\Delta t$$
$$\boldsymbol{r}_i(t_{m+1}) \approx \boldsymbol{r}_i(t_m) + \boldsymbol{v}_i(t_m)\Delta t + \frac{1}{2}\frac{\boldsymbol{F}_i(t_m)}{m_i}(\Delta t)^2.$$

Here, $\boldsymbol{v}_i(t_m)$ is the velocity vector of particle i at time t_m. $\boldsymbol{r}_i(t_m)$ is the position vector of particle i at time t_m. $\boldsymbol{F}_i(t_m)$ is the force vector of particle i at time t_m. m_i is the mass of particle i and Δt is the time step for integration.

3.2.2 NVT ensemble

The NVT ensemble is also called the canonical ensemble. We have implemented several thermostats for the NVT ensemble.

Berendsen thermostat

The velocities are scaled in the Berendsen thermostat [Berendsen1984] as follows:

$$\boldsymbol{v}_i^{\text{scaled}} = \boldsymbol{v}_i \sqrt{1 + \frac{\Delta t}{\tau_T}\left(\frac{T_0}{T} - 1\right)}.$$

Here, τ_T is the coupling time (relaxation time) parameter, T_0 is the target temperature, and T is the instant temperature calculated from the current velocities. We require that $\tau_T/\Delta t \geq 1$.

When $\tau_T/\Delta t = 1$, the above formula reduces to the simple velocity-scaling formula:

$$\boldsymbol{v}_i^{\text{scaled}} = \boldsymbol{v}_i \sqrt{\frac{T_0}{T}}.$$

A larger value of $\tau_T/\Delta t$ represents a weaker coupling between the system and the thermostat. We recommend a value of $\tau_T/\Delta t \approx 100$. The above re-scaling is applied at each time step after the velocity-Verlet integration. This thermostat is usually used for reaching equilibrium and is *not recommended* for sampling the canonical ensemble.

Nose-Hoover chain thermostat

In the Nose-Hoover chain (*NHC*) method [Tuckerman2010], the equations of motion for the particles in the thermostatted region are (those for the thermostat variables are not presented):

$$\frac{d\boldsymbol{r}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i}$$
$$\frac{d\boldsymbol{p}_i}{dt} = \boldsymbol{F}_i - \frac{\pi_0}{Q_0}\boldsymbol{p}_i.$$

Here, $\boldsymbol{r}_i$ is the position of atom i. $\boldsymbol{p}_i$ is the momentum of atom i. m_i is the mass of atom i. $\boldsymbol{F}_i$ is the total force on atom i resulting from the potential model used. $Q_0 = N_{\rm f} k_{\rm B} T_0 \tau_T^2$ is the ''mass" of the thermostat variable directly coupled to the system and π_0 is the corresponding ''momentum". $N_{\rm f}$ is the degree of freedom in the thermostatted region. $k_{\rm B}$ is Boltzmann's constant and T_0 is the target temperature. τ_T is a time parameter, and we suggest a value of $\tau_T/\Delta t \approx 100$, where Δt is the time step.

We use a fixed chain length of 4.

Langevin thermostat

In the Langevin method, the equations of motion for the particles in the thermostatted region are

$$\frac{d\boldsymbol{r}_i}{dt} = \frac{\boldsymbol{p}_i}{m_i}$$
$$\frac{d\boldsymbol{p}_i}{dt} = \boldsymbol{F}_i - \frac{\boldsymbol{p}_i}{\tau_T} + \boldsymbol{f}_i,$$

Here, $\boldsymbol{r}_i$ is the position of atom i. $\boldsymbol{p}_i$ is the momentum of atom i. m_i is the mass of atom i. $\boldsymbol{F}_i$ is the total force on atom i resulted from the potential model used. $\boldsymbol{f}_i$ is a random force with a variation determined by the fluctuation-dissipation relation to recover the canonical ensemble distribution with the target temperature. τ_T is a time parameter, and we suggest a value of $\tau_T/\Delta t \approx 100$, where Δt is the time step. We implemented the integrators proposed in [Bussi2007a] and [Leimkuhler2013].

Bussi-Donadio-Parrinello thermostat

The Berendsen thermostat does not generate a true NVT ensemble. As an extension of the Berendsen thermostat, the Bussi-Donadio-Parrinello (*BDP*) thermostat [Bussi2007b] incorporates a proper randomness into the velocity re-scaling factor and generates a true NVT ensemble. It is also called the stochastic velocity rescaling (*SVR*) thermostat.

In the *BDP* thermostat, the velocities are scaled in the following way:

$$\boldsymbol{v}_i^{\text{scaled}} = \alpha \boldsymbol{v}_i$$

where

$$\alpha^2 = e^{-\Delta t/\tau_T} + \frac{T_0}{TN_f}\left(1 - e^{-\Delta t/\tau_T}\right)\left(R_1^2 + \sum_{i=2}^{N_f} R_i^2\right) + 2e^{-\Delta t/2\tau_T} R_1 \sqrt{\frac{T_0}{TN_f}\left(1 - e^{-\Delta t/\tau_T}\right)}.$$

Here, $\boldsymbol{v}_i$ is the velocity of atom i before the re-scaling. N_f is the degree of freedom in the thermostatted region. T is instant temperature and T_0 is the target temperature. Δt is the time step for integration. τ_T is a time parameter, and we suggest a value of $\tau_T/\Delta t \approx 100$, where Δt is the time step. $\{R_i\}_{i=1}^{N_f}$ are N_f Gaussian distributed random numbers with zero mean and unit variance.

3.2.3 NPT ensemble

The NPT ensemble is also called the isothermal-isobaric ensemble.

Berendsen barostat

The Berendsen barostat [Berendsen1984] is used with the Berendsen thermostat discussed above. The barostat scales the box and positions as follows:

$$\begin{pmatrix} a_x^{\text{scaled}} & b_x^{\text{scaled}} & c_x^{\text{scaled}} \\ a_y^{\text{scaled}} & b_y^{\text{scaled}} & c_y^{\text{scaled}} \\ a_z^{\text{scaled}} & b_z^{\text{scaled}} & c_z^{\text{scaled}} \end{pmatrix} = \begin{pmatrix} \mu_{xx} & \mu_{xy} & \mu_{xz} \\ \mu_{yx} & \mu_{yy} & \mu_{yz} \\ \mu_{zx} & \mu_{zy} & \mu_{zz} \end{pmatrix} \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{pmatrix}$$

and

$$\begin{pmatrix} x_i^{\text{scaled}} \\ y_i^{\text{scaled}} \\ z_i^{\text{scaled}} \end{pmatrix} = \begin{pmatrix} \mu_{xx} & \mu_{xy} & \mu_{xz} \\ \mu_{yx} & \mu_{yy} & \mu_{yz} \\ \mu_{zx} & \mu_{zy} & \mu_{zz} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}.$$

We consider the following three pressure-controlling conditions:

- *Condition 1*: The simulation box is *orthogonal* and only the hydrostatic pressure (trace of the pressure tensor) is controlled. The simulation box must be periodic in all three directions. The scaling matrix only has nonzero diagonal components and the diagonal components can be written as:

$$\mu_{xx} = \mu_{yy} = \mu_{zz} = 1 - \frac{\beta_{\text{hydro}}\Delta t}{3\tau_p}(p_{\text{hydro}}^{\text{target}} - p_{\text{hydro}}^{\text{instant}}).$$

- *Condition 2*: The simulation box is *orthogonal* and the three diagonal pressure components are controlled independently. The simulation box can be periodic or non-periodic in any of the three directions. Pressure is only controlled for periodic directions. The diagonal components of the scaling matrix can be written as:

$$\mu_{xx} = 1 - \frac{\beta_{xx}\Delta t}{3\tau_p}(p_{xx}^{\text{target}} - p_{xx}^{\text{instant}})$$

$$\mu_{yy} = 1 - \frac{\beta_{yy}\Delta t}{3\tau_p}(p_{yy}^{\text{target}} - p_{yy}^{\text{instant}})$$

$$\mu_{zz} = 1 - \frac{\beta_{zz}\Delta t}{3\tau_p}(p_{zz}^{\text{target}} - p_{zz}^{\text{instant}}).$$

- *Condition 3*: The simulation box is *triclinic* and the 6 nonequivalent pressure components are controlled independently. The simulation box must be periodic in all three directions. The scaling matrix components are:

$$\mu_{\alpha\beta} = 1 - \frac{\beta_{\alpha\beta}\Delta t}{3\tau_p}(p_{\alpha\beta}^{\text{target}} - p_{\alpha\beta}^{\text{instant}}).$$

The parameter $\beta_{\alpha\beta}$ is the isothermal compressibility, which is the inverse of the elastic modulus. Δt is the time step and τ_p is the pressure coupling time (relaxation time).

Stochastic cell rescaling barostat

The Berendsen method does not generate a true NPT ensemble. As an extension of the Berendsen method, the stochastic cell rescaling (*SCR*) barostat [Bernetti2020], combined with the *BDP* thermostat, incorporates a proper randomness into the box and position rescaling factor and generates a true NPT ensemble.

In the *SCR* barostat, the scaling matrix is a sum of the scaling matrix as in the Berendsen barostat and a stochastic one. The stochastic scaling matrix components are

$$\mu_{\alpha\beta}^{\mathrm{stochastic}} = \sqrt{\frac{1}{D_{\mathrm{couple}}}} \sqrt{\frac{\beta_{\alpha\beta}\Delta t}{3\tau_p} \frac{2k_{\mathrm{B}}T^{\mathrm{target}}}{V}} R_{\alpha\beta}.$$

Here, $\beta_{\alpha\beta}$, Δt, and τ_p have the same meanings as in the Berendsen barostat. k_{B} is Boltzmann's constant. T^{target} is the target temperature. V is the current volume of the system. $R_{\alpha\beta}$ is a Gaussian random number with zero mean and unit variance. D_{couple} is the number of directions that are coupled together. It is 3, 1, and 1, respectively, for *condition 1*, *condition 2*, and *condition 3* as discussed above.

3.3 Heat current

Using the force expression, one can derive the following expression for the heat current for the whole system (E_i is the total energy of atom i) [Fan2015]:

$$\boldsymbol{J} = \boldsymbol{J}^{\mathrm{pot}} + \boldsymbol{J}^{\mathrm{kin}} = \sum_i \boldsymbol{J}_i^{\mathrm{pot}} + \sum_i \boldsymbol{J}_i^{\mathrm{kin}};$$

where

$$\boldsymbol{J}_i^{\mathrm{kin}} = \boldsymbol{v}_i E_i;$$

and

$$\boldsymbol{J}_i^{\mathrm{pot}} = -\frac{1}{2} \sum_{j \neq i} \boldsymbol{r}_{ij} \left(\frac{\partial U_i}{\partial \boldsymbol{r}_{ij}} \cdot \boldsymbol{v}_j - \frac{\partial U_j}{\partial \boldsymbol{r}_{ji}} \cdot \boldsymbol{v}_i \right).$$

The potential part of the per-atom heat current can also be written in the following equivalent forms [Fan2015]:

$$\boldsymbol{J}_i^{\mathrm{pot}} = -\sum_{j \neq i} \boldsymbol{r}_{ij} \left(\frac{\partial U_i}{\partial \boldsymbol{r}_{ij}} \cdot \boldsymbol{v}_j \right);$$

and

$$\boldsymbol{J}_i^{\mathrm{pot}} = \sum_{j \neq i} \boldsymbol{r}_{ij} \left(\frac{\partial U_j}{\partial \boldsymbol{r}_{ji}} \cdot \boldsymbol{v}_i \right).$$

Therefore, the per-atom heat current can also be expressed in terms of the per-atom virial [Gabourie2021]:

$$\boldsymbol{J}_i^{\mathrm{pot}} = \mathbf{W}_i \cdot \boldsymbol{v}_i.$$

where the per-atom virial tensor cannot be assumed to be symmetric and the full tensor with 9 components should be used [Gabourie2021]. This result has actually been clear from the derivations in [Fan2015] but it was wrongly stated there that the potential part of the heat current *cannot* be expressed in terms of the per-atom virial.

One can also derive the following expression for the heat current from a subsystem A to a subsystem B [Fan2017]:

$$Q_{A \to B} = -\sum_{i \in A} \sum_{j \in B} \left(\frac{\partial U_i}{\partial \boldsymbol{r}_{ij}} \cdot \boldsymbol{v}_j - \frac{\partial U_j}{\partial \boldsymbol{r}_{ji}} \cdot \boldsymbol{v}_i \right).$$

3.4 Heat transport

3.4.1 EMD method

A popular approach for computing the lattice thermal conductivity is to use equilibrium molecular dynamics (*EMD*) simulations and the Green-Kubo (*GK*) formula. In this method, the running thermal conductivity (*RTC*) along the x-direction (similar expressions apply to other directions) can be expressed as an integral of the heat current autocorrelation (*HAC*) function:

$$\kappa_{xx}(t) = \frac{1}{k_B T^2 V} \int_0^t dt' \text{HAC}_{xx}(t').$$

Here, k_B is Boltzmann's constant, V is the volume of the simulated system, T is the absolute temperature, and t is the correlation time. The *HAC* is

$$\text{HAC}_{xx}(t) = \langle J_x(0) J_x(t) \rangle,$$

where $J_x(0)$ and $J_x(t)$ are the total heat current of the system at two time points separated by an interval of t. The symbol $\langle\rangle$ means that the quantity inside will be averaged over different time origins.

- Related keyword in the *run.in file*: *compute_hac*
- Related output file: *hac.out*

We only used the potential part of the heat current. If you are studying fluids, you need to output the heat currents (potential and kinetic part) using the *compute* keyword and calculated the *HAC* by yourself.

We have decomposed the potential part of the heat current into in-plane and out-of-plane components [Fan2017]. If you do not need this decomposition, you can simply sum up some components in the *hac.out* file.

3.4.2 NEMD method

Non-equilibrium molecular dynamics (*NEMD*) can be used to study thermal transport. In this method, two local thermostats at different temperatures are used to generate a non-equilibrium steady state with a constant heat flux.

If the temperature difference between the two thermostats is ΔT and the heat flux is Q/S, the thermal conductance G between the two thermostats can be calculated as

$$G = \frac{Q/S}{\Delta T}.$$

Here, Q is the energy transfer rate between the thermostat and the thermostated region and S is the cross-sectional area perpendicular to the transport direction.

We can also calculate an effective thermal conductivity (also called the apparent thermal conductivity) $\kappa(L)$ for the finite system:

$$\kappa(L) = GL = \frac{Q/S}{\Delta T/L}.$$

where L is the length between the heat source and the heat sink. This is to say that the temperature gradient should be calculated as $\Delta T/L$, rather than that extracted from the linear part of the temperature profile away from the local thermostats. This is an important conclusion in [Li2019].

To generate the non-equilibrium steady state, one can use a pair of local thermostats. Based on [Li2019], the Langevin thermostatting method is recommended. Therefore, the *ensemble* keyword with the first parameter of `heat_lan` should be used to generate the heat current.

- The *compute* keyword should be used to compute the temperature profile and the heat transfer rate Q.
- Related output file: *compute.out*

3.4.3 HNEMD method

The homogeneous non-equilibrium molecular dynamics (*HNEMD*) method for heat transport by Evans has been generalized to general many-body potentials [Fan2019]. This method is physically equivalent to the *EMD* method but can be computationally faster.

In this method, an external force of the form [Fan2019]

$$\boldsymbol{F}_i^{\mathrm{ext}} = E_i \boldsymbol{F}_{\mathrm{e}} + \sum_{j \neq i} \left(\frac{\partial U_j}{\partial \boldsymbol{r}_{ji}} \otimes \boldsymbol{r}_{ij} \right) \cdot \boldsymbol{F}_{\mathrm{e}}$$

is added to each atom i, driving the system out of equilibrium. According to [Gabourie2021], it can also be written as

$$\boldsymbol{F}_i^{\mathrm{ext}} = E_i \boldsymbol{F}_{\mathrm{e}} + \boldsymbol{F}_{\mathrm{e}} \cdot \mathbf{W}_i$$

Here, E_i is the total energy of particle i. U_i is the potential energy of particle i. $\mathbf{W}_i$ is the per-atom virial. $\boldsymbol{r}_{ij} \equiv \boldsymbol{r}_j - \boldsymbol{r}_i$, and $\boldsymbol{r}_i$ is the position of particle i.

The parameter $\boldsymbol{F}_{\mathrm{e}}$ is of the dimension of inverse length and should be small enough to keep the system within the linear response regime. The driving force will induce a non-equilibrium heat current $\langle \boldsymbol{J} \rangle_{\mathrm{ne}}$ linearly related to $\boldsymbol{F}_{\mathrm{e}}$:

$$\frac{\langle J^\mu(t) \rangle_{\mathrm{ne}}}{TV} = \sum_\nu \kappa^{\mu\nu} F_{\mathrm{e}}^\nu,$$

where $\kappa^{\mu\nu}$ is the thermal conductivity tensor, T is the system temperature, and V is the system volume.

A global thermostat should be applied to control the temperature of the system. For this, we recommend using the Nose-Hoover chain thermostat. So one should use the *ensemble* keyword with the first parameter of `nvt_nhc`.

- The *compute_hnemd* keyword should be used to add the driving force and calculate the thermal conductivity.
- The computed results are saved to the *kappa.out* file.

3.4.4 Spectral heat current

In the framework of the *NEMD* and *HNEMD* methods, one can also calculate spectrally decomposed thermal conductivity (or conductance). In this method, one first calculates the following virial-velocity correlation function [Gabourie2021]:

$$\boldsymbol{K}(t) = \sum_i \langle \mathbf{W}_i(0) \cdot \boldsymbol{v}_i(t) \rangle \,,$$

which reduces to the non-equilibrium heat current when $t = 0$.

Then one can define the following Fourier transform pairs [Fan2017]:

$$\tilde{\boldsymbol{K}}(\omega) = \int_{-\infty}^{\infty} dt e^{i\omega t} K(t)$$

where

$$\boldsymbol{K}(t) = \int_{-\infty}^{\infty} \frac{d\omega}{2\pi} e^{-i\omega t} \tilde{\boldsymbol{K}}(\omega)$$

By setting $t = 0$ in the equation above, we can get the following spectral decomposition of the non-equilibrium heat current:

$$\boldsymbol{J} = \int_0^{\infty} \frac{d\omega}{2\pi} \left[2\tilde{\boldsymbol{K}}(\omega) \right].$$

From the spectral decomposition of the non-equilibrium heat current, one can deduce the spectrally decomposed thermal conductance in the *NEMD* method:

$$G(\omega) = \frac{2\tilde{\boldsymbol{K}}(\omega)}{V\Delta T}$$

with

$$G = \int_0^\infty \frac{d\omega}{2\pi} G(\omega).$$

where ΔT is the temperature difference between the two thermostats and V is the volume of the considered system or subsystem.

One can also calculate the spectrally decomposed thermal conductivity in the *HNEMD* method:

$$\kappa(\omega) = \frac{2\tilde{\boldsymbol{K}}(\omega)}{VTF_{\mathrm{e}}}$$

with

$$\kappa = \int_0^\infty \frac{d\omega}{2\pi} \kappa(\omega).$$

where F_{e} is the magnitude of the driving force parameter in the *HNEMD* method.

This calculation is invoked by the *compute_shc* keyword and the results are saved to the *shc.out* file.

3.4.5 Modal analysis methods

A system with N atoms will have $3N$ vibrational modes. Using lattice dynamics, the vibrational modes (or eigenmodes) of the system can be found. The heat flux can be decomposed into contributions from each vibrational mode and the thermal conductivity can be written in terms of those contributions [Lv2016]. To calculate the modal heat current in GPUMD, the velocities must first be decomposed into their modal contributions:

$$\boldsymbol{v}_i(t) = \frac{1}{\sqrt{m_i}} \sum_n \boldsymbol{e}_{i,n} \cdot \dot{\boldsymbol{X}}_n(t)$$

Here, $\dot{\boldsymbol{X}}_n$ is the normal mode coordinates of the velocity of mode n m_i is the mass of atom i $\boldsymbol{e}_{i,n}$ is the eigenvector that gives the magnitude and direction of mode n for atom i $\boldsymbol{v}_i$ is the velocity of atom i

The heat current can be rewritten in terms of the modal velocity to be:

$$\boldsymbol{J}^{\mathrm{pot}} = \sum_i \mathbf{W}_i \cdot \left[\frac{1}{\sqrt{m_i}} \sum_n \boldsymbol{e}_{i,n} \cdot \dot{\boldsymbol{X}}_n(t) \right] = \sum_n \left(\sum_i \frac{1}{\sqrt{m_i}} \mathbf{W}_i \cdot \boldsymbol{e}_{i,n} \right) \cdot \dot{\boldsymbol{X}}_n(t)$$

This means that the modal heat current can be written as:

$$\boldsymbol{J}_n^{\mathrm{pot}} = \left(\sum_i \frac{1}{\sqrt{m_i}} \mathbf{W}_i \cdot \boldsymbol{e}_{i,n} \right) \cdot \dot{\boldsymbol{X}}_n(t)$$

This modal heat current can be used to extend the capabilities of the *EMD* and *HNEMD* methods. The extended methods are called Green-Kubo modal analysis (*GKMA*) [Lv2016] and homogeneous non-equilibrium modal analysis (*HNEMA*) [Gabourie2021].

Green-Kubo modal analysis

The Green-Kubo Modal Analysis (*GKMA*) calculates the modal contributions to thermal conductivity by using [Lv2016] [Gabourie2021]:

$$\kappa_{xx,n}(t) = \frac{1}{k_B T^2 V} \int_0^t dt' \langle J_{x,n}(t') J_x(0) \rangle.$$

Here, $k_{\rm B}$ is Boltzmann's constant, V is the volume of the simulated system, T is the absolute temperature, and t is the correlation time. $J_x(0)$ is the total heat current and and:math:*J_{x,n}(t')* is the mode-specific heat current of the system at two time points separated by an interval of t'. The symbol $\langle\rangle$ means that the quantity inside will be averaged over different time origins.

- Related input file: *eigenvector.in*
- Related keyword in the *run.in file*: *compute_gkma*
- Related output file: *heatmode.out*

For the *GKMA* method, we only used the potential part of the heat current.

Homogeneous non-equilibrium modal analysis

The homogeneous non-equilibrium modal analysis (*HNEMA*) method calculates the modal contributions of thermal conductivity using [Gabourie2021]:

$$\frac{\langle J_n^\mu(t) \rangle_{\rm ne}}{TV} = \sum_\nu \kappa_n^{\mu\nu} F_{\rm e}^\nu,$$

Here, $\kappa_n^{\mu\nu}$ is the thermal conductivity tensor of mode n, T is the system temperature, and V is the system volume. The mode-specific non-equilibrium heat current is $\langle J_n^\mu(t) \rangle_{\rm ne}$ and the driving force parameter is $\boldsymbol{F}_{\rm e}$.

- Related input file: *eigenvector.in*
- Related keyword in the run.in file: *compute_hnema*
- Related output file: *kappamode.out*

For the *HNEMA* method, we only used the potential part of the heat current. A global thermostat should be applied to control the temperature of the system. For this, we recommend using the Nose-Hoover chain thermostat. So one should use the *ensemble* keyword with the first parameter of `nvt_nhc`.

3.4.6 HNEMDEC method

A system with M components has M independent fluxes: the heat flux and any $M-1$ momentum fluxes of the M component. The central idea of Evans-Cummings algorithm is designing such a driving force that produce a dissipative flux that is equivalent to heat flux or momentum flux. By measuring the heat current and momentum current, we obtain onsager coefficents that can be used to derive the thermal conductivity.

In the case of heat flux $\boldsymbol{J}_q$ as dissipative flux, for the i atom belonging to α component:

$$\boldsymbol{F}_i^{\alpha,\rm ext} = (\mathbf{S}_i^\alpha - \frac{m_\alpha}{M}\mathbf{S} + k_B T \frac{M_{tot} - N m_\alpha}{M_{tot} N}\mathbf{I}) \cdot \boldsymbol{F}_{\rm e}$$

where $\mathbf{S}_i^\alpha = E_i^\alpha \mathbf{I} + \mathbf{W}_i^\alpha$, $\mathbf{S} = \sum_{\beta=1}^{M} \sum_{i=1}^{N_\beta} \mathbf{S}_i^\beta$, M_{tot} is the total mass of the system, N is the atom number of the system. Any physical quantity $A(t)$ is related to driving force by correlation funtion:

$$\langle \boldsymbol{A}(t) \rangle = \langle A(0) \rangle + (\int_0^t dt' \frac{\langle \boldsymbol{A}(t') \otimes \boldsymbol{J}_q(0) \rangle}{k_B T}) \cdot \boldsymbol{F}_{\rm e}$$

In the case of momentum flux $\boldsymbol{J}_\gamma$ of γ component as dissipative flux:

$$\boldsymbol{F}_i^{\alpha,\mathrm{ext}} = c_\alpha \boldsymbol{F}_\mathrm{e}$$

where $c_\gamma = \frac{N}{N_\gamma}$, and $c_\beta = -\frac{N m_\beta}{M'}$, $M' = \sum_{\epsilon=1,\epsilon\neq\gamma}^{M} N_\epsilon m_\epsilon$. Similar to the former case,

$$\langle \boldsymbol{A}(t) \rangle = \langle A(0) \rangle + (\frac{N}{M'} + \frac{N}{N_\gamma m_\gamma})(\int_0^t dt' \frac{\langle \boldsymbol{A}(t') \otimes \boldsymbol{J}_\gamma(0) \rangle}{k_B T}) \cdot \boldsymbol{F}_\mathrm{e}$$

Then we can obtain any matrix element Λ_{ij} of onsager matrix by:

$$\Lambda_{ij} = \frac{1}{k_B V} \int_0^t dt' \langle \boldsymbol{J}_i(t') \otimes \boldsymbol{J}_j(0) \rangle$$

The onsager matrix is arranged as:

$$\begin{matrix} \Lambda_{qq} & \Lambda_{q1} & \cdots & \Lambda_{q(M-1)} \\ \Lambda_{1q} & \Lambda_{11} & \cdots & \Lambda_{1(M-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda_{(M-1)q} & \Lambda_{(M-1)1} & \cdots & \Lambda_{(M-1)(M-1)} \end{matrix}$$

The thermal conductivity could be derived from onsager matrix:

$$\kappa = \frac{1}{T^2 (\Lambda^{-1})_{00}}$$

- The *compute_hnemdec* keyword should be used to add the driving force and calculate the thermal conductivity.
- The computed results are saved to the *onsager.out* file.

CHAPTER

FOUR

INTERATOMIC POTENTIALS

4.1 Tersoff potential (1988)

The implementation of the Tersoff (1988) potential in **GPUMD** mimics the one in lammps [Tersoff1988]. The Tersoff-1988 potential has a more general form than the *Tersoff (1989) potential*. When possible, it is, however, recommended to use the Tersoff (1989) potential as it is faster.

4.1.1 Potential form

The site potential can be written as

$$U_i = \frac{1}{2} \sum_{j \neq i} f_C(r_{ij}) \left[f_R(r_{ij}) - b_{ij} f_A(r_{ij}) \right].$$

The function f_C is a cutoff function, which is 1 when $r_{ij} < R$ and 0 when $r_{ij} > S$ and takes the following form in the intermediate region:

$$f_C(r_{ij}) = \frac{1}{2} \left[1 + \cos\left(\pi \frac{r_{ij} - R}{S - R} \right) \right].$$

The repulsive function f_R and the attractive function f_A take the following forms:

$$f_R(r) = A e^{-\lambda r_{ij}}$$
$$f_A(r) = B e^{-\mu r_{ij}}.$$

The bond-order is

$$b_{ij} = \left(1 + \beta^n \zeta_{ij}^n\right)^{-\frac{1}{2n}},$$

where

$$\zeta_{ij} = \sum_{k \neq i,j} f_C(r_{ik}) g_{ijk} e^{\alpha (r_{ij} - r_{ik})^m}$$
$$g_{ijk} = \gamma \left(1 + \frac{c^2}{d^2} - \frac{c^2}{d^2 + (h - \cos\theta_{ijk})^2} \right).$$

Parameter	Unit
A	eV
B	eV
λ	Å^{-1}
μ	Å^{-1}
β	dimensionless
n	dimensionless
c	dimensionless
d	dimensionless
h	dimensionless
R	Å
S	Å
m	dimensionless
α	Å^{-m}
γ	dimensionless

4.1.2 File format

We have adopted a file format that similar but not identical to that used by lammps.

The potential file for a single-element system reads:

```
tersoff_1988 1 element
A_000 B_000 lambda_000 mu_000 beta_000 n_000 c_000 d_000 h_000 R_000 S_000 m_000 alpha_
↪000 gamma_000
```

Here, `element` is the chemical symbol of the element.

The potential file for a double-element system reads:

```
tersoff_1988 2 <list of the 2 elements>
A_000 B_000 lambda_000 mu_000 beta_000 n_000 c_000 d_000 h_000 R_000 S_000 m_000 alpha_
↪000 gamma_000
A_001 B_001 lambda_001 mu_001 beta_001 n_001 c_001 d_001 h_001 R_001 S_001 m_001 alpha_
↪001 gamma_001
A_010 B_010 lambda_010 mu_010 beta_010 n_010 c_010 d_010 h_010 R_010 S_010 m_010 alpha_
↪010 gamma_010
A_011 B_011 lambda_011 mu_011 beta_011 n_011 c_011 d_011 h_011 R_011 S_011 m_011 alpha_
↪011 gamma_011
A_100 B_100 lambda_100 mu_100 beta_100 n_100 c_100 d_100 h_100 R_100 S_100 m_100 alpha_
↪100 gamma_100
A_101 B_101 lambda_101 mu_101 beta_101 n_101 c_101 d_101 h_101 R_101 S_101 m_101 alpha_
↪101 gamma_101
A_110 B_110 lambda_110 mu_110 beta_110 n_110 c_110 d_110 h_110 R_110 S_110 m_110 alpha_
↪110 gamma_110
A_111 B_111 lambda_111 mu_111 beta_111 n_111 c_111 d_111 h_111 R_111 S_111 m_111 alpha_
↪111 gamma_111
```

The extension to more than two components is accordingly.

4.2 Tersoff potential (1989)

The Tersoff (1989) potential supports systems with one or two atom types [Tersoff1989]. It is less general than the *Tersoff (1988) potential* but faster.

4.2.1 Potential form

Below we use $i, j, k, \cdots$ for atom indices and $I, J, K, \cdots$ for atom types.

The site potential can be written as

$$U_i = \frac{1}{2} \sum_{j \neq i} f_{\mathrm{C}}(r_{ij}) \left[f_{\mathrm{R}}(r_{ij}) - b_{ij} f_{\mathrm{A}}(r_{ij}) \right].$$

The function f_{C} is a cutoff function, which is 1 when $r_{ij} < R_{IJ}$ and 0 when $r_{ij} > S_{IJ}$ and takes the following form in the intermediate region:

$$f_{\mathrm{C}}(r_{ij}) = \frac{1}{2} \left[1 + \cos\left(\pi \frac{r_{ij} - R_{IJ}}{S_{IJ} - R_{IJ}} \right) \right].$$

The repulsive function f_{R} and the attractive function f_{A} take the following forms:

$$f_{\mathrm{R}}(r) = A_{IJ} e^{-\lambda_{IJ} r_{ij}}$$
$$f_{\mathrm{A}}(r) = B_{IJ} e^{-\mu_{IJ} r_{ij}}.$$

The bond-order function is

$$b_{ij} = \chi_{IJ} \left(1 + \beta_I^{n_I} \zeta_{ij}^{n_I} \right)^{-\frac{1}{2n_I}},$$

where

$$\zeta_{ij} = \sum_{k \neq i,j} f_{\mathrm{C}}(r_{ik}) g_{ijk}$$
$$g_{ijk} = 1 + \frac{c_I^2}{d_I^2} - \frac{c_I^2}{d_I^2 + (h_I - \cos\theta_{ijk})^2}.$$

Parameter	Unit
A_{IJ}	eV
B_{IJ}	eV
λ_{IJ}	Å^{-1}
μ_{IJ}	Å^{-1}
β_I	dimensionless
n_I	dimensionless
c_I	dimensionless
d_I	dimensionless
h_I	dimensionless
R_{IJ}	Å
S_{IJ}	Å
χ_{IJ}	dimensionless

4.2.2 File format

Single-element systems

In this case, χ_{IJ} is irrelevant. The potential file reads:

```
tersoff_1989 1 <element>
A B lambda mu beta n c d h R S
```

Here, `element` is the chemical symbol of the element.

Two-element systems

In this case, there are two sets of parameters, one for each atom type. The following mixing rules are used to determine some parameters between the two atom types i and j:

$$
\begin{aligned}
A_{IJ} &= \sqrt{A_{II}A_{JJ}} \\
B_{IJ} &= \sqrt{B_{II}B_{JJ}} \\
R_{IJ} &= \sqrt{R_{II}R_{JJ}} \\
S_{IJ} &= \sqrt{S_{II}S_{JJ}} \\
\lambda_{IJ} &= (\lambda_{II} + \lambda_{JJ})/2 \\
\mu_{IJ} &= (\mu_{II} + \mu_{JJ})/2.
\end{aligned}
$$

Here, the parameter $\chi_{01} = \chi_{10}$ needs to be provided. $\chi_{00} = \chi_{11} = 1$ by definition.

The potential file reads:

```
tersoff_1989 2 <list of the 2 elements>
A_0 B_0 lambda_0 mu_0 beta_0 n_0 c_0 d_0 h_0 R_0 S_0
A_1 B_1 lambda_1 mu_1 beta_1 n_1 c_1 d_1 h_1 R_1 S_1
chi_01
```

4.3 Tersoff mini-potential

The Tersoff-mini potential is described in [Fan2020]. It currently only applies to systems with a single atom type. One can use the GPUGA potential to fit this potential for new systems.

4.3.1 Potential form

The site potential can be written as

$$
U_i = \frac{1}{2}\sum_{j\neq i} f_{\mathrm{C}}(r_{ij})\left[f_{\mathrm{R}}(r_{ij}) - b_{ij}f_{\mathrm{A}}(r_{ij})\right].
$$

The function f_{C} is a cutoff function, which is 1 when $r_{ij} < R_{IJ}$ and 0 when $r_{ij} > S_{IJ}$ and takes the following form in the intermediate region:

$$
f_{\mathrm{C}}(r_{ij}) = \frac{1}{2}\left[1 + \cos\left(\pi\frac{r_{ij} - R}{S - R}\right)\right].
$$

The repulsive function f_{R} and the attractive function f_{A} take the following forms:

$$
\begin{aligned}
f_{\mathrm{R}}(r_{ij}) &= \frac{D_0}{S-1}\exp\left(\alpha r_0\sqrt{2S}\right)e^{-\alpha\sqrt{2S}r_{ij}} \\
f_{\mathrm{A}}(r_{ij}) &= \frac{D_0 S}{S-1}\exp\left(\alpha r_0\sqrt{2/S}\right)e^{-\alpha\sqrt{2/S}r_{ij}}.
\end{aligned}
$$

The bond-order function is

$$
b_{ij} = \left(1 + \zeta_{ij}^n\right)^{-\frac{1}{2n}},
$$

where

$$\zeta_{ij} = \sum_{k \neq i,j} f_C(r_{ik}) g_{ijk}$$
$$g_{ijk} = \beta \left(h - \cos\theta_{ijk}\right)^2.$$

4.3.2 Parameters

Parameter	Unit
D_0	eV
α	Å^{-1}
r_0	Å
S	dimensionless
n	dimensionless
β	dimensionless
h	dimensionless
R	Å
S	Å

4.3.3 File format

The potential file reads:

```
tersoff_mini 1 element
D alpha r0 S beta n h R S
```

Here, `element` is the chemical symbol of the element.

4.4 Lennard-Jones potential

The Lennard-Jones (*LJ*) potential is one of the simplest two-body potentials used in *MD* simulations. The pair potential between particles i and j is

$$U_{ij} = 4\epsilon_{ij}\left(\frac{\sigma_{ij}^{12}}{r_{ij}^{12}} - \frac{\sigma_{ij}^{6}}{r_{ij}^{6}}\right).$$

For the implementation in **GPUMD** the potential has neither been shifted nor damped. This is important to keep in mind when using this model as it implies that both energy and force change discontinuously at the cutoff.

The implementation in **GPUMD** supports up to 10 atom types.

There are two parameters, which respectively control the depth (ϵ) and the position (σ) of the potential well. They are given in units of eV and Å, respectively.

4.4.1 File format

If there is only one atom type, the potential file for this potential model reads:

```
lj 1 element
epsilon sigma cutoff
```

Here, `cutoff` is the cutoff distance and `element` is the chemical symbol of the element.

If there are two atom types, the potential file reads:

```
lj 2 <list of the 2 elements>
epsilon_00 sigma_00 cutoff_00
epsilon_01 sigma_01 cutoff_01
epsilon_10 sigma_10 cutoff_10
epsilon_11 sigma_11 cutoff_11
```

If there are three atom types, the potential file reads:

```
lj 3 <list of the 3 elements>
epsilon_00 sigma_00 cutoff_00
epsilon_01 sigma_01 cutoff_01
epsilon_02 sigma_02 cutoff_02
epsilon_10 sigma_10 cutoff_10
epsilon_11 sigma_11 cutoff_11
epsilon_12 sigma_12 cutoff_12
epsilon_20 sigma_20 cutoff_20
epsilon_21 sigma_21 cutoff_21
epsilon_22 sigma_22 cutoff_22
```

The extension to more than three atom types should be apparent from the examples above.

4.5 Embedded atom method

GPUMD suppports two different analytical forms of embedded atom method (*EAM*) potentials. Using the form by Zhou *et al.* can simulate alloys with up to 10 atom types [Zhou2004], while using the form of Dai *et al.* the implementation only applies to systems with a single atom type [Dai2006].

4.5.1 Potential form

General form

The site potential energy is

$$U_i = \frac{1}{2}\sum_{j\neq i}\phi(r_{ij}) + F(\rho_i).$$

Here, the part with $\phi(r_{ij})$ is a pairwise potential and $F(\rho_i)$ is the embedding potential, which depends on the electron density ρ_i at site i. The many-body part of the EAM potential comes from the embedding potential.

The density ρ_i is contributed by the neighbors of i:

$$\rho_i = \sum_{j\neq i} f(r_{ij}).$$

Therefore, the form of an *EAM* potential is completely determined by the three functions: ϕ, f, and F.

Version from [Zhou2004]

The pair potential between two atoms of the same type a is

$$\phi^{aa}(r) = \frac{A^a \exp[-\alpha(r/r_e^a - 1)]}{1 + (r/r_e^a - \kappa^a)^{20}} - \frac{B^a \exp[-\beta(r/r_e^a - 1)]}{1 + (r/r_e^a - \lambda^a)^{20}}.$$

The contribution of the electron density from an atom of type a is

$$f^a(r) = \frac{f_e^a \exp[-\beta(r/r_e^a - 1)]}{1 + (r/r_e^a - \lambda^a)^{20}}.$$

The pair potential between two atoms of different types a and b is then constructed as

$$\phi^{ab}(r) = \frac{1}{2}\left[\frac{f^b(r)}{f^a(r)}\phi^{aa}(r) + \frac{f^a(r)}{f^b(r)}\phi^{bb}(r)\right].$$

The embedding energy function is piecewise:

$$F(\rho) = \begin{cases} \sum_{i=0}^{3} F_{ni}\left(\frac{\rho}{\rho_n} - 1\right)^i & \rho < 0.85\rho_e \\ \sum_{i=0}^{3} F_i\left(\frac{\rho}{\rho_e} - 1\right)^i & 0.85\rho_e \le \rho < 1.15\rho_e \\ F_e\left[1 - \ln\left(\frac{\rho}{\rho_s}\right)^\eta\right]\left(\frac{\rho}{\rho_s}\right)^\eta & \rho \ge 1.15\rho_e \end{cases}$$

Version from [Dai2006]

This is a very simple *EAM*-type potential, which is an extension of the Finnis-Sinclair potential. The function for the pair potential is

$$\phi(r) = \begin{cases} (r-c)^2 \sum_{n=0}^{4} c_n r^n & r \le c \\ 0 & r > c \end{cases}$$

The function for the density is

$$f(r) = \begin{cases} (r-d)^2 + B^2(r-d)^4 & r \le d \\ 0 & r > d \end{cases}$$

The function for the embedding energy is

$$F(\rho) = -A\rho^{1/2}.$$

4.5.2 File format

The potential file for the version from [Zhou2004] reads:

```
eam_zhou_2004 num_types <list of elements>
r_e f_e rho_e rho_s alpha beta A B kappa lambda F_n0 F_n1 F_n2 F_n3 F_0 F_1 F_2 F_3 eta␣
↪F_e cutoff
```

There are `num_types` rows of parameters but here we have only written a single row above. The order of the rows should be consistent with the `<list of elements>` in the first line.

The last parameter `cutoff` is the cutoff distance, which is not intrinsic to the model. The order of the parameters is the same as in Table III of [Zhou2004]. For multi-component systems, **GPUMD** will use the largest cutoff for every atom type.

The potential file for the version from [Dai2006] reads:

```
eam_dai_2006 1 Element
A d c c_0 c_1 c_2 c_3 c_4 B
```

Here, `Element` is the chemical symbol of the element.

4.5.3 Table format

GPUMD supports the eam/alloy format as defined in LAMMPS.

Note

The user needs to modify the first line of the potential file as follows:

eam/alloy <num_types> <list of elements>

The last two parts can be directly copied from the fourth line of the original potential file. For example:

```
eam/alloy 2 Cu Ni
```

Since the first three lines are comments, this modification does not affect usage in LAMMPS.

4.6 Force constant potential

The force constant potential (*FCP*) is obtained through a systematic expansion of the energy in displacements as described [Brorsson2021].

4.6.1 Potential form

In the force constant potential, the potential energy is calculated as a Taylor expansion in terms of the atomic displacements $\{u_i^a\}$ relative to a set of reference (equilibrium) positions as

$$U = U_2 + U_3 + U_4 + U_5 + U_6 + \cdots$$

with

$$U_2 = \frac{1}{2!} \sum_{ij} \sum_{ab} \Phi_{ij}^{ab} u_i^a u_j^b$$

$$U_3 = \frac{1}{3!} \sum_{ijk} \sum_{abc} \Phi_{ijk}^{abc} u_i^a u_j^b u_k^c$$

$$U_4 = \frac{1}{4!} \sum_{ijkl} \sum_{abcd} \Phi_{ijkl}^{abcd} u_i^a u_j^b u_k^c u_l^d$$

$$U_5 = \frac{1}{5!} \sum_{ijklm} \sum_{abcde} \Phi_{ijklm}^{abcde} u_i^a u_j^b u_k^c u_l^d u_m^e$$

$$U_6 = \frac{1}{6!} \sum_{ijklmn} \sum_{abcdef} \Phi_{ijklmn}^{abcdef} u_i^a u_j^b u_k^c u_l^d u_m^e u_n^f.$$

Here, Φ_{ij}^{ab}, Φ_{ijk}^{abc}, Φ_{ijkl}^{abcd}, Φ_{ijklm}^{abcde}, and Φ_{ijklmn}^{abcdef} are the second-order, third-order, fourth-order, fifth-order, and sixth-order force constants. The indices i, j, k, l, m, and n refer to the atoms and can take integer values from 0 to $N-1$, where N is the number of atoms in the system. The indices a, b, c, d, e, and f refer to the axes in the Cartesian coordinate system and can take integer values 0, 1, and 2, which correspond to the x, y, and z axes, respectively. In **GPUMD**, we only consider force constants up to sixth order.

4.6.2 File format

One needs to prepare several files related to this potential, which can be conveniently generated using the hiphive package [Eriksson2019], except for the driver potential file below (which is very easy to prepare).

driver file

The driver potential file for this potential model reads:

```
fcp number_of_atom_types <list of atom symbols>
highest_force_order highest_heat_current_order
path_to_force_constant_files
```

- `fcp` is the name of this potential and tells the code that we are using the force constant potential.
- `number_of_atom_types` is the number of atom types defined in the *simulation model file*.
- `<list of atom symbols>` is a list of all the elements in the potential (can be in any order).
- `highest_force_order` is the highest order of the force constants used in the potential. For example, when `highest_order` is 4, second-order, third-order, and fourth-order forces constants will be used (and should be prepared).
- `highest_heat_current_order` is the highest order of the force constants used for heat current calculations. It can only be 2 or 3, which means using the second order force constants only or using both the second and third order force constants for heat current calculations, respectively.
- `path_to_force_constant_files` is the path to the force constant files (see below). **Important:** There should be no trailing slash (`/`) after the folder name.

force constant files

The force constant data should be prepared in some files named:

```
clusters_order2.in
clusters_order3.in
clusters_order4.in
clusters_order5.in
clusters_order6.in
fcs_order2.in
fcs_order3.in
fcs_order4.in
fcs_order5.in
fcs_order6.in
```

These files should be in the folder you specified in the driver potential file (see above). If you only consider force constants up to the 4th order, you do not need the files with numbers 5 and 6. These files can be generated by the hiphive package. Here, we therefore do not describe the formats of these files.

equilibrium position file

Because this potential is defined in terms of the atomic displacements, one has to define the equilibrium (reference) positions of the atoms in the system. A file called `r0.in` is used for this purpose. This file should be in the folder you specified in the driver potential file (see above). The format of this file is:

```
x_0 y_0 z_0
x_1 y_1 z_1
x_2 y_2 z_2
x_3 y_3 z_3
...
```

where each line gives the position of one atom. The order of the atoms should be consistent with that in the *simulation model file*. The coordinates are in units of Ångstrom. This file is also generated by the hiphive package.

4.7 Neuroevolution potential

The neuroevolution potential (*NEP*) approach was proposed in [Fan2021] (NEP1) and later improved in [Fan2022a] (NEP2), [Fan2022b] (NEP3), and [Song2024] (NEP4). Currently, **GPUMD** only supports NEP3 and NEP4, as NEP1 and NEP2 are deprecated. Both versions are identical for single-component systems. For multi-component systems, NEP4 usually has higher accuracy, if all the other hyperparameters are the same.

GPUMD not only allows one to carry out simulations using *NEP* models via the *gpumd executable* but even the construction of such models via the *nep executable*.

4.7.1 The neural network model

NEP uses a simple feedforward neural network (*NN*) to represent the site energy of atom i as a function of a descriptor vector with $N_{\rm des}$ components,

$$U_i(\mathbf{q}) = U_i\left(\{q_\nu^i\}_{\nu=1}^{N_{\rm des}}\right).$$

There is a single hidden layer with $N_{\rm neu}$ neurons in the NN and we have

$$U_i = \sum_{\mu=1}^{N_{\rm neu}} w_\mu^{(1)} \tanh\left(\sum_{\nu=1}^{N_{\rm des}} w_{\mu\nu}^{(0)} q_\nu^i - b_\mu^{(0)}\right) - b^{(1)},$$

where $\tanh(x)$ is the activation function in the hidden layer, $\mathbf{w}^{(0)}$ is the connection weight matrix from the input layer (descriptor vector) to the hidden layer, $\mathbf{w}^{(1)}$ is the connection weight vector from the hidden layer to the output node, which is the energy U_i, $\mathbf{b}^{(0)}$ is the bias vector in the hidden layer, and $b^{(1)}$ is the bias for node U_i.

4.7.2 The descriptor

The descriptor for atom i consists of a number of radial and angular components as described below.

The **radial descriptor** components are defined as

$$q_n^i = \sum_{j\neq i} g_n(r_{ij})$$

with

$$0 \leq n \leq n_{\rm max}^{\rm R},$$

where the summation runs over all the neighbors of atom i within a certain cutoff distance. There are thus $n_{\rm max}^{\rm R}+1$ radial descriptor components.

For the **angular descriptor** components, we consider 3-body to 5-body ones. The formulation is similar but not identical to the atomic cluster expansion (*ACE*) approach [Drautz2019]. For 3-body ones, we define ($0 \leq n \leq n_{\rm max}^{\rm A}$, $1 \leq l \leq l_{\rm max}^{\rm 3b}$)

$$q_{nl}^i = \sum_{m=-l}^{l} (-1)^m A_{nlm}^i A_{nl(-m)}^i,$$

where

$$A_{nlm}^i = \sum_{j\neq i} g_n(r_{ij}) Y_{lm}(\theta_{ij}, \phi_{ij}),$$

and $Y_{lm}(\theta_{ij}, \phi_{ij})$ are the spherical harmonics as a function of the polar angle θ_{ij} and the azimuthal angle ϕ_{ij}. For expressions of the 4-body and 5-body descriptor components, we refer to [Fan2022b].

The radial functions $g_n(r_{ij})$ appear in both the radial and the angular descriptor components. In the radial descriptor components,

$$g_n(r_{ij}) = \sum_{k=0}^{N_{\rm bas}^{\rm R}} c_{nk}^{ij} f_k(r_{ij}),$$

with

$$f_k(r_{ij}) = \frac{1}{2}\left[T_k\left(2\left(r_{ij}/r_{\rm c}^{\rm R}-1\right)^2-1\right)+1\right] f_{\rm c}(r_{ij}),$$

and

$$f_{\rm c}(r_{ij}) = \begin{cases} \frac{1}{2}\left[1+\cos\left(\pi\frac{r_{ij}}{r_{\rm c}^{\rm R}}\right)\right], & r_{ij} \le r_{\rm c}^{\rm R}; \\ 0, & r_{ij} > r_{\rm c}^{\rm R}. \end{cases}$$

In the angular descriptor components, $g_n(r_{ij})$ have similar forms but with $N_{\rm bas}^{\rm R}$ changed to $N_{\rm bas}^{\rm A}$ and with $r_{\rm c}^{\rm R}$ changed to $r_{\rm c}^{\rm A}$.

4.7.3 Model dimensions

Number of ...	Count
atom types	$N_{\rm typ}$
radial descriptor components	$n_{\rm max}^{\rm R}+1$
3-body angular descriptor components	$(n_{\rm max}^{\rm A}+1)l_{\rm max}^{\rm 3b}$
4-body angular descriptor components	$(n_{\rm max}^{\rm A}+1)$ or zero (if not used)
5-body angular descriptor components	$(n_{\rm max}^{\rm A}+1)$ or zero (if not used)
descriptor components	$N_{\rm des}$ is the sum of the above numbers of descriptor components
trainable parameters c_{nk}^{ij} in the descriptor	$N_{\rm typ}^2[(n_{\rm max}^{\rm R}+1)(N_{\rm bas}^{\rm R}+1)+(n_{\rm max}^{\rm A}+1)(N_{\rm bas}^{\rm A}+1)]$
trainable *NN* parameters	$N_{\rm nn}=(N_{\rm des}+2)N_{\rm neu}+1$ (NEP3)
	$N_{\rm nn}=(N_{\rm des}+2)N_{\rm neu}N_{\rm typ}+1$ (NEP4)

The total number of trainable parameters is the sum of the number of trainable descriptor parameters and the number of *NN* parameters $N_{\rm nn}$.

4.7.4 Optimization procedure

The name of the *NEP* model is owed to the use of the separable natural evolution strategy (*SNES*) that is used for the optimization of the parameters [Schaul2011]. The interested reader is referred to [Schaul2011] and [Fan2021] for details.

The key quantity in the optimization procedure is the loss (or objective) function, which is being minimized. It is defined as a weighted sum over the loss terms associated with energies, forces and virials as well as the $\mathcal{L}_1$ and $\mathcal{L}_2$

norms of the parameter vector.

$$
\begin{aligned}
L(\boldsymbol{z}) = & \lambda_{\mathrm{e}} \left(\frac{1}{N_{\mathrm{str}}} \sum_{n=1}^{N_{\mathrm{str}}} \left(U^{\mathrm{NEP}}(n, \boldsymbol{z}) - U^{\mathrm{tar}}(n) \right)^2 \right)^{1/2} \\
& + \lambda_{\mathrm{f}} \left(\frac{1}{3N} \sum_{i=1}^{N} \left(\boldsymbol{F}_i^{\mathrm{NEP}}(\boldsymbol{z}) - \boldsymbol{F}_i^{\mathrm{tar}} \right)^2 \right)^{1/2} \\
& + \lambda_{\mathrm{v}} \left(\frac{1}{6N_{\mathrm{str}}} \sum_{n=1}^{N_{\mathrm{str}}} \sum_{\mu\nu} \left(W_{\mu\nu}^{\mathrm{NEP}}(n, \boldsymbol{z}) - W_{\mu\nu}^{\mathrm{tar}}(n) \right)^2 \right)^{1/2} \\
& + \lambda_1 \frac{1}{N_{\mathrm{par}}} \sum_{n=1}^{N_{\mathrm{par}}} |z_n| \\
& + \lambda_2 \left(\frac{1}{N_{\mathrm{par}}} \sum_{n=1}^{N_{\mathrm{par}}} z_n^2 \right)^{1/2}.
\end{aligned}
$$

Here, N_{str} is the number of structures in the training data set (if using a full batch) or the number of structures in a mini-batch (see the *batch keyword* in the *nep.in input file*) and N is the total number of atoms in these structures. $U^{\mathrm{NEP}}(n, \boldsymbol{z})$ and $W_{\mu\nu}^{\mathrm{NEP}}(n, \boldsymbol{z})$ are the per-atom energy and virial tensor predicted by the *NEP* model with parameters $\boldsymbol{z}$ for the n^{th} structure, and $\boldsymbol{F}_i^{\mathrm{NEP}}(\boldsymbol{z})$ is the predicted force for the i^{th} atom. $U^{\mathrm{tar}}(n)$, $W_{\mu\nu}^{\mathrm{tar}}(n)$, and $\boldsymbol{F}_i^{\mathrm{tar}}$ are the corresponding target values. That is, the loss terms for energies, forces, and virials are defined as the respective *RMSE* values between the *NEP* predictions and the target values. The last two terms represent $\mathcal{L}_1$ and $\mathcal{L}_2$ regularization terms of the parameter vector. The weights λ_{e}, λ_{f}, λ_{v}, λ_1, and λ_2 are tunable hyper-parameters (see the eponymous keywords in the *nep.in input file*). When calculating the loss function, we use eV/atom for energies and virials and eV/Å for force components.

4.8 Hybrid NEP+ILP potential

The hybrid *NEP* + *ILP* potential [Bu2025] in **GPUMD** combines the neuroevolution potential (*NEP*), [Fan2022b] (NEP3), and [Song2024] (NEP4), for intralayer interactions and the interlayer potential (*ILP*) [Ouyang2018] [Ouyang2020] for interlayer interactions to simulate van der Waals materials. Now this hybrid potential supports to simulate homo- and heterostructures based on graphene, h-BN and transition metal dichalcogenides (TMDs) layered materials. The *nep* potential here doesn't support `USE_TABLE` flag to accelerate now.

4.8.1 Potential form

The site potential of *ILP* can be written as:

$$
U_i^{\mathrm{ILP}} = \mathrm{Tap}(r_{ij}) \left[U^{\mathrm{att}}(r_{ij}) + U^{\mathrm{rep}}(r_{ij}, \boldsymbol{n}_i, \boldsymbol{n}_j) \right]
$$

The function $\mathrm{Tap}(r_{ij})$ is a cutoff function and takes the following form in the intermediate region:

$$
\mathrm{Tap}(r_{ij}) = 20 \left(\frac{r_{ij}}{R_{\mathrm{cut}}} \right)^7 - 70 \left(\frac{r_{ij}}{R_{\mathrm{cut}}} \right)^6 + 84 \left(\frac{r_{ij}}{R_{\mathrm{cut}}} \right)^5 - 35 \left(\frac{r_{ij}}{R_{\mathrm{cut}}} \right)^4 + 1
$$

The repulsive term and the attractive term take the following forms:

$$
\begin{aligned}
U^{\mathrm{att}}(r_{ij}) &= -\frac{1}{1 + e^{-d_{ij}\left[r_{ij}/(S_{\mathrm{R},ij} \cdot r_{ij}^{\mathrm{eff}}) - 1\right]}} \frac{C_{6,ij}}{r_{ij}^6}. \\
U^{\mathrm{rep}}(r_{ij}, \boldsymbol{n}_i, \boldsymbol{n}_j) &= e^{\alpha_{ij}\left(1 - \frac{\gamma_{ij}}{\beta_{ij}}\right)} \left\{ \epsilon_{ij} + C_{ij} \left[e^{-\left(\frac{\rho_{ij}}{\gamma_{ij}}\right)^2} + e^{-\left(\frac{\rho_{ji}}{\gamma_{ij}}\right)^2} \right] \right\}.
\end{aligned}
$$

where $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are normal vectors of atom i and j, ρ_{ij} and ρ_{ji} are the lateral interatomic distance, which can be expressed as:

$$\rho_{ij}^2 = r_{ij}^2 - (\boldsymbol{r}_{ij} \cdot \boldsymbol{n}_i)^2$$
$$\rho_{ji}^2 = r_{ij}^2 - (\boldsymbol{r}_{ij} \cdot \boldsymbol{n}_j)^2$$

Other variables are all fitting parameters.

The site potential of *NEP* can be written as:

$$U_i^{\mathrm{NEP}} = \sum_{\mu=1}^{N_{\mathrm{neu}}} w_\mu^{(1)} \tanh\left(\sum_{\nu=1}^{N_{\mathrm{des}}} w_{\mu\nu}^{(0)} q_\nu^i - b_\mu^{(0)}\right) - b^{(1)},$$

More details of *NEP* potential are in *Neuroevolution potential*. Note that in hybrid *NEP* + *ILP* potential, the *NEP* potential will just compute the intralayer interactions.

4.8.2 File format

This hybrid potential requires 3 kinds of files: one for *ILP* potential, one for *NEP* potential and the other for mapping *NEP* potential to groups in model file. We have adopted the *ILP* file format that similar but not identical to that used by lammps. The *NEP* potential file is not required to modify, while to make the *ILP* and *NEP* potentials identify the layers, it's required to set some groups in `model.xyz` file.

In `run.in` file, the `potential` setting is as:

```
potential <ilp file> <nep map file>
```

where `ilp file` and `nep map file` are the filenames of the *ILP* potential file and *NEP* mapping file.

`ilp file` is similar to other empirical potential files in **GPUMD**. But in addition, *ILP* uses different `group_ids` to identify the different layers, so you need to add two `group_methods` in `ilp file`:

```
nep_ilp <number of atom types> <list of elements>
<group_method for layers> <group_method for sublayers>
beta alpha delta epsilon C d sR reff C6 S rcut1 rcut2
...
```

- `nep_ilp` is the name of this hybrid potential.
- `number of atom types` is the number of atom types defined in the `model.xyz`.
- `list of element` is a list of all the elements in the potential (can be in any order).
- `group_method for layers` is the `group_method` set in `model.xyz` to identify different layers. For example, monolayer graphene and monolayer MoS_2 are both single layer so for the atoms in each layer the `group_id` of `group_method for layers` are the same.
- `group_method for sublayers` is used to identify the different sublayers. For example, monolayer graphene contains one sublayer while monolayer MoS_2 contains three sublayers, one Mo sublayer and two S sublayers. For the atoms in each sublayer the `group_id` of `group_method for sublayers` are the same.
- The last line(s) is(are) parameters of *ILP*. `rcut1` is used for calculating the normal vectors and `rcut2` is the cutoff of *ILP*, usually 16Å.

`nep_map_file` can map one or more *NEP* potential files to different layers. The setting is as:

```
<group_method for layers> <number of NEP files> <list of NEP files>
<number of groups>
<NEP_id for group_0>
<NEP_id for group_1>
...
```

- `group_method for layers` is the same as the setting in `ilp file`.
- `number of NEP files` is the number of *NEP* files used in your simulation.
- `list of NEP files` is a list of all the *NEP* filenames. Note that the first file will be identified as `NEP_0` and then `NEP_1` and so on.
- `number of groups` is the number of groups in `group_method for layers`.
- The last `number of groups` lines map the *NEP* to each group. If `NEP_id for group_0` is set to 0, the intralayer interactions between atoms within `group_id` 0 are computed by the first *NEP* file (`NEP_0`) in `list of NEP files`. If set to 1, then computed by the second *NEP* file (`NEP_1`) and so on.

4.8.3 Examples

Example 1: bilayer graphene

Assume your have three files: *ILP* potential file (`C.ilp`), *NEP* potential file (`C.nep`) and *NEP* mapping file (`map.nep`). The potential setting in `run.in` file is as:

```
potential C.ilp map.nep
```

Assume that the first line in `C.nep` is:

```
nep3 1 C
```

and `group_method` 0 is used to identify the different layers. Then `C.ilp` is required to set as:

```
nep_ilp 1 C
0 0
beta_CC alpha_CC delta_CC epsilon_CC C_CC d_CC sR_CC reff_CC C6_CC S_CC rcut1_CC rcut2_CC
```

The first **0** in the second line represents *ILP* potential uses `group_method` 0 to identify different layers. The second **0** represents `group_method` 0 is used to identify the sublayers. For the system with only graphene and h-BN, just set it the same as the previous number.

Then, `map.nep` file required to set as:

```
0 1 C.nep
2
0
0
```

The first **0** in the first line represents *NEP* potential uses `group_method` 0 to identify different layers. The next **1** represents there is just one *NEP* potential file. The number in the second line represents there are two groups in the `group_method` 0. The last two lines represent the `group_0` and `group_1` in `group_method` 0 will use `C.nep` potential file (`NEP_0`).

Example 2: bilayer h-BN / MoS_2

Assume your have four files: *ILP* potential file (`BNMoS.ilp`), *NEP* potential files (`BN.nep`, `MoS.nep`) and *NEP* mapping file (`map.nep`). The potential setting in `run.in` file is as:

```
potential BNMoS.ilp map.nep
```

Assume the first line in `BN.nep` is:

```
nep4 2 B N
```

and in `MoS.nep` is:

```
nep4 2 Mo S
```

We also assume the `group_method` 0 is used to identify the different layers and `group_method` 1 is used to identify the different sublayers for *ILP*. In `group_method` 1, atoms in the sublayers of Mo and S should be set as the different `group_id`. Then `BNMoS.ilp` is required to set as:

```
nep_ilp 4 B N Mo S
0 1
beta_BB alpha_BB delta_BB epsilon_BB C_BB d_BB sR_BB reff_BB C6_BB S_BB rcut1_BB rcut2_BB
beta_BN alpha_BN delta_BN epsilon_BN C_BN d_BN sR_BN reff_BN C6_BN S_BN rcut1_BN rcut2_BN
beta_BMo alpha_BMo delta_BMo epsilon_BMo C_BMo d_BMo sR_BMo reff_BMo C6_BMo S_BMo rcut1_
↪BMo rcut2_BMo
beta_BS alpha_BS delta_BS epsilon_BS C_BS d_BS sR_BS reff_BS C6_BS S_BS rcut1_BS rcut2_BS
...
beta_SS alpha_SS delta_SS epsilon_SS C_SS d_SS sR_SS reff_SS C6_SS S_SS rcut1_SS rcut2_SS
```

Assume `group_id` of MoS_2 is 0 and of h-BN is 1. Then `map.nep` file is set as:

```
0 2 BN.nep MoS.nep
2
1
0
```

The **1** in the third line means `group_0` (MoS_2) uses `MoS.nep` potential file (`NEP_1`) and the last **0** means `group_1` (h-BN) uses `BN.nep` potential file (`NEP_0`).

4.9 Hybrid SW+ILP potential

The hybrid *SW* + *ILP* potential in **GPUMD** combines the Stillinger-Weber potential (*SW*) [Stillinger1985] for intralyer interactions and the interlayer potential (*ILP*) [Ouyang2018] [Ouyang2020] for interlayer interactions to simulate homostructures based on transition metal dichalcogenides layered materials. The *SW* term of this hybrid potential uses the modification version and more details are in [Jiang2015] and [Jiang2019].

4.9.1 Potential form

The site potential of *ILP* can be written as:

$$U_i^{\mathrm{ILP}} = \mathrm{Tap}(r_{ij}) \left[U^{\mathrm{att}}(r_{ij}) + U^{\mathrm{rep}}(r_{ij}, \boldsymbol{n}_i, \boldsymbol{n}_j) \right]$$

The function $\mathrm{Tap}(r_{ij})$ is a cutoff function and takes the following form in the intermediate region:

$$\mathrm{Tap}(r_{ij}) = 20\left(\frac{r_{ij}}{R_{\mathrm{cut}}}\right)^7 - 70\left(\frac{r_{ij}}{R_{\mathrm{cut}}}\right)^6 + 84\left(\frac{r_{ij}}{R_{\mathrm{cut}}}\right)^5 - 35\left(\frac{r_{ij}}{R_{\mathrm{cut}}}\right)^4 + 1$$

The repulsive term and the attractive term take the following forms:

$$U^{\rm att}(r_{ij}) = -\frac{1}{1+e^{-d_{ij}\left[r_{ij}/(S_{{\rm R},ij}\cdot r_{ij}^{\rm eff})-1\right]}}\frac{C_{6,ij}}{r_{ij}^6}.$$
$$U^{\rm rep}(r_{ij},\boldsymbol{n}_i,\boldsymbol{n}_j) = e^{\alpha_{ij}\left(1-\frac{\gamma_{ij}}{\beta_{ij}}\right)}\left\{\epsilon_{ij}+C_{ij}\left[e^{-\left(\frac{\rho_{ij}}{\gamma_{ij}}\right)^2}+e^{-\left(\frac{\rho_{ji}}{\gamma_{ij}}\right)^2}\right]\right\}.$$

where $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are normal vectors of atom i and j, ρ_{ij} and ρ_{ji} are the lateral interatomic distance, which can be expressed as:

$$\rho_{ij}^2 = r_{ij}^2 - (\boldsymbol{r}_{ij}\cdot\boldsymbol{n}_i)^2$$
$$\rho_{ji}^2 = r_{ij}^2 - (\boldsymbol{r}_{ij}\cdot\boldsymbol{n}_j)^2$$

Other variables are all fitting parameters.

The site potential of modified *SW* can be written as:

$$U_i^{\rm SW} = \sum_i\sum_{j>i}\phi_2(r_{ij}) + \sum_i\sum_{j\neq i}\sum_{k>j}\phi_3(r_{ij},r_{ik},\theta_{ijk})$$
$$\phi_2(r_{ij}) = A_{ij}\epsilon_{ij}\left[B_{ij}\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{p_{ij}} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{q_{ij}}\right]\exp\left(\frac{\sigma_{ij}}{r_{ij}-a_{ij}\sigma_{ij}}\right)$$
$$\phi_3(r_{ij},r_{ik},\theta_{ijk}) = \lambda_{ijk}\epsilon_{ijk}\left[f_C(\delta)\delta\right]^2\exp\left(\frac{\gamma_{ij}\sigma_{ij}}{r_{ij}-a_{ij}\sigma_{ij}}\right)\exp\left(\frac{\gamma_{ik}\sigma_{ik}}{r_{ik}-a_{ik}\sigma_{ik}}\right)$$
$$\delta = \cos\theta_{ijk} - \cos\theta_{0ijk}$$

where ϕ_2 and ϕ_3 are two-body and three-body terms. The cutoff of *SW* potential is $a\cdot\sigma$. For some materials, such as borophene and transition metal dichalcogenides, some unnecessary angle types should be excluded in the three-body interaction by multiplying the cutoff function $f_C(\delta)$. Here, for transition metal dichalcogenides, δ_1 is set to 0.25 and δ_2 is set to 0.35.

4.9.2 File format

This hybrid potential requires 2 potential files: *ILP* potential file and *SW* potential file. We have adopted the *ILP* file format that similar but not identical to that used by lammps. To identify the layers, it's required to set two `group_methods` in `model.xyz` file. `group_method` 0 is used to identify the different layers and `group_method` 1 is used to identify different sublayers. One transition metal dichalcogenide layer has three sublayers, i.e., one MoS_2 layer has one Mo sublayer and two S sublayers. For atoms in the same layer, the `group_id` of `group_method` 0 must be the same and for atoms in the same sublayer, the `group_id` of `group_method` 1 must be the same. Now this hybrid potential could be only used to simulate transition metal dichalcogenide homostructures (MX_2), with **M** a transition metal atom (Mo, W, etc.) and **X** a chalcogen atom (S, Se, or Te).

In `run.in` file, the `potential` setting is as:

```
potential <ilp file> <sw file>
```

where `ilp file` and `sw file` are the filenames of the *ILP* potential file and *SW* potential file. `ilp file` is similar to other empirical potential files in **GPUMD**:

```
sw_ilp <number of atom types> <list of elements>
beta alpha delta epsilon C d sR reff C6 S rcut1 rcut2
...
```

- `sw_ilp` is the name of this hybrid potential.
- `number of atom types` is the number of atom types defined in the `model.xyz`. Here, this value must be set to **2** for transition metal dichalcogenide homostructures.

- `list of element` is a list of all the elements in the potential.
- The last line(s) is(are) parameters of *ILP*. `rcut1` is used for calculating the normal vectors and `rcut2` is the cutoff of *ILP*, usually 16Å.

More specifically, for MX_2, the `ilp file` is required to set as:

```
sw_ilp 2 M X
beta_MM alpha_MM delta_MM epsilon_MM C_MM d_MM sR_MM reff_MM C6_MM S_MM rcut1_MM rcut2_MM
beta_MX alpha_MX delta_MX epsilon_MX C_MX d_MX sR_MX reff_MX C6_MX S_MX rcut1_MX rcut2_MX
beta_XM alpha_XM delta_XM epsilon_XM C_XM d_XM sR_XM reff_XM C6_XM S_XM rcut1_XM rcut2_XM
beta_XX alpha_XX delta_XX epsilon_XX C_XX d_XX sR_XX reff_XX C6_XX S_XX rcut1_XX rcut2_XX
```

The `sw file` use the same atomic type list as the `ilp file` and just contains parameters of *SW*. The potential file reads, specifically:

```
A_MM B_MM a_MM sigma_MM gamma_MM
A_MX B_MX a_MX sigma_MX gamma_MX
A_XX B_XX a_XX sigma_XX gamma_XX
lambda_MMM cos0_MMM
lambda_MMX cos0_MMX
lambda_MXM cos0_MXM
lambda_MXX cos0_MXX
lambda_XMM cos0_XMM
lambda_XMX cos0_XMX
lambda_XXM cos0_XXM
lambda_XXX cos0_XXX
```

4.10 Hybrid Tersoff+ILP potential

The hybrid Tersoff + *ILP* potential in **GPUMD** combines the Tersoff (1988) potential [Tersoff1988] for intralyer interactions and the interlayer potential (*ILP*) [Ouyang2018] [Ouyang2020] for interlayer interactions to simulate homo- and heterostructures based on graphene and h-BN layered materials. The Tersoff term of this hybrid potential uses the same implementation as *Tersoff (1988) potential*.

4.10.1 Potential form

The site potential of *ILP* can be written as:

$$U_i^{\mathrm{ILP}} = \mathrm{Tap}(r_{ij}) \left[U^{\mathrm{att}}(r_{ij}) + U^{\mathrm{rep}}(r_{ij}, \boldsymbol{n}_i, \boldsymbol{n}_j) \right]$$

The function $\mathrm{Tap}(r_{ij})$ is a cutoff function and takes the following form in the intermediate region:

$$\mathrm{Tap}(r_{ij}) = 20\left(\frac{r_{ij}}{R_{\mathrm{cut}}}\right)^7 - 70\left(\frac{r_{ij}}{R_{\mathrm{cut}}}\right)^6 + 84\left(\frac{r_{ij}}{R_{\mathrm{cut}}}\right)^5 - 35\left(\frac{r_{ij}}{R_{\mathrm{cut}}}\right)^4 + 1$$

The repulsive term and the attractive term take the following forms:

$$U^{\mathrm{att}}(r_{ij}) = -\frac{1}{1 + e^{-d_{ij}\left[r_{ij}/(S_{\mathrm{R},ij} \cdot r_{ij}^{\mathrm{eff}}) - 1\right]}} \frac{C_{6,ij}}{r_{ij}^6}.$$

$$U^{\mathrm{rep}}(r_{ij}, \boldsymbol{n}_i, \boldsymbol{n}_j) = e^{\alpha_{ij}\left(1 - \frac{\gamma_{ij}}{\beta_{ij}}\right)} \left\{ \epsilon_{ij} + C_{ij} \left[e^{-\left(\frac{\rho_{ij}}{\gamma_{ij}}\right)^2} + e^{-\left(\frac{\rho_{ji}}{\gamma_{ij}}\right)^2} \right] \right\}.$$

where $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are normal vectors of atom i and j, ρ_{ij} and ρ_{ji} are the lateral interatomic distance, which can be expressed as:

$$\rho_{ij}^2 = r_{ij}^2 - (\boldsymbol{r}_{ij} \cdot \boldsymbol{n}_i)^2$$
$$\rho_{ji}^2 = r_{ij}^2 - (\boldsymbol{r}_{ij} \cdot \boldsymbol{n}_j)^2$$

Other variables are all fitting parameters.

The site potential of Tersoff can be written as:

$$U_i^{\text{Tersoff}} = \frac{1}{2} \sum_{j \neq i} f_C(r_{ij}) \left[f_R(r_{ij}) - b_{ij} f_A(r_{ij}) \right].$$

The function f_C is a cutoff function, which is 1 when $r_{ij} < R$ and 0 when $r_{ij} > S$ and takes the following form in the intermediate region:

$$f_C(r_{ij}) = \frac{1}{2} \left[1 + \cos \left(\pi \frac{r_{ij} - R}{S - R} \right) \right].$$

The repulsive function f_R and the attractive function f_A take the following forms:

$$f_R(r) = A e^{-\lambda r_{ij}}$$
$$f_A(r) = B e^{-\mu r_{ij}}.$$

The bond-order is

$$b_{ij} = \left(1 + \beta^n \zeta_{ij}^n \right)^{-\frac{1}{2n}},$$

where

$$\zeta_{ij} = \sum_{k \neq i,j} f_C(r_{ik}) g_{ijk} e^{\alpha (r_{ij} - r_{ik})^m}$$
$$g_{ijk} = \gamma \left(1 + \frac{c^2}{d^2} - \frac{c^2}{d^2 + (h - \cos \theta_{ijk})^2} \right).$$

4.10.2 File format

This hybrid potential requires 2 potential files: *ILP* potential file and Tersoff potential file. We have adopted the *ILP* file format that similar but not identical to that used by lammps. To identify the different layers, it's required to set one `group_methods` in `model.xyz` file. Now this hybrid potential could be only used to simulate homo- and heterostructures of graphene and h-BN.

In `run.in` file, the `potential` setting is as:

```
potential <ilp file> <tersoff file>
```

where `ilp file` and `tersoff file` are the filenames of the *ILP* potential file and Tersoff potential file. `ilp file` is similar to other empirical potential files in **GPUMD**:

```
tersoff_ilp <number of atom types> <list of elements>
<group_method for layers>
beta alpha delta epsilon C d sR reff C6 S rcut1 rcut2
...
```

- `tersoff_ilp` is the name of this hybrid potential.
- `number of atom types` is the number of atom types defined in the `model.xyz`.

- `list of element` is a list of all the elements in the potential.
- `group_method for layers` is the `group_method` set in `model.xyz` to identify different layers. For example, monolayer graphene and monolayer h-BN are both single layer so for the atoms in each layer the `group_id` of `group_method for layers` are the same.
- The last line(s) is(are) parameters of *ILP*. `rcut1` is used for calculating the normal vectors and `rcut2` is the cutoff of *ILP*, usually 16Å.

More specifically, for graphene, if `group_method` 0 is used for different layers, the `ilp file` is required to set as:

```
tersoff_ilp 1 C
0
beta_CC alpha_CC delta_CC epsilon_CC C_CC d_CC sR_CC reff_CC C6_CC S_CC rcut1_CC rcut2_CC
```

The `tersoff file` use the same atomic type list as the `ilp file` and just contains parameters of Tersoff potential. The potential file reads, specifically for graphene:

```
A_CCC B_CCC lambda_CCC mu_CCC beta_CCC n_CCC c_CCC d_CCC h_CCC R_CCC S_CCC m_CCC alpha_
↪CCC gamma_CCC
```

More parameter details are in *Tersoff (1988) potential* and *NEP+ILP potential*.

4.11 Angular Dependent Potential (ADP)

GPUMD supports the Angular Dependent Potential (ADP), which is an extension of the embedded atom method (*EAM*) that includes angular forces through dipole and quadrupole distortions of the local atomic environment.

The ADP was developed to provide a more accurate description of directional bonding and angular forces in metallic systems, particularly for materials where traditional EAM potentials fail to capture the full complexity of atomic interactions. The ADP formalism is especially useful for modeling complex crystal structures, defects, and phase transformations in metals and alloys.

4.11.1 Potential form

General form

The ADP is described in detail by Mishin et al. [Mishin2005] and has been successfully applied to various metallic systems including the Cu-Ta system [Pun2015], U-Mo alloys [Starikov2018], etc. The total energy of atom i is given by:

$$E_i = F_\alpha \left(\sum_{j \neq i} \rho_\beta(r_{ij}) \right) + \frac{1}{2} \sum_{j \neq i} \phi_{\alpha\beta}(r_{ij}) + \frac{1}{2} \sum_s (\mu_{is})^2 + \frac{1}{2} \sum_{s,t} (\lambda_{ist})^2 - \frac{1}{6} \nu_i^2$$

where:

- F_α is the embedding energy as a function of the total electron density at atom i
- $\rho_\beta(r_{ij})$ is the electron density contribution from atom j at distance r_{ij}
- $\phi_{\alpha\beta}(r_{ij})$ is the pair potential interaction between atoms of types α and β
- α and β are the element types of atoms i and j
- s and t are indices running over Cartesian coordinates (x, y, z)
- μ_{is} is the dipole distortion tensor (3 components)
- λ_{ist} is the quadrupole distortion tensor (6 independent components)

- ν_i is the trace of the quadrupole tensor

Angular terms

The dipole distortion tensor represents the first moment of the local environment:

$$\mu_{is} = \sum_{j \neq i} u_{\alpha\beta}(r_{ij}) r_{ij}^s$$

where $u_{\alpha\beta}(r)$ is a tabulated function and r_{ij}^s is the s-component of the vector from atom i to atom j.

The quadrupole distortion tensor represents the second moment of the local environment:

$$\lambda_{ist} = \sum_{j \neq i} w_{\alpha\beta}(r_{ij}) r_{ij}^s r_{ij}^t$$

where $w_{\alpha\beta}(r)$ is another tabulated function. The trace of the quadrupole tensor is:

$$\nu_i = \lambda_{ixx} + \lambda_{iyy} + \lambda_{izz}$$

The angular terms μ and λ introduce directional dependence into the potential energy, allowing the ADP to capture angular forces that are absent in the traditional EAM formalism. These terms are essential for accurately modeling materials with complex bonding environments.

4.11.2 File format

General structure

The ADP potential file follows the extended DYNAMO setfl format, which is compatible with LAMMPS and other molecular dynamics codes. The file structure consists of:

Header section (lines 1-5):

- Lines 1-3: Comment lines (can contain any text, typically author and date information)
- Line 4: `Nelements Element1 Element2 ... ElementN`
  - `Nelements`: Number of elements in the potential
  - `Element1`, `Element2`, etc.: Element symbols (e.g., Cu, Ta, Mo)
- Line 5: `Nrho drho Nr dr cutoff`
  - `Nrho`: Number of points in the embedding function $F(\rho)$ tabulation
  - `drho`: Spacing between tabulated ρ values
  - `Nr`: Number of points in the pair potential and density function tabulations
  - `dr`: Spacing between tabulated r values
  - `cutoff`: Cutoff distance for all functions (in Angstroms)

Per-element sections (repeated `Nelements` times):

Each element section contains:

- Line 1: `atomic_number mass lattice_constant lattice_type`
  - `atomic_number`: Atomic number of the element
  - `mass`: Atomic mass (in amu)
  - `lattice_constant`: Equilibrium lattice constant (in Angstroms)

  - `lattice_type`: Crystal structure (e.g., fcc, bcc, hcp)
- Next `Nrho` values: Embedding function $F(\rho)$
  - Tabulated values of F at $\rho = 0, \Delta\rho, 2\Delta\rho, ..., (N_\rho - 1)\Delta\rho$
  - Units: eV
- Next `Nr` values: Electron density function $\rho(r)$
  - Tabulated values at $r = 0, \Delta r, 2\Delta r, ..., (N_r - 1)\Delta r$
  - Units: electron density

Pair potential section:

For all element pairs (i, j) with $i \geq j$ (upper triangular, since $\phi_{ij} = \phi_{ji}$):

- `Nr` values: Pair potential $\phi_{ij}(r)$
  - Tabulated as $r \times \phi(r)$ (scaled by distance)
  - Units: eV·Angstrom
  - Order: (1,1), (2,1), (2,2), (3,1), (3,2), (3,3), etc.

Dipole function section:

For all element pairs (i, j) with $i \geq j$:

- `Nr` values: Dipole function $u_{ij}(r)$
  - Tabulated as $u(r)$ (NOT scaled by distance)
  - Units: electron density·Angstrom
  - Same ordering as pair potentials

Quadrupole function section:

For all element pairs (i, j) with $i \geq j$:

- `Nr` values: Quadrupole function $w_{ij}(r)$
  - Tabulated as $w(r)$ (NOT scaled by distance)
  - Units: electron density·Angstrom^2
  - Same ordering as pair potentials

4.11.3 Table format

GPUMD supports the standard ADP format as defined in LAMMPS.

Note

The user needs to modify the first line of the potential file as follows:

adp <num_types> <list of elements>

The last two parts can be directly copied from the fourth line of the original potential file. For example:

```
adp 2 Cu Ta
```

Since the first three lines are comments in LAMMPS, this modification does not affect usage in LAMMPS.

4.11.4 Usage

To use an ADP potential in GPUMD, specify it in the `run.in` input file:

```
potential <potential_file>
```

where <potential_file> is the path to the ADP potential file.

Examples

Single-element system (pure Ta):

```
potential Ta.adp
```

Multi-element system (Cu-Ta alloy):

```
potential CuTa_LJ15_2014.adp.txt
```

Multi-element system (U-Mo alloy):

```
potential U_Mo.adp
```

> **Note**
>
> Element types are automatically matched between `model.xyz` and the ADP potential file based on element symbols. The order and number of atom types in `model.xyz` can be different from those in the potential file.

4.11.5 References

CHAPTER

FIVE

GPUMD EXECUTABLE

5.1 Input files

To run one a simulation using the `gpumd` executable, one has to prepare at least two input files that respectively define the *simulation model*, i.e., an atomic configuration (`model.xyz`), and specify the *simulation protocol* (`run.in`).

5.1.1 Simulation protocol (`run.in`)

The `run.in` file is used to define the simulation protocol. The code will execute the commands in this file one by one. If the code encounters an invalid command in this file on start-up, it will report an error message and terminate. In this input file, blank lines and lines starting with `#` are ignored. One can thus write comments after `#`. All other lines should be of the form:

```
keyword parameter_1 parameter_2 ...
```

The overall structure of a `run.in` file is as follows:

- First, set up the potential model using the *potential* keyword. * Multiple NEP potentials may be used using the *dump_observer* keyword. These NEP potentials may either be used to evaluate energy, forces and virials along the trajectory, or averaged together to run the MD on an average PES.
- Then, if needed, use the *minimize* keyword to minimize the energy of the whole system.
- Then one can use the following keywords to carry out static calculations:
  - Use the *compute_cohesive* keyword to compute the cohesive energy curve.
  - Use the *compute_elastic* keyword to compute the elastic constants.
  - Use the *compute_phonon* keyword to compute the phonon dispersions.
- Then, if one wants to carry out *MD* simulations, one has to set up the initial velocities using the *velocity* keyword and carry out a number of *MD* runs as follows:
  - Specify an integrator using the *ensemble* keyword and optionally add keywords to further control the evolution and measurement processes.
  - Use the *run* keyword to run a number of *MD* steps according to the above settings.
  - The last two steps can be repeated.

The following tables provide an overview of the different keywords. A complete list can also be found *here*. The last two columns indicate whether the command is executed immediately (*Exec.*) and whether it is propagted from one `run` command to the next (*Prop.*).

Simulation setup

Keyword	Brief description	Exec.	Prop.
velocity	Set the initial velocities	Yes	N/A
potential	Set up the interaction model	Yes	N/A
dftd3	Add the DFT-D3 dispersion correction to the NEP model	Yes	N/A
change_box	Change the box	Yes	N/A
deform	Deform the simulation box	No	No
ensemble	Specify the integrator for a *MD* run	No	No
fix	Fix (freeze) atoms	No	No
time_step	Specify the integration time step	No	Yes

Actions

Keyword	Brief description	Exec.	Prop.
minimize	Perform an energy minimization	Yes	N/A
run	Run a number of *MD* steps	Yes	No
compute	Compute some time and space-averaged quantities	No	No
compute_cohesive	Compute the cohesive energy curve	Yes	N/A
compute_elastic	Compute the elastic constants	Yes	N/A
compute_dos	Compute the phonon density of states (*PDOS*)	No	No
compute_gkma	Compute the modal heat current using the *GKMA* method	No	No
compute_hac	Compute the thermal conductivity using the *EMD* method	No	No
compute_hnema	Compute the modal thermal conductivity using the *HNEMA* method	No	No
compute_hnemd	Compute the thermal conductivity using the *HNEMD* method	No	No
compute_hnemdec	Compute the multicomponent system thermal conductivity using the *HNEMDEC* method	No	No
compute_phonon	Compute the phonon dispersion	Yes	N/A
compute_sdc	Compute the self-diffusion coefficient (*SDC*)	No	No
compute_msd	Compute the mean-square displacement (*MSD*)	No	No
compute_shc	Compute the spectral heat current (*SHC*)	No	No

Output

Keyword	Brief description	Exec	Prop.
active	Run on-the-fly active learning, saving structures that exceeds a set threshold maximum force uncertainty over all specified NEP potentials.	No	No
dump_exyz	Write positions and other quantities in extended XYZ format	No	No
dump_obse	Write positions and other quantities for each of the observing NEP potentials, or the average of them, in the extended XYZ format.	No	No
dump_force	Write the atomic forces	No	No
dump_posit	Write the atomic positions	No	No
dump_netc	Write the atomic positions in netCDF format	No	No
dump_resta	Write a restart file	No	No
dump_therm	Write thermodynamic quantities	No	No
dump_veloc	Write the atomic velocities	No	No

5.1.2 Simulation model (`model.xyz`)

The `model.xyz` file defines the simulation model, including for example the initial positions, velocities, and boundary conditions. The file needs to be provided in extended XYZ format, which is described here.

File format

Line 1

The first line should have one item only, which is the number of atoms in the model N.

Line 2

This line consists of a number of `keyword=value` pairs separated by spaces. Spaces before and after = *are allowed*. All the characters are case-insensitive. `value` can be a single item or a number of items enclosed by double quotes, such as `keyword="value_1 value_2 value_3"`. Here, the different values are separated by spaces and spaces after the left " and before the right " are allowed. For example, one can write `keyword=" value_1 value_2 value_3 "`.

Essentially any keyword is allowed, but we only read the following ones:

- (*Optional*) `pbc="pbc_a pbc_b pbc_c"`, where `pbc_a`, `pbc_b`, and `pbc_c` can be `T` or `F`, which means box is periodic or non-periodic (free, or open) in the $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ directions. We use the minimum-image convention to account for the periodic boundary conditions for non-*NEP* potentials but will consider sufficiently many periodic images for the *NEP* potentials. Therefore, one needs to make sure that the box size is large enough (the thickness in each direction is larger than twice of the potential cutoff) to incorporate enough neighbors for non-*NEP* potentials but does not need to care about this for NEP potentials. The default is `pbc="T T T"`
- (*Mandatory*) `lattice="ax ay az bx by bz cx cy cz"` specifies the cell vectors:

$$\begin{aligned}\boldsymbol{a} &= a_x\boldsymbol{e}_x + a_y\boldsymbol{e}_y + a_z\boldsymbol{e}_z\\ \boldsymbol{b} &= b_x\boldsymbol{e}_x + b_y\boldsymbol{e}_y + b_z\boldsymbol{e}_z\\ \boldsymbol{c} &= c_x\boldsymbol{e}_x + c_y\boldsymbol{e}_y + c_z\boldsymbol{e}_z\end{aligned}$$

- `properties=property_name:data_type:number_of_columns`

  We only read the following items:

  - `species:S:1` atom type (*Mandatory*)

- `pos:R:3` position vector (*Mandatory*)
- `mass:R:1` mass (*Optional*: default mass values will be used when this is missing)
- `charge:R:1` charge (*Optional*: default charge is zero)
- `vel:R:3` velocity vector (*Optional*)
- `group:I:number_of_grouping_methods` grouping methods (*Optional*)

Line 3 and forward

Each line must contain the same number of items, which are determined by the `property` keyword on line 2. The meaning of the grouping methods is illustrated by the example below.

Units

The mass should be given in units of the unified atomic mass unit (amu). The charge is in units of e (proton charge). The cell dimensions and atom coordinates should be given in units of Ångstrom. Velocities need to be specified in units of Å/fs.

Example

Assume we have the following `model.xyz` file:

```
10
pbc="T F F" lattice="4 0 0 0 1 0 0 0 1" properties=species:S:1:pos:R:3:group:I:3
C  0 0 0 0 0 0
Si 1 0 0 0 1 0
C  2 0 0 0 2 0
Si 3 0 0 0 3 0
C  4 0 0 0 4 0
Si 5 0 0 1 5 0
C  6 0 0 1 6 0
Si 7 0 0 1 7 0
C  8 0 0 1 8 0
Si 9 0 0 1 9 0
```

This means

- There are 10 atoms.
- Use periodic boundary conditions in the x direction and free boundary conditions in the other directions.
- The box lengths in the three directions are respectively 4 Å, 1 Å, and 1 Å.
- There are 5 carbon atoms and 5 silicon atoms. The default masses will be used for the atoms (as there are no specific values given in the input file).
- The 10 atoms are located along a line in the x direction with equal spacing, from 0 Å to 9 Å.
- There is no velocity data in this file.
- There are 3 grouping methods
  - In grouping method 0, atom 0 to atom 4 have group label 0 (which means they are in group 0), and atom 5 to atom 9 have group label 1 (which means they are in group 1).
  - In grouping method 1, atom m ($0 \leq m \leq 9$) has group label m. That is, each group consists of a single atom.
  - In grouping method 2, all the atoms have group label 0. That is, all the atoms are in the same group.

5.1.3 Primitive cell (basis.in)

This file is used to define the unit cell for phonon calculations.

File format

The format of this file must be as follows:

```
N_basis
id(0) mass(0)
id(1) mass(1)
...
id(N_basis-1) mass(N_basis-1)
map(0)
map(1)
...
map(N-1)
```

Here, * `N_basis` is the number of atoms in the unit cell you choose. For example, it can be 2 for diamond silicon if you use the primitive cell as the unit cell. * The next `N_basis` lines contain the atom indices (using the order as in the simulation model file; starting from 0) and masses for the basis atoms. * The remaining `N` lines map the `N` atoms in the simulation model to the basis atoms. If the n-th atom in the simulatoin model file is equivalent to (under translation) the m-th basis atom in the unit cell, we have map(n)=m.

5.1.4 k-points (kpoints.in)

This file is used to specify the k points needed for phonon calculations.

File format

The format of this file must be as follows:

```
N_kpoints
kx(0) ky(0) kz(0)
kx(1) ky(1) kz(1)
...
kx(N_kpoints-1) ky(N_kpoints-1) kz(N_kpoints-1)
```

Here, * `N_kpoints` is the number of k points you want to consider. * The remaining lines give the k vectors (in units of 1/Å) you want to consider. * The user has to make sure that the k vectors are defined in the reciprocal space with respect to the unit cell chosen.

5.1.5 Eigenvectors for modal analysis

5.2 Input parameters

Below you can find a listing of keywords for the `run.in` input file.

5.2.1 replicate

Syntax

This keyword is used as follows:

```
replicate <n_a> <n_b> <n_c>
```

Here, `n_a`, `n_b` and `n_c` are the numbers of replicas in the a, b and c directions. They must be positive integers.

Example

To build a 1*2*4 supercell, just write:

```
replicate 1 2 4
```

Caveats

- Use this command before defining potential.
- The group information and velocities of atoms are also replicated.
- Don't replicate along non-periodic dimensions.

5.2.2 `velocity`

This keyword is used to initialize the velocities of the atoms in the system according to a given temperature.

Syntax

- This keyword can be used in the following ways:

```
velocity <initial_temperature>
velocity <initial_temperature> seed <seed_number>
```

- The temperature is in units of kelvin (K).
- You can specify a seed for the random number generator to generate a fixed temperature distribution, which is optional. If not specified, the velocities are initialized differently each time.

Example

The command:

```
velocity 10
```

means that one wants to set the initial temperature to 10 K.

Caveats

- The initial velocities generated in this fashion do not obey the Maxwell distribution. This is, however, not a problem since during the MD simulation, the Maxwell distribution will be achieved automatically within a short time.
- The total linear and angular momenta are set to zero.
- If there are already velocity data in the *simulation model file*, the initial temperature will not be used and the velocities will be initialized as those in the simulation model file.
- If there are not velocity data in the *simulation model file*, and this keyword is missing, velocities will be initialized with a default temperature of 300 K.

5.2.3 correct_velocity

This keyword allows one to enforce zero linear and angular momenta at regular intervals.

Syntax

- This keyword can have one or two parameters, which are the interval between which the linear and angular momenta are set to zero and an optional group method:

```
correct_velocity <interval> [<group_method>]
```

- `interval` must be larger than or equal to 10. A value between 10 and 100 should be good.
- The `group_method` must be defined in `model.xyz`.

Example 1

The command:

```
correct_velocity 10
```

implies that the linear and angular momenta are set to zero every 10 steps.

Example 2

The command:

```
correct_velocity 50 3
```

implies that the linear and angular momenta for the individual groups in group method 3 are set to zero every 50 steps.

5.2.4 potential

This keyword is used to specify the interatomic potential model for the system.

Available potential models

- *Tersoff-1989 potential*
- *Tersoff-1988 potential*
- *Tersoff mini potential*
- *Embedded atom method (EAM) potential*
- *Force constant potential (FCP)*
- *Lennard-Jones (LJ) potential*
- *Neuroevolution potential (NEP)*
- *Hybrid NEP+ILP potential*
- *Hybrid SW+ILP potential*
- *Hybrid Tersoff+ILP potential*
- *Deep Potential (DP)*

Syntax

This keyword needs one parameter, `potential_filename`, which is the filename (including relative or absolute path) of the potential file to be used.

Example

```
potential Si_NEP.txt
```

5.2.5 compute_extrapolation

This keyword is used to compute the extrapolation grade of structures in an NEP potential.

The extrapolation grade *gamma* can be considered as the uncertainty of a structure relative to the training set.

A structure with large *gamma* tends to have higher energy and force errors.

Similiar methods have been applied to MTP ([Podryabinkin2023]) and ACE ([Lysogorskiy2023]). You can refer to their papers for more details.

Before computing *gamma*, you need to obtain an *active set* from your training set. There are some Python scripts to do it <https://github.com/psn417/nep_active>.

There are also some Python scripts to perform active learning automatically <https://github.com/psn417/nep_maker>.

Syntax

This keyword is used as follows:

```
compute_extrapolation asi_file <asi_file> gamma_low <gamma_low> gamma_high <gamma_high>␣
↪check_interval <check_interval> dump_interval <dump_interval>
```

`asi_file` is the name of the Active Set Inversion (ASI) file. This file is generate by the Python script in <https://github.com/psn417/nep_active>.

`gamma_low`: Only if the max gamma value of a structure exceeds *gamma_low*, then the structure will be dumped into *extrapolation_dump.xyz* file. The default value is *0*.

`gamma_high`: If the max gamma value of a structure exceeds *gamma_high*, then the simulation will stop. The default value is very large so it will never stop.

`check_interval`: Since calculating gamma value is slow, you can check the gamma value every *check_interval* steps. The default value is *1* (check every step).

`dump_interval`: You can set the minimum interval between dumps to *dump_interval* steps. The default value is *1*.

Example

```
compute_extrapolation asi_file active_set.asi gamma_low 5 gamma_high 10 check_interval␣
↪10 dump_interval 10
```

This means that the structures with max gamma between 5-10 will be dumped. The gamma value will be checked every 10 steps.

5.2.6 dftd3

This keyword is used to add the DFT-D3 dispersion correction to NEP. It has no effect when any other potential is used. For theoretical background on DFT-D3, see [Grimme2010] and [Grimme2011].

Syntax

dftd3 <functional> <potential_cutoff> <coordination_number_cutoff>

where `functional` is the exchange-correlation functional used in generating the reference data for training the NEP model, `potential_cutoff` is the cutoff radius (in units of Angstrom) for the D3 potential, and `coordination_number_cutoff` is the cutoff radius (in units of Angstrom) for calculating the coordination numbers.

This keyword can be put anywhere in the `run.in` input file.

We have only implemented the Becke-Johnson (BJ) damping, and a full list of supported functionals can be found from the following link: https://www.chemiebn.uni-bonn.de/pctc/mulliken-center/software/dft-d3/functionalsbj

Example

```
dftd3 pbe 12 6
```

This will add the DFT-D3 dispersion correction to NEP with the PBE functional and a cutoff radius of 12 Angstrom for the D3 potential and a cutoff radius of 6 Angstrom for the coordination number.

Tips

- It usually requires to test the convergence with respect to the cutoff radii, and a balance between accuracy and efficiency must be achieved.
- The user is responsible for not double counting the dispersion correction, i.e., it is not a good idea to add the DFT-D3 correction to a NEP model that has been trained against a dataset containing dispersion correction (no matter what flavor it is).
- Currently cannot use DFT-D3 for the multi-GPU version of NEP.

5.2.7 change_box

This keyword is used to change the simulation box. The box variables and the atom positions are changed according to the following equations:

$$\begin{pmatrix} a_x^{\rm new} & b_x^{\rm new} & c_x^{\rm new} \\ a_y^{\rm new} & b_y^{\rm new} & c_y^{\rm new} \\ a_z^{\rm new} & b_z^{\rm new} & c_z^{\rm new} \end{pmatrix} = \begin{pmatrix} \mu_{xx} & \mu_{xy} & \mu_{xz} \\ \mu_{yx} & \mu_{yy} & \mu_{yz} \\ \mu_{zx} & \mu_{zy} & \mu_{zz} \end{pmatrix} \begin{pmatrix} a_x^{\rm old} & b_x^{\rm old} & c_x^{\rm old} \\ a_y^{\rm old} & b_y^{\rm old} & c_y^{\rm old} \\ a_z^{\rm old} & b_z^{\rm old} & c_z^{\rm old} \end{pmatrix};$$

$$\begin{pmatrix} x_i^{\rm new} \\ y_i^{\rm new} \\ z_i^{\rm new} \end{pmatrix} = \begin{pmatrix} \mu_{xx} & \mu_{xy} & \mu_{xz} \\ \mu_{yx} & \mu_{yy} & \mu_{yz} \\ \mu_{zx} & \mu_{zy} & \mu_{zz} \end{pmatrix} \begin{pmatrix} x_i^{\rm old} \\ y_i^{\rm old} \\ z_i^{\rm old} \end{pmatrix}.$$

The deformation matrix $\mu_{\alpha\beta}$ will be specified by the parameters of this keyword, as we detail below.

Syntax

This keyword accepts 1 or 3 or 6 parameters.

In the case of 1 parameter δ (in units of Ångstrom):

```
change_box <delta>
```

we have

$$\begin{pmatrix} \mu_{xx} & \mu_{xy} & \mu_{xz} \\ \mu_{yx} & \mu_{yy} & \mu_{yz} \\ \mu_{zx} & \mu_{zy} & \mu_{zz} \end{pmatrix} = \begin{pmatrix} \frac{a_x^{\rm old}+\delta}{a_x^{\rm old}} & 0 & 0 \\ 0 & \frac{b_y^{\rm old}+\delta}{b_y^{\rm old}} & 0 \\ 0 & 0 & \frac{c_z^{\rm old}+\delta}{c_z^{\rm old}} \end{pmatrix}$$

In the case of 3 parameters, δ_{xx} (in units of Ångstrom), δ_{yy} (in units of Ångstrom), and δ_{zz} (in units of Ångstrom):

```
change_box <delta_xx> <delta_yy> <delta_zz>
```

we have

$$\begin{pmatrix} \mu_{xx} & \mu_{xy} & \mu_{xz} \\ \mu_{yx} & \mu_{yy} & \mu_{yz} \\ \mu_{zx} & \mu_{zy} & \mu_{zz} \end{pmatrix} = \begin{pmatrix} \frac{a_x^{\rm old}+\delta_{xx}}{a_x^{\rm old}} & 0 & 0 \\ 0 & \frac{b_y^{\rm old}+\delta_{yy}}{b_y^{\rm old}} & 0 \\ 0 & 0 & \frac{c_z^{\rm old}+\delta_{zz}}{c_z^{\rm old}} \end{pmatrix}$$

In the case of 6 parameters (the box type must be triclinic), δ_{xx} (in units of Ångstrom), δ_{yy} (in units of Ångstrom), δ_{zz} (in units of Ångstrom), ϵ_{yz} (dimensionless strain), ϵ_{xz} (dimensionless strain), and ϵ_{xy} (dimensionless strain):

```
change_box <delta_xx> <delta_yy> <delta_zz> <epsilon_yz> <epsilon_xz> <epsilon_xy>
```

we have

$$\begin{pmatrix} \mu_{xx} & \mu_{xy} & \mu_{xz} \\ \mu_{yx} & \mu_{yy} & \mu_{yz} \\ \mu_{zx} & \mu_{zy} & \mu_{zz} \end{pmatrix} = \begin{pmatrix} \frac{a_x^{\rm old}+\delta_{xx}}{a_x^{\rm old}} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{yx} & \frac{b_y^{\rm old}+\delta_{yy}}{b_y^{\rm old}} & \epsilon_{yz} \\ \epsilon_{zx} & \epsilon_{zy} & \frac{c_z^{\rm old}+\delta_{zz}}{c_z^{\rm old}} \end{pmatrix}$$

5.2.8 deform

This keyword is used to deform the simulation box, which can be used to do tensile tests.

Syntax

The `deform` keyword requires 4 parameters:

```
deform <A_per_step> <deform_x> <deform_y> <deform_z>
```

Here, `A_per_step` specifies the speed of the increase of the box length, which is in units of Ångstrom/step. For example, suppose the box length (in a given direction) in the beginning of a run is 100 Ångstrom and this parameter is 10^{-5} Ångstrom/step, then a run with 10^6 steps will change the box length by 10%. This gives a strain rate of 10^8 s $^{-1}$ if the time step is 1 fs. The second parameter `deform_x` can be 0 or 1, where 0 means do not deform the x direction and 1 means deform the x direction. The last two parameters have similar meanings for the y and z directions.

Example

For uniaxial tensile test, one can first equilibrate the system and then deform the box:

```
# equilibration stage
ensemble npt_scr 300 300 100 0 0 0 100 100 100 1000
run 1000000

# production stage
ensemble npt_scr 300 300 100 0 0 0 100 100 100 1000
deform 0.00001 1 0 0
run 1000000
```

Caveats

- Currently, one must use the NPT ensemble when using this keyword. That is, the code assumes that the pressure components in the directions which are not deformed will be controlled. If one does not want to control the pressure (box) in a direction that is not deformed, one can set the elastic constant for that direction to be larger than 2000 GPa.
- In the equilibration stage, it is also recommended to use the NPT ensemble to obtain the zero strain state before applying the deformation.
- One must control the three pressure components independently when using this keyword.

5.2.9 `move`

This keyword is used to move part of the system with a constant velocity.

Syntax

The `move` keyword requires 4 parameters:

```
move <moving_group_id> <velocity_x> <velocity_y> <velocity_z>
```

Here, `moving_group_id` specifies the group id for the moving part, which should be defined in the grouping method 0. The next three parameters specify the moving velocity vector, in units of Ångstrom/fs.

Example

One can first equilibrate the system and then move one group of atoms and at the same time fix another group of atoms:

```
# equilibration stage
ensemble npt_scr 300 300 100 0 0 0 100 100 100 1000
run 1000000

# production stage
ensemble nvt_scr 300 300 100
fix  0            # fix atoms in group 0
move 1 0.001 0 0  # move atoms in group 1, with a speed of 0.001 Ångstrom/fs in the x␣
↪direction
run  1000000
```

Caveats

- One cannot use NPT when using this keyword.
- Currently, the moving group must be defined in the grouping method 0. This might be extended in a future version.

5.2.10 `ensemble` (overview)

This keyword is used to set up an integration method (an integrator). There are different categories of methods accessible via this keyword, which are described on the following pages:

- *"standard" ensembles*
- *MTTK integrators*
- *integrators for thermal conductivity simulations*
- *integrators for path integral molecular dynamics simulations*

- *MSST integrator for simulating compressive shock wave*
- *NPHug integrator for simulating compressive shock wave*
- *NEMD for simulating compressive shock wave*

Units and suggested parameters

The units of temperature and pressure for this keyword are K and GPa, respectively.

The temperature coupling constant <T_coup> means $\tau_T/\Delta t$, where τ_T is the relaxation time of the thermostat and Δt is the time step for integration. We require $\tau_T/\Delta t \geq 1$ and a good choice is $\tau_T/\Delta t \approx 100$.

When $\tau_T/\Delta t > 100000$, the Berendsen thermostat in `npt_ber` will be completely ignored, leaving only the Berendsen barostat. In this case, the NPT ensemble reduces to the NPH ensemble that can be useful for melting point calculations using the two-phase method. We have not yet achieved a similar NPH ensemble using the *stochastic cell rescaling method*.

The pressure coupling constant <p_coup> means $\tau_p/\Delta t$, where τ_p is the relaxation time of the barostat and Δt is the time step for integration. We require $\tau_p/\Delta t \geq 1$ and a good choice is $\tau_p/\Delta t \approx 1000$.

The elastic constants are in units of GPa.

Caveats

One should use one and only one instance of this keyword for each *run keyword*.

5.2.11 `ensemble` (standard)

The `ensemble` keyword is used to set up an integration method (an "integrator"). The integrators described on this page provide conventional methods for controlling temperature and pressure during classical *MD* simulations. Background information pertaining to the different methods referred to on this page can be found *here*.

Syntax

`nve`

If the first parameter is `nve`, it means that the ensemble for the current run is NVE (micro-canonical). There is no need to further specify any other parameters and the full command is simply:

```
ensemble nve
```

`nvt_ber`

If the first parameter is `nvt_ber`, it means that the ensemble for the current run is NVT (canonical) generated by using the *Berendsen method*. In this case, one needs to specify an initial target temperature <T_1>, a final target temperature <T_2>, and a parameter <T_coup>, which reflects the strength of the coupling between the system and the thermostat. The full command is:

```
ensemble nvt_ber <T_1> <T_2> <T_coup>
```

The target temperature (not the instant system temperature) will vary linearly from <T_1> to <T_2> during a run. The choice of <T_coup> is discussed *below*.

nvt_nhc

If the first parameter is `nvt_nhc`, it is similar to the case of `nvt_ber`, but using the *Nose-Hoover chain method*.

nvt_bdp

If the first parameter is `nvt_bdp`, it is similar to the case of `nvt_ber`, but using the *Bussi-Donadio-Parrinello method*.

nvt_lan

If the first parameter is `nvt_lan`, it is similar to the case of `nvt_ber`, but using the *Langevin method* as proposed in [Bussi2007a].

nvt_bao

If the first parameter is `nvt_bao`, it is similar to the case of `nvt_ber`, but using the Langevin method with BAOAB splitting [Leimkuhler2013].

npt_ber

If the first parameter is `npt_ber`, it means that the ensemble for the current run is NPT (isothermal–isobaric) generated by using the *Berendsen barostat*. In this case, apart from the same parameters as in the case of `nvt_ber`, one needs to further specify some target pressure(s), the same number of estimated elastic moduli, and a pressure coupling constant `<p_coup>`. The general format is:

```
ensemble npt_ber <T_1> <T_2> <T_coup> {<pressure_control_parameters>}
```

with three different options for specifying `pressure_control_parameters`:

- *Condition 1*: Cell shape updates are isotropic

  ```
  <p_hydro> <C_hydro> <p_coup>
  ```

  This means you regard your system as isotropic and want to control the three box lengths uniformly according to the hydrostatic pressure `<p_hydro> = (p_xx + p_yy + p_zz)/3`. All directions should have periodic boundary conditions. Currently, we require the box to be orthogonal.

- *Condition 2*: Cell shape updates are orthorhombic

  ```
  <p_xx> <p_yy> <p_zz> <C_xx> <C_yy> <C_zz> <p_coup>
  ```

  In this case, the simulation box must be orthogonal. The three box lengths will be controlled independently according to their respective target pressures. Any direction can be either periodic or nonperiodic and pressure controlling will only be effective in periodic directions.

- *Condition 3*: Cell shape updates are triclinic

  ```
  <p_xx> <p_yy> <p_zz> <p_yz> <p_xz> <p_xy> <C_xx> <C_yy> <C_zz> <C_yz> <C_xz> <C_xy>
  ↪<p_coup>
  ```

  The simulation box must be triclinic and all the directions must be periodic. All cell components will be controlled independently according to the 6 target pressure components.

  Elastic constants in literature may use a different nomenclature. The correspondence is as follows:

  `C_xx=C_xxxx=C_11`, `C_yy=C_yyyy=C_22`, `C_zz=C_zzzz=C_33`,

`C_yz=C_yzyz=C_44`, `C_zx=C_zxzx=C_55`, `C_xy=C_xyxy=C_66`

It is sufficient for the elastic constant tensor `C_ab` to be a (very rough) estimate as long as it is of the right magnitude. It is used to convert the coupling constant (or relaxation time, see *here*) of the barostat into suitable internal units. If a elastic constant component exceeds 2000 GPa, the coupling constant for that component will be zero and that box component will not be changed.

`npt_scr`

If the first parameter is `npt_scr`, it is similar to the case of `npt_ber`, but using the *stochastic cell rescaling method*.

5.2.12 `ensemble` (MTTK)

The variants of the `ensemble` keyword described on this page implement the Nosé-Hoover thermostat [Hoover1996] and the Parrinello-Rahman barostat [Parrinello1981]. Both the thermostat and barostat are enhanced by the *Nosé-Hoover chain method* as recommended in [Martyna1994]. The resulting so-called Martyna-Tuckerman-Tobias-Klein (*MTTK*) integrators enable one to perform simulations in the isothermal-isobaric (NPT) or isenthalpic (NPH) ensembles.

This implementation of the *MTTK* integrator provides more fine-grained control than *"standard" ensembles*. The style of the `npt_mttk` and `nph_mttk` keywords is therefore slightly different.

Syntax

The parameters for running in the isothermal-isobaric ensemble (NPT) can be specified as follows:

```
ensemble npt_mttk temp <T_1> <T_2> <direction> <p_1> <p_2> tperiod <tau_temp> pperiod
↪<tau_press>
```

<T_1> and <T_2> specify the initial and final temperature, respectively. The temperature will vary linearly from <T_1> to <T_2> during the simulation process. The optional <tau_temp> parameter, which defaults to `100`, determines the period of the thermostat in units of the timestep. It determines how strongly the system is coupled to the thermostat.

The <direction> parameter can assume one or more of the following values: `iso`, `aniso`, `tri`, `x`, `y`, `z`, `xy`, `yz`, `xz`. Here, `iso`, `aniso`, and `tri` use hydrostatic pressure as the target pressure. `iso` updates the simulation cell isotropically. `aniso` updates the dimensions of the simulation cell along x, y, and z independently. `tri` updatess all six degrees of freedom of the simulation cell. Using `x`, `y`, `z`, `xy`, `yz`, `xz` allows one to specify each stress component independently.

The parameters <p_1> and <p_2> specify the initial and final pressure, respectively. Finally, the optional parameter <tau_press>, which defaults to `1000`, determines the period of the barostat in units of the timestep. It determines how strongly the system is coupled to the barostat and should be ≥ 200 timesteps.

The `nph_mttk` keyword can be used in analoguous fashion to run simulations in the isenthalpic (NPH) ensemble:

```
ensemble nph_mttk <direction> <p_1> <p_2> <tau_press>
```

Examples

Below follow some examples of how to use these keywords for different ensembles.

NPT Ensemble

```
ensemble npt_mttk temp 300 300 iso 10 10
```

This command sets the target temperature to 300 K and the target pressure to 10 GPa. The cell shape will not change during the simlation but only the volume. These conditions are suitable for simulating liquids. If not constrained, the cell shape may undergo extreme changes since liquids have a vanishing shear modulus (in the long-time limit).

```
ensemble npt_mttk temp 300 1000 iso 100 100
```

This command ramps the temperature from 300 K to 1000 K, while keeping the pressure at 100 GPa.

```
ensemble npt_mttk temp 300 300 aniso 10 10
```

This command replaces `iso` with `aniso`. The three dimensions of the cell thus change independently, but *xy*, *xz* and *yz* remain unchanged.

```
ensemble npt_mttk temp 300 300 tri 10 10
```

All six degrees of freedom of the simulation cell are allowed to change. The simulated system will converge to fully hydrostatic pressure. Note that with `iso` and `aniso`, there is no guarantee that the pressure is hydrostatic, as the system is constrained.

```
ensemble npt_mttk temp 300 300 x 5 5 y 0 0 z 0 0
```

Using these settings one applies a pressure of 5 GPa along the x direction, and 0 GPa along the y and z directions.

```
ensemble npt_mttk temp 300 300 x 5 5
```

Using this setup one applies 5 GPa of pressure along the x direction while fixing the cell dimensions along the other directions.

NPH Ensemble

```
ensemble nph_mttk iso 10 10
```

When using this command one performs a NPH simulation at 10 GPa, allowing only changes in the volume but not the cell shape.

5.2.13 `ensemble` (thermal transport)

The `ensemble` keyword is used to set up an integration method (an integrator). The integrators described on this page provide methods for studying thermal transport via classical *MD* simulations. Background information pertaining to the methods referred to on this page can be found *here*.

Syntax

`heat_nhc`

If the first parameter is `heat_nhc`, it means heating a source region and simultaneously cooling a sink region using local *Nose-Hoover chain thermostats*. The full command is:

```
ensemble heat_nhc <T> <T_coup> <delta_T> <label_source> <label_sink>
```

The target temperatures in the source region with label `<label_source>` and the sink region with label `<label_sink>` are `<T>`+ and `<T>`-, respectively. Therefore, the temperature difference between the two regions is two times `<delta_T>`. In the command above, the parameter `<T_coup>` has the same meaning as in the case of `nvt_nhc`. Both `<label_source>` and `<label_sink>` refer to the 0-th grouping method.

heat_bdp

If the first parameter is heat_bdp, it is similar to the case of heat_nhc, but using the *Bussi-Donadio-Parrinello method*.

heat_lan

If the first parameter is heat_lan, it is similar to the case of heat_nhc, but using the *Langevin method*.

5.2.14 ensemble (PIMD)

The ensemble keyword is used to set up an integration method (an integrator). The integrators described on this page enable one to carry out path integral molecular dynamics (*PIMD*) simulations and thereby to incorporate quantum dynamical effects.

Syntax

pimd

If the first parameter is pimd, it means that the current run will use path-integral molecular dynamics (*PIMD*).

It can be used in the following ways:

```
ensemble pimd <num_beads> <T_1> <T_2> <T_coup>
ensemble pimd <num_beads> <T_1> <T_2> <T_coup> {<pressure_control_parameters>}
```

In both cases, num_beads is the number of beads in the ring polymer, which should be a positive even integer no larger than 128. The first case is similar to the NVT ensemble with nvt_lan as the Langevin thermostat is used for both the internal and the centroid modes [Ceriotti2010]. The second case is similar to the NPT ensemble with npt_ber, where a Berendsen barostat is added compared to the first case. Note that pimd (that is, not rpmd or trpmd described below) must be the first run that requires to set num_beads and one cannot change num_beads from run to run.

rpmd

If the first parameter is rpmd, it means that the current run will use ring-polymer molecular dynamics (*RPMD*) [Craig2004].

It can be used as follows:

```
ensemble rpmd <num_beads>
```

This can be understood as the NVE version of *PIMD*, where no thermostat is applied.

trpmd

If the first parameter is trpmd, it means that the current run will use thermostatted ring-polymer molecular dynamics (*TRPMD*) [Rossi2014].

It can be used as follows:

```
ensemble trpmd <num_beads>
```

This is similar to *RPMD*, but the Langevin thermosat is applied to the internal modes.

5.2.15 `ensemble (TI_Liquid)`

This keyword is used to set up a nonequilibrium thermodynamic integration integrator. Please check [Leite2016], [Leite2019] and [Menon2021] for more details.

Syntax

The parameters can be specified as follows:

```
ensemble ti_liquid temp <temperature> tperiod <tau_temperature> tequil <equilibrium_time>
↪ tswitch <switch_time> press <pressure> sigmasqrd <sigmasqrd-value> p <p-value>
```

- <temperature>: The temperature of the simulation.
- <tau_temperature>: This parameter is optional, and defaults to 100. It determines the period of the thermostat in units of the timestep. It determines how strongly the system is coupled to the thermostat.
- <equilibrium_time>: The number timesteps to equilibrate the system.
- <switch_time>: The number timesteps to vary lambda from 0 to 1.
- <sigmasqrd> and <p>: Specify the TI settings. Implemented values for <p> are 1, 25, 50, 75, 100.
- <pressure>: Although this is an NVT ensemble, you can assign a pressure value to help GPUMD compute the Gibbs free energy. It does not affect the simlation process.

If you do not specify <equilibrium_time> and <switch_time>, they will be automatically set in a 1:4 ratio.

Example

```
ensemble ti_liquid temp 1000 tswitch 4000 tequil 1000 press 0 tperiod 100 sigmasqrd 2 p␣
↪100
```

This command switch lambda for 4000 timesteps, and equilibrate for 1000 timesteps.

Output file

This command will produce a csv file. The columns are lambda, dlambda, potential energy and UF energy (eV/atom).

This command will also produce a yaml file, which contains Gibbs free energy and other information.

Additionally, important information will be displayed on the screen during the simulation process, so it is recommended to carefully review and take note of these details.

5.2.16 `ensemble (TI_Spring)`

This keyword is used to set up a nonequilibrium thermodynamic integration integrator. Please check [Freitas2016] for more details.

Syntax

The parameters can be specified as follows:

```
ensemble ti_spring temp <temperature> tperiod <tau_temperature> tequil <equilibrium_time>
↪ tswitch <switch_time> press <pressure> spring <element_name> <spring_constant>
```

- <temperature>: The temperature of the simulation.
- <tau_temperature>: This parameter is optional, and defaults to 100. It determines the period of the thermostat in units of the timestep. It determines how strongly the system is coupled to the thermostat.

- <equilibrium_time>: The number timesteps to equilibrate the system.
- <switch_time>: The number timesteps to vary lambda from 0 to 1.
- <element_name> and <spring_constant>: Specify the spring constants of elements.
- <pressure>: Although this is an NVT ensemble, you can assign a pressure value to help GPUMD compute the Gibbs free energy. It does not affect the simlation process.

Please note that the spring constants should be placed at the end of the command. If there are no `spring` keyword, the spring constants will be computed automatically through the MSD of atoms. If you do not specify <equilibrium_time> and <switch_time>, they will be automatically set in a 1:4 ratio.

Example

```
ensemble ti_spring temp 300 tequil 1000 tswitch 4000 spring Si 6 O 5
```

This command switch lambda for 4000 timesteps, and equilibrate for 1000 timesteps. The spring constant is 6 eV/A^2 for Si and 5 eV/A^2 for O.

```
ensemble ti_spring temp 300 tequil 1000 tswitch 4000
```

This command calculates spring constants automatically.

Output file

This command will produce a csv file. The columns are lambda, dlambda, potential energy and spring energy (eV/atom).

This command will also produce a yaml file, which contains Gibbs free energy and other information.

Additionally, important information will be displayed on the screen during the simulation process, so it is recommended to carefully review and take note of these details.

5.2.17 `ensemble` (TI_AS)

This keyword is used to set up a nonequilibrium thermodynamic integration integrator with Adiabatic switching (AS) path. It calculates Gibbs free energy along an isothermal path. Please check [Cajahuaringa2022] for more details.

Syntax

The parameters can be specified as follows:

```
ensemble ti_as temp <temperature> tperiod <tau_temperature> <pressure_control> <pmin>
↪<pmax> pperiod <tau_pressure> tswitch <switch_time> tequil <equilibrium_time>
```

- <temperature>: The temperature of the AS simulation.
- <pmin> and <pmax>: The pressure range of the AS simulation.
- <pressure_control>: Please refer to MTTK ensemble for more information.
- <tau_temperature> and <tau_pressure>: These parameters are optional. Please refer to MTTK ensemble for more information.
- <equilibrium_time>: The number timesteps to equilibrate the system.
- <switch_time>: The number timesteps to vary pressure.

If you do not specify <equilibrium_time> and <switch_time>, they will be automatically set in a 1:4 ratio.

Example

```
ensemble ti_as temp 300 aniso 10 30 tswitch 10000
```

This command switch pressure from 10 to 30 GPa in 10000 timesteps.

Output file

This command will produce a csv file. The columns are pressure and volume per atom.

The following code can help you calculate the Gibbs free energy:

```
from pandas import read_csv
import matplotlib.pyplot as plt
import numpy as np
from ase.units import kB, GPa
import yaml
from scipy.integrate import cumtrapz

with open("ti_spring_300.yaml", "r") as f:
    y =  yaml.safe_load(f)

T0 = y["T"]
G0 = y["G"]

ti = read_csv("ti_as.csv")
n = int(len(ti)/2)
forward = ti[:n]
backward = ti[n:][::-1]
backward.reset_index(inplace=True)
p = forward["p"]
V1 = forward["V"]
V2 = backward["V"]

w = (cumtrapz(V1,p,initial=0) + cumtrapz(V2,p,initial=0))*0.5

G = G0 + w
plt.plot(p/GPa, G, label="AS")
plt.legend()
plt.xlabel("P (GPa)")
plt.ylabel("G (eV/atom)")
```

5.2.18 `ensemble` (TI_RS)

This keyword is used to set up a nonequilibrium thermodynamic integration integrator with Reversible Scaling (RS) path. It calculates Gibbs free energy along an isobaric path. Please check [Koning2001] and [Freitas2016] for more details.

Syntax

The parameters can be specified as follows:

```
ensemble ti_rs temp <tmin> <tmax> tperiod <tau_temperature> <pressure_control> <pressure>
↪ pperiod <tau_pressure> tswitch <switch_time> tequil <equilibrium_time>
```

- <tmin> and <tmax>: The temperature range of the RS simulation.
- <pressure_control> and <pressure>: The pressure of RS simualtion. Please refer to MTTK ensemble for more information.
- <tau_temperature> and <tau_pressure>: These parameters are optional. Please refer to MTTK ensemble for more information.
- <equilibrium_time>: The number timesteps to equilibrate the system.
- <switch_time>: The number timesteps to vary lambda.

If you do not specify <equilibrium_time> and <switch_time>, they will be automatically set in a 1:4 ratio.

Example

```
ensemble ti_rs temp 300 3000 aniso 10 tswitch 10000 tequil 1000
```

This command switch lambda for 10000 timesteps.

Output file

This command will produce a csv file. The columns are lambda, dlambda, and enthalpy (eV/atom).

The following code can help you calculate the Gibbs free energy:

```
import yaml
import numpy as np
import matplotlib.pyplot as plt
from pandas import read_csv
from scipy.integrate import cumtrapz
from ase.units import kB

# you need to run a ti_spring simulation at t_min
with open("ti_spring_300.yaml", "r") as f:
    y =  yaml.safe_load(f)

T0 = y["T"]
G0 = y["G"]

rs = read_csv("ti_rs.csv")
n = int(len(rs)/2)
forward = rs[:n]
backward = rs[n:][::-1]
backward.reset_index(inplace=True)
dl = forward["dlambda"]
l = forward["lambda"]
H1 = forward["enthalpy"]
H2 = backward["enthalpy"]
T = T0/l

w = (cumtrapz(H1,l,initial=0) + cumtrapz(H2,l,initial=0))*0.5

G = (G0 + 1.5*kB*T0*np.log(l) + w)/l
plt.plot(T, G, label="RS")
plt.legend()
```

(continues on next page)

(continued from previous page)

```
plt.xlabel("T (K)")
plt.ylabel("G (eV/atom)")
```

5.2.19 ensemble (TI)

This keyword is used to set up a equilibrium thermodynamic integration integrator. It is for testing purpose and its only differece from *ti_spring keyword* is that the lambda value is fixed instead of changing.

Syntax

The parameters can be specified as follows:

```
ensemble ti lambda <lambda> temp <temperature> tperiod <tau_temperature> spring <element_
↪name> <spring_constant>
```

- <temperature>: The temperature of the simulation.
- <tau_temperature>: This parameter is optional, and defaults to 100. It determines the period of the thermostat in units of the timestep. It determines how strongly the system is coupled to the thermostat.
- <element_name> and <spring_constant>: Specify the spring constants of elements.
- <lambda>: The lambda value of the simulation.

Example

```
ensemble ti_spring temp 300 lambda 0.3 spring Si 6 O 5
```

This command uses lambda value 0.3 (30% spring force and 70% original force field). The spring constant is 6 eV/A^2 for Si and 5 eV/A^2 for O.

5.2.20 ensemble (NEMD shock methods)

In a typical NEMD shock simulation, a shock wave is generated by a moving *wall*. The *wall* can be simulated in multiple ways:

- wall_piston: A fixed layer of atoms.
- wall_mirror: A momentum mirror that reflects atoms.
- wall_harmonic: A harmonic potential that pushes atoms away. It is softer than a momentum mirror.

Any of these methods can generate a shock wave. While there are other possible methods, the above methods are usually sufficient.

Please note that the shock wave propagates in the x-direction. Another wall is placed on the opposite side of the cell to prevent atoms from escaping. It is important **NOT** to use non-periodic boundary conditions in the x-direction, as GPUMD currently does not handle it well. A vacuum layer is automatically added on the other side of the cell.

It is recommended to use this with the *dump_shock_nemd keyword* to get the spatial distribution of thermo information.

Syntax

The parameters are specified as follows:

```
ensemble wall_piston vp <vp> thickness <thickness>
```

- <vp>: Indicates the velocity of the moving piston in km/s.

- <thickness>: Defines the thickness of the wall in Angstroms. This keyword is optional with the default value of 20.

```
ensemble wall_mirror vp <vp>
```

- <vp>: Indicates the velocity of the moving piston in km/s.

```
ensemble wall_harmonic vp <vp> k <k>
```

- <vp>: Indicates the velocity of the moving piston in km/s.
- <k>: Defines the strength of the harmonic wall in eV/A^2. This keyword is optional with the default value of 10.

5.2.21 `ensemble` (MSST)

This keyword is used to set up a multi-scale shock technique (*MSST*) integrator. Please check [Reed2003] for more details.

Syntax

The parameters can be specified as follows:

```
ensemble msst <direction> <shock_velocity> qmass <qmass_value> mu <mu_value> tscale
↪<tscale_value> p0 <p0> v0 <v0> e0 <e0>
```

- <direction>: The direction of the shock wave. It can be `x`, `y` or `z`.
- <shock_velocity>: The shock velocity of the shock wave in km/s.
- <qmass_value>: The mass of the simulation cell. It affects the compression speed. Its unit is $\mathrm{amu}^2/4$.
- <mu_value>: The artificial viscosity. It improves convergence. Its unit is $\sqrt{\mathrm{amu} \times \mathrm{eV}}/2$.
- <tscale_value>: The ratio of kinetic energy that turns into cell kinetic energy. This helps speed up the simulation. This keyword is optional, and the default value is `0`.

If keywords <p0>, <v0> or <e0> are not supplied, these quantities will be calculated on the first step. In most cases, you don't need to specify these quantities.

Example

```
ensemble msst x 15 qmass 10000 mu 10
```

This command performs an *MSST* simulation with a shock wave velocity of 15 km/s in the x direction.

5.2.22 `ensemble` (NPHug)

This keyword sets up a Hugoniot thermostat integrator.

This integrator lets you specify a target stress, and adjust temperature to make the system converge to Hugoniot.

In this implementation, we use the barostat and thermostat of *MTTK* integrator, so it is very similiar to *mttk ensemble*.

Please check [Ravelo2004] for more details.

Syntax

The parameters can be specified as follows:

```
ensemble nphug <direction> <p_1> <p_2> tperiod <tau_temp> pperiod <tau_press> p0 <p0> v0
↪<v0> e0 <e0>
```

The <direction> parameter can assume one or more of the following values: `iso`, `aniso`, `tri`, `x`, `y`, `z`. Here, `iso`, `aniso`, and `tri` use hydrostatic pressure as the target pressure. `iso` updates the simulation cell isotropically. `aniso` updates the dimensions of the simulation cell along x, y, and z independently. `tri` updatess all six degrees of freedom of the simulation cell. Using `x`, `y`, `z` allows one to specify each stress component independently.

The parameters <p_1> and <p_2> specify the target pressure, and they should be equal. Finally, the optional parameter <tau_press>, which defaults to `1000`, determines the period of the barostat in units of the timestep. It determines how strongly the system is coupled to the barostat.

If keywords <p0>, <v0> or <e0> are not supplied, these quantities will be calculated on the first step. In most cases, you don't need to specify these quantities.

Example

```
ensemble nphug x 300 300
```

This command performs a simulation using a Hugoniot thermostat with a 300 GPa Hugoniot pressure in the x direction.

5.2.23 `fix`

This keyword can be used to fix (freeze) a group of atoms

Syntax

This keyword requires a single parameter which is the label of the group in which the atoms are to be fixed (velocities and forces are always set to zero such that the atoms in the group do not move). The full command reads:

```
fix <group_label>
```

Here, the `group_label` refers to the grouping method 0 defined in the *simulation model file*.

Caveats

This keyword is not propagating, which means that it only affects the simulation within the run it belongs to.

5.2.24 `time_step`

Set the time step for integration.

Syntax

This keyword can accept either one or two parameters. If there is only one parameter, it is the time step (in units of fs) for a run:

```
time_step <dt_in_fs>
```

If there are two parameters, the first one is the time step and the second one is the maximum distance (in units of Ångstrom) any atom in the system can travel within one step:

```
time_step <dt_in_fs> <max_distance_per_step>
```

Examples

Example 1

To set the time step to 0.5 fs, one can add:

```
time_step 0.5
```

before the *run keyword*.

Example 2

To set the time step to 2.0 fs and to require that no atom in the system can travel more than 0.05 Å per step, one can add:

```
time_step 2.0 0.05
```

before the *run keyword*.

Caveats

- There is a default value (1 fs) for the time step. That means, if you forget to set one, a time step of 1 fs will be used.
- This keyword is propagating. That means, its effect will be passed from one run to the next. Most of the other keywords are non-propagating.

5.2.25 `plumed`

Invoke the PLUMED plugin during a MD run.

Syntax

This keyword has the following format:

```
plumed <plumed_file> <interval> <if_restart>
```

The `plumed_file` parameter is the name of the PLUMED input file. The `interval` parameter is the interval (number of steps) of calling the PLUMED subroutines. The `if_restart` parameter determines if the PLUMED plugin should restart its calculations, which includes appending to its output files, reading the previous biases, etc.

Requirements and specifications

- This keyword requires an external package to operate. Instructions for how to set up the PLUMED package can be found *here*.
- Increasing the `interval` parameter will speed up your simulation if you only want PLUMED to calculate collective variables (CVs). But you have to set it to 1 if your simulation was biased by PLUMED.

Examples

Example 1

To calculate the distance between atoms, one can add:

```
plumed plumed.dat 1000 0
```

before the *run command*. The plumed.dat file, for example, may look like:

```
UNITS LENGTH=A TIME=fs ENERGY=eV
FLUSH STRIDE=1
DISTANCE ATOMS=1,2 LABEL=d1
PRINT FILE=colvar ARG=d1
```

The output file created by PLUMED (the colvar file) will be updated every 1000 steps.

Example 2

To restrain the distance between atoms, one can add:

```
plumed plumed.dat 1 0
```

before the *run command*. The plumed.dat file, for example, may look like:

```
UNITS LENGTH=A TIME=fs ENERGY=eV
FLUSH STRIDE=1
DISTANCE ATOMS=1,2 LABEL=d1
RESTRAINT ARG=d1 AT=0.0 KAPPA=2.0 LABEL=restraint
PRINT FILE=colvar ARG=d1,restraint.bias
```

Caveats

- This keyword is not propagating. That means, its effect will not be passed from one run to the next.

5.2.26 `mc`

The `mc` keyword is used to carry out Monte Carlo (*MC*) trial steps, usually in combination with a *MD* simulation. Three *MC* ensembles are supported, including the canonical, the semi-grand canonical (*SGC*), and the variance-constrained semi-grand canonical (*VCSGC*) [Sadigh2012a] [Sadigh2012b] ensemble.

This keyword can only be used in combination with *NEP* models.

Syntax

`canonical`

If the first parameter is `canonical`, the system will be sampled in the canonical *MC* ensemble. It can be used as follows:

```
mc canonical <md_steps> <mc_trials> <T_i> <T_f> [group <grouping_method> <group_id>]
```

This means that `mc_trials` *MC* trials are performed every `md_steps` *MD* steps, while the instant temperature for the *MC* ensemble changes linearly from `T_i` to `T_f`.

`sgc`

If the first parameter is `sgc`, the system will be sampled in the *SGC MC* ensemble. It can be used as follows:

```
mc sgc <md_steps> <mc_trials> <T_i> <T_f> <num_species> {<species_0> <mu_0> <species_1>
↪<mu_1> ...} [group <grouping_method>  <group_id>]
```

This means that `mc_trials` *MC* trials are performed every `md_steps` *MD* steps, while the instant temperature for the *MC* ensemble changes linearly from `T_i` to `T_f`.

`num_species` specifies the number of species that are to be included in the sampling. It must be no less than 2 and no larger than 4. After specifying the number of species, one needs to specify their chemical symbols (`species_i`) and chemical potentials (`mu_i`) in units of eV. The species can be listed in arbitrary order. Note that only the differences between the chemical potentials matter.

vcsgc

If the first parameter is `vcsgc`, the system will be sampled in the *VCSGC MC* ensemble. It can be used in the following way:

```
mc vcsgc <md_steps> <mc_trials> <T_i> <T_f> <num_species> {<species_0> <phi_0> <species_
↪1> <phi_1> ...} kappa [group <grouping_method>  <group_id>]
```

This means that `mc_trials` *MC* trials are performed every `md_steps` *MD* steps, while the instant temperature for the *MC* ensemble changes linearly from `T_i` to `T_f`.

`num_species` specifies the number of species that are to be included in the sampling. It must be no less than 2 and no larger than 4. After specifying the number of species, one needs to specify their chemical symbols (`species_i`) and chemical potentials (`phi_i` = ϕ_i). The species can be listed in arbitrary order. Next one needs to specify the (dimensionless) `kappa` parameter (κ).

The ϕ and κ parameters constrain the average and variance of the species concentrations, respectively. One can usually achieve a sampling of the full composition range by varying ϕ_i between -1.2 and +1.2, which thus play a role that is equivalent to the μ_i parameters in the *SGC* ensemble. Typically a κ value of 100 is suitable. If the concentration fluctuations are too large (e.g., deep with miscibility gaps) one should increase this value.

The choice of parameters that we use here differs from the original papers [Sadigh2012a] [Sadigh2012b] in terms of normalization and follows the expressions in e.g., [Rahm2021].

General

- The listed species must be supported by the *NEP* model.
- For all the *MC* ensembles, there is an option to specify the grouping method `grouping_method` and the group ID `group_id` in the given grouping method, after the parameter `group`. The functionality is illustrated in the example section below.
- There must be at least one listed species in the initial model system or specified group. For example, if you list Au and Cu for doing *SGC MC*, the system or the specified group must have some Au or Cu atoms (or both); otherwise the *MC* trial cannot get started.

Example 1

An example for sampling in the canonical ensemble is:

```
mc canonical 100 200 500 100 group 1 3
```

This means

- Perform 200 *MC* trials after every 100 *MD* steps.
- Change the temperature for the *MC* simulation linearly from 500 to 100 K.
- Only the atoms in group 3 of grouping method 1 will be considered during *MC* sampling.

Example 2

Here is an example for *MC* sampling the *SGC* ensemble:

```
mc sgc 100 1000 300 300 2 Cu 0 Au 0.6
```

This means

- Perform 1000 *MC* trials after every 100 *MD* steps.
- The temperature for the *MC* ensemble will be kept at 300 K.
- Only the Cu and Au atoms are involved in the *MC* process. The Au atoms have a chemical potential of 0.6 eV relative to the Cu atoms.

Example 3

Here is an example for sampling in the *VCSGC* ensemble:

```
mc vcsgc 200 1000 500 500 2 Al -2 Ag 0 10000
```

This means

- Perform 1000 *MC* trials after every 200 *MD* steps.
- The temperature for the *MC* ensemble will be kept at 500 K.
- Only the Al and Ag atoms are involved in the *MC* process. The dimensionless ϕ parameters for Al and Ag are -2 and 0, respectively. The dimensionless κ parameter is 10000.

5.2.27 electron_stop

This keyword is used to apply electron stopping forces to high-energy atoms during a run. This is usually used in radiation damage simulations.

Syntax

This keyword is used as follows:

```
electron_stop <file>
```

The first line of the file should have 3 values: * the first is the number of data points to be listed N; * the second is the minimum of the energy range $E_{\rm min}$; * the third is the maximum of the energy range $E_{\rm max}$.

The stopping power data listed after this line should have N lines, each corresponding to one energy, which increases linearly from $E_{\rm min}$ to $E_{\rm max}$ with a spacing of $(E_{\rm max} - E_{\rm min})/(N-1)$. For these N lines, the number of columns is the number of species for the potential energy model used. That is, there should be one column with the stopping power for each species. The order of the species should follow that as defined in the potential file.

Example

An example is:

```
electron_stop my_stopping_power.txt
```

5.2.28 add_force

This keyword is used to add force on atoms in a selected group at each step during a run.

Syntax

This keyword is used in one of the following two ways:

```
add_force <group_method> <group_id> <Fx> <Fy> <Fz> # usage 1
add_force <group_method> <group_id> <add_force_file> # usage 2
```

- Force is added to atoms in group `group_id` of group method `group_method`.
- In the first usage, the constant force with components `Fx`, `Fy`, and `Fz` is added on each selected atom.
- In the second usage, a series of forces specified in the file `add_force_file` will be periodically added on each selected atom.
- Force is in units of eV/Å.

Example 1

Add a constant force of 0.1 eV/Å in the x direction on atoms in group 1 of group method 2:

```
add_force 2 1 0.1 0 0
```

Example 2

Add force on atoms in group 2 of group method 0, where the file `add_force.txt` contains a number of rows of 3 values specifying a sequence of force vectors to be periodically applied during the run:

```
add_force 0 2 add_force.txt
```

Note

This keyword can be used multiple times during a run.

5.2.29 add_efield

This keyword is used to add force due to electric field on atoms in a selected group at each step during a run. The force equals to the product of the electric field and the charge of the atom as specified in `model.xyz` via `charge:R:1`.

Syntax

This keyword is used in one of the following two ways:

```
add_efield <group_method> <group_id> <Ex> <Ey> <Ez> # usage 1
add_efield <group_method> <group_id> <add_efield_file> # usage 2
```

- Electric field is applied to atoms in group `group_id` of group method `group_method`.
- In the first usage, the constant electric field with components `Ex`, `Ey`, and `Ez` is applied to each selected atom.
- In the second usage, a series of electric fields specified in the file `add_efield_file` will be periodically applied to each selected atom.
- Electric field is in units of V/Å.

Example 1

Add a constant electric field of 0.1 V/Å in the x direction on atoms in group 1 of group method 2:

```
add_efield 2 1 0.1 0 0
```

Example 2

Add electric field on atoms in group 2 of group method 0, where the file `add_efield.txt` contains a number of rows of 3 values specifying a sequence of electric field vectors to be periodically applied during the run:

```
add_efield 0 2 add_efield.txt
```

Note

This keyword can be used multiple times during a run.

5.2.30 `minimize`

This keyword is used to minimize the energy of the system. Currently, the fast inertial relaxation engine (FIRE) [Bitzek2006] [Guénolé2020] method and the steepest descent (SD) method have been implemented.

Syntax

This keyword is used as follows:

```
minimize <method> <force_tolerance> <maximal_number_of_steps> <box_change> <hydrostatic_
↪strain>
```

Here, `method` can be `sd` (the steepest descent method) or `fire` (the FIRE method). `force_tolerance` is in units of eV/Å. When the largest absolute force component among the $3N$ force components in the system is smaller than `force_tolerance`, the energy minimization process will stop even though the number of steps (interations) performed is smaller than `maximal_number_of_steps`. `maximal_number_of_steps` is the maximal number of steps (interations) to be performed for the energy minimization process. `box_change` specifies whether to optimize the box during minimization. It only supports the FIRE method now. `hydrostatic_strain` indicates whether hydrostatic strain should be applied to change the box.

Examples

Example 1

The command:

```
minimize sd 1.0e-6 10000
```

means that one wants to do an energy minimization using the steepest descent method, with a force tolerance of 10^{-6} eV/Å for up to 10,000 steps.

Example 2

If you have no idea how small `force_tolerance` should be, you can simply asign a negative number to it:

```
minimize sd -1 10000
```

In this case, the energy minimization process will definitely run 10,000 steps.

Example 3

The command:

```
minimize fire 1.0e-5 1000
```

means that one wants to do an energy minimization using the FIRE method, with a force tolerance of 10^{-5} eV/Å for up to 1,000 steps.

Example 4

The command:

```
minimize fire 1.0e-5 1000 1
```

means that one wants to optimize box during the energy minimization process.

The command:

```
minimize fire 1.0e-5 1000 1 1
```

means that one wants to optimize box using hydrostatic strain during the energy minimization process.

Example 5

If you want to ouput the optimized structure, use *dump_xyz* in the *nve* ensemble like the following command:

```
minimize fire 1.0e-5 1000 1

ensemble    nve
time_step   0
dump_xyz    -1 0 1 relaxed.xyz
run         1
```

Caveats

- This keyword should occur after the *potential keyword*.

5.2.31 run

Run a number of *MD* steps according to the settings specified for the current run.

Syntax

This keyword only requires a single parameter, which is the number of steps to be run:

```
run <number_of_steps>
```

Example

To run one million steps, just write:

```
run 1000000
```

Caveats

- The number of steps should be compatible with some parameters in some other keywords.
- We can regard a run as a block of commands from the *ensemble keyword* to `run`:

```
# the first run
ensemble ...
# other commands
run ...

# the second run
ensemble ...
# other commands
run ...

# the third run
ensemble ...
# other commands
run ...
```

5.2.32 `compute`

This keyword is used to compute and output space and time averaged quantities. The results are written to the *compute.out output file*.

Syntax

It is used in the following way:

```
compute <grouping_method> <sample_interval> <output_interval> {<quantity>}
```

The first parameter `grouping_method` refers to the grouping method defined in the *simulation model file*. This parameter should be an integer and a number m means the m-th grouping method (we count from 0) in the *simulation model file*.

The second parameter `sample_interval` means sampling the quantities every so many time steps.

The third parameter `output_interval` means averaging over so many sampled data before giving one output.

Starting from the fourth parameter, one can list the quantities to be computed. The allowed names for `quantity` are:

- `temperature`, which yields the temperature
- `potential`, which yields the potential energy
- `force`, which yields the force vector
- `virial`, which yields all virial components
- `jp`, which yields the potential part of the heat current vector
- `jk`, which yields the kinetic part of the heat current vector
- `momentum`, which yields the momentum

One can write one or more (distinct) names in any order.

Example

For example:

```
compute 0 100 10 temperature
```

means using the 0-th grouping method defined in the *simulation model file*, sampling `temperature` every 100 time steps and averaging over 10 data points before writing to file. That is, there is only one output every $100 \times 10 = 1000$ time steps.

5.2.33 `compute_adf`

This keyword computes the angular distribution function (*ADF*) for all atoms or specific triples of species. Each *ADF* is represented as a histogram, created by measuring the angles formed between a central atom and two neighboring atoms, and binning these angles into *num_bins* bins. Only neighbors with distances $rc\_min < R < rc\_max$ are considered, where *rc_min* and *rc_max* are specified separately for the first and second neighbor atoms in each *ADF* calculation. Currently, this feature is only available for classical *MD*. The results are written to the *adf.out* file.

Syntax

For global *ADF*, the keyword is used as follows:

```
compute_adf <interval> <num_bins> <rc_min> <rc_max>
```

This means the *ADF* calculations will be performed every `interval` steps, with `num_bins` data points.

For local *ADF*, the keyword is used as follows:

```
compute_adf <interval> <num_bins> <itype1> <jtype1> <ktype1> <rc_min_j1> <rc_max_j1> <rc_
↪min_k1> <rc_max_k1> ...
```

This means the *ADF* calculations will be performed every `interval` steps, with `num_bins` data points. The angle formed by the central atom I and neighboring atoms J and K is included in the *ADF* if the following conditions are met:

- The distance between atoms I and J is between *rc_min_jN* and *rc_max_jN*.
- The distance between atoms I and K is between *rc_min_kN* and *rc_max_kN*.
- The type of atom I matches *itypeN*.
- The type of atom J matches *jtypeN*.
- The type of atom K matches *ktypeN*.
- Atoms I, J, and K are distinct.

The *ADF* value for a bin is computed by dividing the histogram count by the total number of triples that satisfy the criteria, ensuring that the integral of the *ADF* with respect to the angle is 1. In other words, the *ADF* is a probability density function.

Example

- compute_adf 100 30 0.0 1.2 # Total *ADF* every 100 MD steps with 30 data points for bond lengths between 0.0 and 1.2
- compute_adf 500 50 0 1 1 0.0 1.2 0.0 1.3 # Calculate 0-1-1 *ADF* every 500 MD steps with 50 data points for I-J bond lengths between 0.0 and 1.2, and I-K bond lengths between 0.0 and 1.3
- compute_adf 500 50 0 1 1 0.0 1.2 0.0 1.3 1 0 1 0.0 1.2 0.0 1.3 # Calculate 0-1-1 and 1-0-1 *ADF* every 500 MD steps with 50 data points

5.2.34 compute_cohesive

This keyword is used for computing the cohesive energy curve. The results are written to the *cohesive.out output file*.

Syntax

This keyword is used as follows:

```
compute_cohesive <e1> <e2> <num_points>
```

Here, `e1` is the smaller box-scaling factor, `e2` is the larger box-scaling factor, and `num_points` is the number of points sampled uniformly from `e1` to `e2`.

Examples

The command:

```
compute_cohesive 0.9 1.2 301
```

means that one wants to compute the cohesive energy curve from the box-scaling factor 0.9 to the box-scaling factor 1.2, with 301 points. The box-scaling points will be 0.9, 0.901, 0.902, ..., 1.2.

Caveats

This keyword must occur after the *potential keyword*.

5.2.35 compute_dos

This keyword computes the phonon density of states (*PDOS*) using the mass-weighted velocity autocorrelation (*VAC*) function. The output is normalized such that the integral of the *PDOS* over all frequencies equals $3N$, where N is the number of atoms. If this keyword appears in a run, the mass-weighted *VAC* function will be computed and directly used to compute the *PDOS*.

The results of these calculations will be written to *mvac.out* (for the mass-normalized *VAC* function) and *dos.out* (for the *DOS*).

Syntax

For this keyword, the command looks like:

```
compute_dos <sample_interval> <Nc> <omega_max> [{<optional_arg>}]
```

with parameters defined as:

- `sample_interval`: Sampling interval of the velocity data
- `Nc`: Maximum number of correlation steps
- `omega_max`: Maximum angular frequency $\omega_{max} = 2\pi\nu_{max}$ used in the *PDOS* calculation

The optional arguments (`optional_arg`) provide additional functionality by allowing special keywords. The keywords for this function are `group` and `num_dos_points`. These keywords can be used in any order but the parameters associated with each must follow directly.

The option `group` has two parameters:

```
group <group_method> <group>
```

where `group_method` is the grouping method to use for computation and `group` is the index of the group to use. If `group` is -1, it means to calculate the DOS for every group in the `group_method`. If each atom is in one group, one can get the per-atom DOS and then calculate the phonon participation ratio.

The option `num_dos_points` has one parameter:

```
num_dos_points <points>
```

where `points` is the number of frequency points to be used in the DOS calculation. It defaults to `Nc` if the `num_dos_points` option is not specified.

Example

An example for the use of this keyword is:

```
compute_dos 5 200 400.0 group 1 2 num_dos_points 300
```

This means that you

- want to calculate the *PDOS*
- the velocity data will be recorded every 5 steps
- the maximum number of correlation steps is 200
- the maximum angular frequency you want to consider is $\omega_{max} = 2\pi\nu_{max} = 400$ THz
- you would like to compute only over group 2 in group method 1
- you would like the maximum angular frequency to be evenly divided into 300 points for output.

Caveats

This keyword cannot be used in the same run as the *compute_sdc keyword*.

5.2.36 `compute_elastic`

This keyword is used to compute the elastic constants. The results are written to the file `elastic.out`.

Syntax

This keyword is used as follows:

```
compute_elastic <strain_value> <symmetry_type>
```

`strain_value` is the amount of strain to be applied in the calculations.

`symmetry_type` is the symmetry type of the material considered. Currently, it can only be `cubic`.

Example

For example, the command:

```
compute_elastic 0.01 cubic
```

means that one wants to compute the elastic constants for a cubic system with a strain of 0.01.

Caveats

This keyword must occur after the *potential keyword*.

5.2.37 compute_gkma

The `compute_gkma` keyword can be used to calculate the modal heat current using the Green-Kubo modal analysis (*GKMA*) method [Lv2016]. The results are written to the *heatmode.out output file*.

Syntax

```
compute_gkma <sample_interval> <first_mode> <last_mode> <bin_option> <size>
```

`sample_interval` is the sampling interval (in number of steps) used to compute the heat modal heat current.

`first_mode` and `last_mode` are the first and last mode, respectively, in the *eigenvector.in input file* to include in the calculation.

`bin_option` determines which binning technique to use. The options are `bin_size` and `f_bin_size`.

`size` defines how the modes are added to each bin. If `bin_option` is `bin_size`, then this is an integer describing how many modes are included per bin. If `bin_option` = `f_bin_size`, then binning is by frequency and this is a float describing the bin size in THz.

Examples

Example 1

```
compute_gkma 10 1 27216 f_bin_size 1.0
```

This means that

- you want to calculate the modal heat current with the *GKMA* method
- the modal heat flux will be sampled every 10 steps
- the range of modes you want to include of calculations are from 1 to 27216
- you want to bin the modes by frequency with a bin size of 1 THz

Example 2

```
compute_gkma 10 1 27216 bin_size 1
```

This example is identical to Example 1, except the modes are binned by count. Here, each bin only has one mode (i.e., all modes are included in the output).

Example 3

```
compute_gkma 10 1 27216 bin_size 10
```

This example is identical to Example 2, except each bin has 10 modes.

Caveats

This computation can be very memory intensive. The memory requirements are comparable to the size of the *eigenvector.in input file*.

Depending on the number of steps to run, sampling interval, and number of bins, the *heatmode.out output file* can become very large as well (i.e., many GBs).

This keyword cannot be used in the same run as the *compute_hnema keyword*. The keyword that appears last will be used in the run.

5.2.38 compute_hac

This keyword can be used to calculate the heat current autocorrelation (*HAC*) and running thermal conductivity (*RTC*) using the *Green-Kubo method*. The results will be written to the *hac.out output file*.

Syntax

This keyword has 3 parameters:

```
compute_hac <sampling_interval> <correlation_steps> <output_interval>
```

The first parameter is the sampling interval for the heat current data. The second parameter is the maximum correlations steps. The third parameter for is the output interval of the *HAC* and *RTC* data.

Examples

Example 1

```
time_step 1
compute_hac 10 100000 1
run 10000000
```

This means that

- You want to calculate the thermal conductivity using the *Green-Kubo method* (the *EMD* method) in this run, which contains 10 milillion steps with a time step of 1 fs.
- The heat current data will be recorded every 10 steps. Therefore, there will be 1 million heat current data in each direction.
- The maximum number of correlation steps is 10^5, which is one tenth of the number of heat current data. This is a very sound choice. The maximum correlation time will be $10^5 \times 10 = 10^6$ time steps, i.e., 1 ns.
- The *HAC*/*RTC* data will not be averaged before outputting, generating 10^5 rows of data in the output file.

Example 2

```
compute_hac 10 100000 10
```

This is similar to the above example but with one expection: The *HAC*/*RTC* data will be averaged for every 10 data before outputing, generating 10^4 rows of data in the output file.

5.2.39 compute_hnema

The `compute_hnema` keyword is used to calculate the modal thermal conductivity using the *homogeneous nonequilibrium modal analysis* (*HNEMA*) method [Gabourie2021]. The results are written to the *kappamode.out output file*.

Syntax

```
compute_hnema <sample_interval> <output_interval> <Fe_x> <Fe_y> <Fe_z> <first_mode>
↪<last_mode> <bin_option> <size>
```

`sample_interval` is the sampling interval (in number of steps) used to compute the heat modal heat current. Must be a divisor of `output_interval`.

`output_interval` is the interval to output the modal thermal conductivity. Each modal thermal conductivity output is averaged over all samples per output interval.

`Fe_x` is the x direction component of the external driving force F_e in units of Å^{-1}.

`Fe_y` is the y direction component of the external driving force F_e in units of Å^{-1}.

`Fe_z` is the z direction component of the external driving force F_e in units of Å^{-1}.

`first_mode` and `last_mode` are the first and last mode, respectively, in the *eigenvector.in input file* to include in the calculation.

`bin_option` determines which binning technique to use. The options are `bin_size` and `f_bin_size`.

`size` defines how the modes are added to each bin. If `bin_option` is `bin_size`, then this is an integer describing how many modes are included per bin. If `bin_option` is `f_bin_size`, then binning is by frequency and this is a float describing the bin size in THz.

Examples

Example 1

```
compute_hnema 10 1000 0.000008 0 0 1 27216 f_bin_size 1.0
```

This means that

- you want to calculate the modal thermal conductivity with the *HNEMA* method
- the modal thermal conductivity will be sampled every 10 steps
- the average of all sampled modal thermal conductivities will be output every 1000 time steps
- the external driving force is along the x direction and has a magnitude of 0.8×10^{-5} Å^{-1}
- the range of modes you want to include of calculations are from 1 to 27216
- you want to bin the modes by frequency with a bin size of 1 THz.

Example 2

```
compute_hnema 10 1000 0.000008 0 0 1 27216 bin_size 1
```

This example is identical to Example 1, except the modes are binned by count. Here, each bin only has one mode (i.e. all modes are included in the output).

Example 3

```
compute_hnema 10 1000 0.000008 0 0 1 27216 bin_size 10
```

This example is identical to Example 2, except each bin has 10 modes.

Caveats

This computation can be very memory intensive. The memory requirements are comparable to the size of the *eigenvector.in input file*.

This keyword cannot be used in the same run as the *compute_gkma keyword*. The keyword used last will be used in the run.

5.2.40 compute_hnemd

This keyword is used to calculate the thermal conductivity using the *homogeneous non-equilibrium molecular dynamics* (*HNEMD*) method [Fan2019]. The results are written to the *kappa.out output file*.

Syntax

```
compute_hnemd <output_interval> <Fe_x> <Fe_y> <Fe_z>
```

The first parameter is the output interval.

The next three parameters are the x, y, and z components of the external driving force $\boldsymbol{F}_e$ in units of Å^{-1}.

Usually, there should be only one nonzero component of $\boldsymbol{F}_e$. According to Eq. (8) of [Fan2019]:

- Using a nonzero x component of $\boldsymbol{F}_e$, one can obtain the xx, yx and zx components of the thermal conductivity tensor.
- Using a nonzero y component of $\boldsymbol{F}_e$, one can obtain the xy, yy and zy components of the thermal conductivity tensor.
- Using a nonzero z component of $\boldsymbol{F}_e$, one can obtain the xz, yz and zz components of the thermal conductivity tensor.

Examples

Example 1

```
compute_hnemd 1000 0.00001 0 0
```

This means that

- you want to calculate the thermal conductivity using the *HNEMD* method;
- the thermal conductivity will be averaged and output every 1000 steps (the heat current is sampled for every step);
- the external driving force is along the x direction and has a magnitude of 10^{-5} Å^{-1}.

Note that one should control the temperature when using this keyword. Otherwise, the system will be heated up by the external driving force.

Important: For this purpose, the *Nose-Hoover chain thermostat* is recommended. The *Langevin thermostat* cannot be used for this purpose because it will affect the dynamics of the system.

Example 2

```
compute_hnemd 1000 0 0.00001 0
```

This is similar to the above example, but the external driving force is applied along the y direction.

5.2.41 compute_hnemdec

This keyword is used to calculate the multicomponent system thermal conductivity using the *homogeneous non-equilibrium molecular dynamics Evans-Cummings algorithm* (*HNEMDEC*) method. The results are written to the *onsager.out output file*.

Syntax

```
compute_hnemdec <driving_force> <output_interval> <Fe_x> <Fe_y> <Fe_z>
```

`driving_force` determines which type of driving force to use. It could be zero or non zero positive integer, which means thermal driving force or diffusive driving force. In the case of zero, heat flux is equilvalent to dissipative flux, with driving force F_e in the units of Å^{-1}. For a non zero positive integer such as i, a momentum flux of the ith element in the order of the first line of nep.txt is produced as dissipative flux, with driving force F_e in the units of eV/Å.

`output_interval` is the interval to output the onsager coefficients.

`Fe_x` is the x direction component of the external driving force F_e.

`Fe_y` is the y direction component of the external driving force F_e.

`Fe_z` is the z direction component of the external driving force F_e.

Examples

Example 1

```
compute_hnemdec 0 1000 0.00001 0 0
```

This means that

- you want to calculate the onsager coefficients using the *HNEMDEC* method with heat flux as dissipative flux;
- the onsager coefficients will be averaged and output every 1000 steps (the heat current and momentum current is sampled for every step);
- the external driving force is along the x direction and has a magnitude of 10^{-5} Å^{-1}.

Note that one should control the temperature when using this keyword. Otherwise, the system will be heated up by the external driving force.

Important: For this purpose, the *Nose-Hoover chain thermostat* is recommended. The *Langevin thermostat* cannot be used for this purpose because it will affect the dynamics of the system.

Example 2

```
compute_hnemdec 2 1000 0 0.00001 0
```

The external driving force is diffusive driving force that will produced a momentum flux of the second element as dissipative flux and has a magnitude of 10^{-5} eV/Å. The force is applied along the y direction.

5.2.42 compute_orientorder

This keyword computes the local, rotationally invariant Steinhardt order parameters q_l and w_l, as described in [Steinhardt1983] and [Mickel2013]. The results are written to the *orientorder.out* file.

For an atom i, the parameter $q_l(i)$ is defined as:

$$q_l(i) = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} |q_{lm}(i)|^2}$$

where the quantity $q_{lm}(i)$ is obtained by summing the spherical harmonics over atom i and its neighbors j:

$$q_{lm}(i) = \frac{1}{N_b} \sum_{j=1}^{N_b} Y_{lm}(\theta(\boldsymbol{r}_{ij}), \phi(\boldsymbol{r}_{ij})).$$

If `wl` is set to `True`, the quantity w_l is computed. This is a weighted average of the $q_{lm}(i)$ values using Wigner 3-j symbols, yielding a **rotationally invariant combination**:

$$w_l(i) = \sum_{m_1+m_2+m_3=0} \begin{pmatrix} l & l & l \\ m_1 & m_2 & m_3 \end{pmatrix} q_{lm_1}(i) q_{lm_2}(i) q_{lm_3}(i).$$

If the `wl_hat` parameter is `True`, the w_l order parameter is normalized as:

$$w_l(i) = \frac{\sum_{m_1+m_2+m_3=0} \begin{pmatrix} l & l & l \\ m_1 & m_2 & m_3 \end{pmatrix} q_{lm_1}(i) q_{lm_2}(i) q_{lm_3}(i)}{\left(\sum_{m=-l}^{l} |q_{lm}(i)|^2 \right)^{3/2}}.$$

If `average` is `True`, the averaging procedure replaces $q_{lm}(i)$ with $\overline{q}_{lm}(i)$, which is the mean value of $q_{lm}(k)$ over all neighbors k of particle i, including atom i itself:

$$\overline{q}_{lm}(i) = \frac{1}{N_b} \sum_{k=0}^{N_b} q_{lm}(k).$$

Syntax

```
compute_orientorder <interval> <mode_type> <mode_parameters> <ndegrees> <degree1>
↪<degree2> ... <average> <wl> <wlhat>
```

- **interval**: perform the calculation every `interval` steps.
- **mode_type**: the neighbor selection mode. Currently supports `cutoff` or `nnn` (nearest neighbor number).
- **mode_parameters**: - for `cutoff` mode, this is the cutoff distance; - for `nnn` mode, this is the number of neighbors. **Note:** if an atom has fewer neighbors than `nnn` within 6 Å, the result is set to 0.
- **ndegrees**: number of spherical harmonic degrees (l) to compute.
- **degree1..degreeN**: individual degree values.
- **average**: whether to calculate the averaged Steinhardt order parameter. `1` = True, `0` = False. Defaults to False.
- **wl**: whether to compute the w_l version of the Steinhardt order parameter. Defaults to False.
- **wlhat**: whether to compute the **normalized** w_l version. Defaults to False.

Examples

- `compute_orientorder 100 cutoff 4.0 3 4 6 8` → Every 100 MD steps, compute three degrees (4, 6, and 8) using a 4 Å cutoff.

- `compute_orientorder 50 nnn 12 2 4 6 0 1` → Every 50 MD steps, compute two degrees (4 and 6) with 12 nearest neighbors. Additionally, compute the w_l version.
- `compute_orientorder 100 nnn 12 2 4 6 1 1 1` → Every 100 MD steps, compute two degrees (4 and 6) with 12 nearest neighbors. Use the **averaged definition**, and calculate both the original and the normalized w_l versions.

5.2.43 compute_phonon

This keyword can be used to compute the phonon dispersions using the finite-displacement method. The results are written to the *D.out output file* and the *omega2.out output file*.

To use this keyword, please make sure the following files have been preprared in the input directory:

Input filename	Brief description
`basis.in`	Define the mapping from the atom label to the basis label
`kpoints.in`	Specify the $\boldsymbol{k}$-points

Syntax

This keyword is used as follows:

```
compute_phonon <cutoff> <displacement>
```

`cutoff` is the cutoff distance (in units of Å) for calculating the force constants.

`displacement` is the displacement (in units of Å) for calculating the force constants using the finite-displacement method.

Example

For example, the command:

```
compute_phonon 5.0 0.01
```

means that one wants to compute the phonon dispersion using a cutoff distance of 5 Å and a displacement of 0.01 Å.

Caveats

This keyword should occur after all the `potential` keywords.

For two-body potentials, the cutoff distance for force constants can be the same as that for force evaluations. However, for many-body potentials, the cutoff distance for force constants usually needs to be twice of the potential cutoff distance. Also make sure that the box size in any direction is at least twice of this force constant cutoff distance.

Related tutorial

The use of the `calculate_phonon` keyword is illustrated in this tutorial.

5.2.44 compute_sdc

This keyword computes the self-diffusion coefficient (*SDC*) from the velocity autocorrelation (*VAC*) function. If this keyword appears in a run, the *VAC* function will be computed and integrated to obtain the *SDC*. The results will be written to *sdc.out output file*.

Syntax

For this keyword, the command looks like:

```
compute_sdc <sample_interval> <Nc> [<optional_arg>]
```

with parameters defined as

- `sample_interval`: Sampling interval of the velocity data
- `Nc`: Maximum number of correlation steps

The optional argument `optional_arg` allows an additional special keyword. The keyword for this function is `group`. The parameters are:

- `group`, where `group_method` is the grouping method to use for computation and `group` is the group in the grouping method to use

Examples

An example of this function is:

```
compute_sdc 5 200 group 1 1
```

This means that you

- want to calculate the *SDC*
- the velocity data will be recorded every 5 steps
- the maximum number of correlation steps is 200
- you would like to compute only over group 1 in group method 1.

Caveats

This function cannot be used in the same run with the *compute_dos keyword*.

5.2.45 compute_msd

This keyword computes the self-diffusion coefficient (*SDC*) from the mean-square displacement (*MSD*) function. If this keyword appears in a run, the *MSD* function will be computed and the *SDC* is also calculated as a time derivative of it. The results will be written to *msd.out output file*.

Syntax

For this keyword, the command looks like:

```
compute_msd <sample_interval> <Nc> [<optional_arg>]
```

with parameters defined as

- `sample_interval`: Sampling interval of the position data
- `Nc`: Maximum number of correlation steps

The optional argument `optional_arg` allows three additional special keyword. The first special keyword is `group`. The parameters are:

- `group`, where `group_method` is the grouping method to use for computation and `group` is the group in the grouping method to use

The second special keyword is `all_groups`. This keyword computes the *MSD* and *SDC* for each group in the specified grouping method. Note that `group` and `all_groups` cannot be used together. A typical usecase could be to compute the *MSD* for each molecule in a system. The parameters are:

- `all_groups`, where `group_method` is the grouping method to use for computation

Finally, the third special keyword is `save_every`. This keyword saves the internal *MSD* and *SDC* computed so far during the simulation, which can be helpful during long running simulations. The file will have a name formatted as `msd_step[step].out`. The parameters are:

- `save_every`, where `interval` is the number of steps between saving a copy. Note that the copy can only be written at most every `sample_interval` steps. Furthermore, the first `msd_step[step].out` file will be written after `Nc` times `sample_interval` steps. Subsequent files will be written every `interval`.

Examples

An example of this function is:

```
compute_msd 5 200 group 1 1
```

This means that you

- want to calculate the *MSD*
- the position data will be recorded every 5 steps
- the maximum number of correlation steps is 200
- you would like to compute only over group 1 in group method 1.

To compute the *MSD* for all groups in group method 1 and save a copy of the *MSD* every 100 000 steps, one can write:

```
compute_msd 5 200 all_groups 1 save_every 100000
```

5.2.46 compute_shc

The `compute_shc` keyword is used to compute the non-equilibrium virial-velocity correlation function $K(t)$ and the spectral heat current (*SHC*) $J_q(\omega)$, in a given direction, for a group of atoms, as defined in Eq. (18) and the left part of Eq. (20) of [Fan2019]. The results are written to the *shc.out output file*.

Syntax

```
compute_shc <sample_interval> <Nc> <transport_direction> <num_omega> <max_omega> [{
↪<optional_arg>}]
```

`sample_interval` is the sampling interval (number of steps) between two correlation steps. This parameter must be an integer that is ≥ 1 and ≤ 10.

`Nc` is the total number of correlation steps. This parameter must be an integer that is ≥ 100 and ≤ 1000.

`transport_direction` is the direction of heat transport to be measured. It can only be 0, 1, and 2, corresponding to the x, y, and z directions, respectively.

`num_omega` is the number of frequency points one wants to consider.

`max_omega` is the maximum angular frequency (in units of THz) one wants to consider. The angular frequency data will be `max_omega/num_omega, 2*max_omega/num_omega, ..., max_omega`.

`<optional_arg>` can only be `group`, which requires two parameters:

```
group <grouping_method> <group_id>
```

This means that $K(t)$ will be calculated for atoms in group `group_id` of grouping method `grouping_method`. Usually, `group_id` should be ≥ 0 and smaller than the number of groups in grouping method `grouping_method`. If `grouping_method` is assigned and `group_id` is -1, it means to calculate the $K(t)$ for every `group_id` except for `group_id` 0 in the assigned `grouping_method`. Since it is very time and memory consuming to calculate the all group $K(t)$ for a large system, so one can assign the part that don't want to calculate to `group_id` 0. Also, grouping method `grouping_method` must be defined in the *simulation model input file*. If this option is missing, it means computing $K(t)$ for the whole system.

Examples

Example 1

The command:

```
compute_shc 2 250 0 1000 400.0
```

means that

- you want to calculate $K(t)$ for the whole system
- the sampling interval is 2
- the maximum number of correlation steps is 250
- the transport direction is x
- you want to consider 1000 frequency points
- the maximum angular frequency is 400 THz

Example 2

The command:

```
compute_shc 1 500 1 500 200.0 group 0 4
```

means that

- you want to calculate $K(t)$ for atoms in group 4 defined in grouping method 0
- the sampling interval is 1 (sample the data at each time step)
- the maximum number of correlation steps is 500
- the transport direction is y
- you want to consider 500 frequency points
- the maximum angular frequency is 200 THz

Example 3

The command:

```
compute_shc 1 500 1 500 200.0 group 1 -1
```

means that

- you want to calculate $K(t)$ for all `group_id` except for `group_id` 0 defined in grouping method 1

- the sampling interval is 1 (sample the data at each time step)
- the maximum number of correlation steps is 500
- the transport direction is y
- you want to consider 500 frequency points
- the maximum angular frequency is 200 THz

Caveats

This computation can be memory consuming.

If you want to use the in-out decomposition for 2D materials, you need to make the basal plane in the xy directions.

5.2.47 compute_viscosity

This keyword can be used to calculate the stress autocorrelation function and viscosity using the Green-Kubo method. The results will be written to the *viscosity.out output file*.

Syntax

This keyword has 2 parameters:

```
compute_viscosity <sampling_interval> <correlation_steps>
```

The first parameter is the sampling interval for the stress data. The second parameter is the total number of correlations steps.

5.2.48 compute_lsqt

This keyword is used to compute the electronic transport properties using the linear-scaling quantum transport (*LSQT*) [Fan2021b] approach. *LSQT* and *MD* are coupled to account for electron-phonon scattering. Currently, only a tight-binding model for carbon is supported (hard coded). The results will be written into the files lsqt_dos.out, lsqt_velocity.out, and lsqt_sigma.out. This feature is preliminary and changes might be made in the near future.

Syntax

This keyword is used as follows:

```
compute_lsqt <transport_direction> <num_moments> <num_energies> <E_1> <E_2> <E_max>
```

- `transport_direction` is the transport direction, which can take values `x`, `y`, and `z`.
- `num_moments` is the number of the Chebyshev moments for the energy resolution operator.
- The `num_energies` energy points increase linearly from `E_1` (eV) to `E_2` (eV).
- `E_max` (eV) is an energy value that should be (slightly) larger than the maximum of the absolute energy of the tight-binding model. This can be determined by trial and error.

Example

```
compute_lsqt x 3000 10001 -8.1 8.1 8.2
```

5.2.49 compute_rdf

This keyword is used to compute the radial distribution function (*RDF*) for all atoms or pairs of species. It works for both classical *MD* and *PIMD*. The results will be written to the *rdf.out* file.

Syntax

This keyword is used as follows:

```
compute_rdf <cutoff> <num_bins> <interval> [atom <i1> <i2> atom <i3> <i4> ...]
```

This means that the *RDF* calculations will be performed every `interval` steps, with `num_bins` data points evenly distributed from 0 to `cutoff` (in units of Ångstrom) in terms of the distance between atom pairs.

Without the optional parameters, only the total *RDF* will be calculated.

To additionally calculate the partial *RDF* for a pair of species, one can specify the types of the two species after the word "atom". The types 0, 1, 2, ... correspond to the species in the potential file in order. Currently, one can specify at most 6 pairs.

Example

compute_rdf 8.0 400 1000 # total RDF every 1000 MD steps with 400 data up to 8 Angstrom compute_rdf 8.0 400 1000 atom 0 0 atom 1 1 atom 0 1 # additionally calculate 3 partial RDFs

5.2.50 compute_angular_rdf

This keyword is used to compute the angular-dependent radial distribution function (*ARDF*) for all atoms or pairs of species. It works for classical *MD*. The results will be written to the *angular_rdf.out* file.

Mathematical details

The ARDF is defined as

$$g(r,\theta) = \frac{n(r,\theta)}{\rho 2 r^2 \Delta r \Delta\theta},$$

where $n(r,\theta)$ is the number of pairs of atoms at a distance $(r-\Delta r/2, r+\Delta r/2]$ and an angle $(\theta-\Delta\theta/2, \theta+\Delta\theta/2]$ from each other, ρ is the number density, r is the distance between the atoms, θ is the angle between the atoms, Δr is the bin width in distance, and $\Delta\theta$ is the bin width in angle.

Syntax

This keyword is used as follows:

```
compute_angular_rdf <cutoff> <r_num_bins> <angular_num_bins> <interval> [atom <i1> <i2>␣
↪atom <i3> <i4> ...]
```

This means that the ARDF calculations will be performed every `interval` steps, with `r_num_bins` data points evenly distributed from 0 to `cutoff` (in units of Ångstrom) in terms of the distance between atom pairs, and `angular_num_bins` data points evenly distributed from -Pi/2 to Pi/2 in terms of the angle.

Without the optional parameters, only the total *ARDF* will be calculated.

To additionally calculate the partial *ARDF* for a pair of species, one can specify the types of the two species after the word "atom". The types 0, 1, 2, ... correspond to the species in the potential file in order. Currently, one can specify at most 6 pairs.

Example

compute_angular_rdf 8.0 400 100 1000 # total ARDF every 1000 MD steps with 400 bins in distance and 100 bins in angle up to 8 Angstrom compute_angular_rdf 8.0 400 100 1000 atom 0 0 atom 1 1 atom 0 1 # additionally calculate 3 partial ARDFs

5.2.51 active

Run on-the-fly active learning, based on committee uncertainty estimates over a group of supplied NEP potentials. Note that this mode is only supported with NEP potentials. Furthermore, the molecular dynamics simulation is propagated using the first NEP potential specified in *run.in*.

The uncertainty σ_f is estimated as the maximum force sample standard deviation on any atom i,

$$\sigma_f = \max_i \sqrt{\sigma_{i,x}^2 + \sigma_{i,y}^2 + \sigma_{i,z}^2},$$

where $\sigma_{i,k}^2$, $k \in x, y, z$, are the sample variances in the k Cartesian direction calculated over the M models. If the uncertainty exceeds the specified threshold, $\sigma_f > \delta$, for a structure in a step of an molecular dynamics simulation, then that structure is appended to the file *active.xyz* in the extended XYZ format. Additionally, the simulation time t and σ_f are written to the file *active.out* regardless of if $\sigma_f > \delta$.

active takes five arguments. The first four sets the interval for uncertainty estimation and what per atom quantities are outputted in *active.xyz*, the fifth keyword sets the threshold δ in units of eV/Å.

Syntax

```
active <interval> <has_velocity> <has_force> <has_uncertainty> <threshold>
```

`interval` is the interval (number of steps) between checking the uncertainty. If set to 1, the uncertainty will be computed for every step of the MD simulation. `has_velocity` can be 1 or 0, which means the velocities will or will not be included in the exyz output. `has_force` can be 1 or 0, which means the forces will or will not be included in the exyz output. `has_uncertainty` can be 1 or 0, which means the per atom uncertainties will or will not be included in the exyz output. `threshold` is a non-negative float, and corresponds to the threshold δ in units of eV/Å.

Examples

Example 1

To run on-the-fly active learning using a committee of 5 NEP potentials, checking if the uncertainty exceeds 0.01 eV/Å every tenth MD step, write:

```
potential nep0
potential nep1
potential nep2
potential nep3
potential nep4
...
active 10 1 1 1 0.01
```

before the *run keyword*. This will generate two output files, *active.xyz* and *active.out*.

Caveats

- This keyword is not propagating. That means, its effect will not be passed from one run to the next.
- Molecular dynamics will be run with the first potential specified.

- If the system has exploded, unphysical structures may be saved since no upper bound is set on the uncertainty σ_f. Ensure that the resulting structures in *active.xyz* are physical.

5.2.52 dump_exyz

Write some data into dump.xyz in extended XYZ format.

Syntax

```
dump_exyz <interval> <has_velocity>
dump_exyz <interval> <has_velocity> <has_force>
dump_exyz <interval> <has_velocity> <has_force> <has_potential>
dump_exyz <interval> <has_velocity> <has_force> <has_potential> <separated>
```

Here, the `interval` parameter is the output interval (number of steps) of the data. `has_velocity` can be 1 or 0, which means the velocities will or will not be included in the output. `has_force` can be 1 or 0, which means the forces will or will not be included in the output. `has_potential` can be 1 or 0, which means the atomic potential energies will or will not be included in the output. The atomic positions will always be included in the output. `separated` can be 1 or 0, which means the output will or will not be separated into individual frames.

Examples

```
dump_exyz 1000         # dump positions every 1000 steps
dump_exyz 1000 1       # dump positions and velocities
dump_exyz 1000 1 1     # dump positions, velocities, and forces
dump_exyz 1000 1 1 1   # dump positions, velocities, forces, and potentials
dump_exyz 1000 0 1 1   # dump positions, forces and potentials
dump_exyz 100 0 1 1 1 # dump positions, forces and potentials into dump.100.xyz, dump.
↪200.xyz and so on
```

Caveats

- This keyword is not propagating. That means, its effect will not be passed from one run to the next.
- The output file has an appending behavior and will result in a single *dump.xyz* file no matter how many times the simulation is run.

5.2.53 dump_xyz

Write per-atom data into user-specified file(s) in extended XYZ format.

Syntax

```
dump_xyz <grouping_method> <group_id> <inverval> <filename> {<property_1> <property_2> ..
↪.}
```

- `grouping_method` and `group_id` are the grouping method and the related group ID to be used.

If `grouping_method` is negative, `group_id` will be ignored and data for the whole system will be output.

- `interval` is the output interval (number of steps) of the data.
- `filename` is the output file.

If it is ended by a star (*), the data for one frame will be output to one file, named by changing the star to the step number.

- Then one can write the properties to be output, and the allowed properties include: `mass`, `velocity`, `force`, `potential`, `virial`, `group`, and `unwrapped_position`.
- The wrapped positions will always be included in the output.

Examples

```
ensemble xxx # some ensemble

# dump positions every 1000 steps, for the whole system:
dump_xyz -1 1 1000 positions.xyz

# dump many other quantities every 100 steps, for atoms in group 0 of grouping method 1:
dump_xyz 1 0 100 properties.xyz mass velocity potential force virial

run 1000000
```

Caveats

- This keyword is not propagating. That means, its effect will not be passed from one run to the next.
- The output file has an appending behavior.
- Different from many of the other keywords, this keyword is allowed to be invoked multiple times within one run.

5.2.54 dump_beads

Write some data (positions and optionally velocities and forces) into *beads_dump_<k>.xyz* in extended XYZ format, where *<k>* runs from 0 to *number_of_beads - 1* as set in a run related to path-integral molecular dynamics (*PIMD*). Each file thus contains the data for one bead and the format is identical to the *dump.xyz output file* as generated by the *dump_exyz keyword*.

Caveats

- The centroid data (the average over all the beads) for the ring polymer in PIMD-related runs are still output by the *dump_exyz keyword*.
- This keyword should not be used in a run not related to PIMD.

5.2.55 dump_observer

Writes atomistic properties such as positions, velocities and forces for each of the supplied NEP potentials in the extended XYZ format. Takes the same arguments as *dump_exyz keyword*, and additionally a keyword *mode*. *mode* can either be set to *observe* or *average*.

If set to *observe*, the first of the supplied NEP potentials will be used to propagate the molecular dynamics run, and the remaining potentials will be evaluated every *interval_thermo* and *interval_exyz* time steps. Every *interval_thermo* timesteps files in the style of *thermo.out* will be written, and every *interval_exyz* timesteps extended XYZ-files will be written. The files are named according to the following convention:

- **.out**: *observer0.out*, *observer1.out*, ..., *observer(N-1).out* for *N* supplied potentials.
- **.xyz**: *observer0.xyz*, *observer1.xyz*, ..., *observer(N-1).xyz* for *N* supplied potentials.

The index of these *observer(index)* files correspond to the index of each potential in the *run.in* file. Thus, *observer0* corresponds to the first potential, *observer1* to the second and so on. In this mode, *observer0* corresponds to the main potential.

If set to *average*, all supplied NEP potentials will be evaluated at every timestep, with the average of all potentials used to propagate the molecular dynamics. In this case, two files will be written: *observer.out* every *interval_thermo* timesteps, and *observer.xyz* every *interval_exyz* timesteps. These files contains the thermo and atomistic properties as calculated with the average potential.

Note that the supplied potentials must have their atomic species written in the same order, i.e. the line *nep* n_species species0 species1* must be the same in all potential files.

Syntax

```
dump_observer <mode> <interval_thermo> <interval_exyz> <has_velocity> <has_force>
```

`mode` corresponds to the two cases described above, and can be either *observe* or *average*. `interval_thermo` parameter is the output interval (number of steps) for writing thermo files. `interval_exyz` parameter is the output interval (number of steps) of writing exyz files. `has_velocity` can be 1 or 0, which means the velocities will or will not be included in the exyz output. `has_force` can be 1 or 0, which means the forces will or will not be included in the exyz output.

Examples

Example 1

To use one NEP potential to propagate the MD (*nep0*), and another (*nep1*) to observe thermo properties every 100 steps and write exyz files every 1000 steps, write:

```
potential nep0
potential nep1
...
dump_observer observe 100 1000 1 1
```

before the *run keyword*. This will generate four output files, *observer0.xyz*, *observer0.out*, *observer1.xyz* and *observer1.out*, containing thermo properties and positions, velocities and forces as calculated with *nep0* and *nep1* respectively.

Example 2

To run MD with an average of two NEP potentials, *nep0* and *nep1*, and dump the positions, velocities and forces every 1000 steps, write:

```
potential nep0
potential nep1
...
dump_observer average 100 1000 1 1
```

before the *run keyword*. This will generate two output files, *observer.out* containing thermo properties and *observer.xyz*, containing positions, velocities and forces as calculated with the average of *nep0* and *nep1*. *observer.out* will be written every 100 timesteps, and *observer.xyz* every 1000 timesteps.

Caveats

- This keyword is not propagating. That means, its effect will not be passed from one run to the next.
- If *mode* is set to *observe*, then the output file has an appending behavior and will result in two files, *observer(index).out* and *observer(index).xyz* file for each potential no matter how many times the simulation is run.

- If *mode* is set to *average*, then the output file has an appending behavior and will result in a single *observer.xyz* file no matter how many times the simulation is run.

5.2.56 dump_dipole

Predicts the dipole for the current configuration of atoms during MD, using the second supplied NEP. The first potential should be a regular NEP potential model and is used to run the MD, whilst the second NEP should be a *nep*_dipole* model.

Syntax

```
dump_dipole  <interval>
```

`interval` parameter is the output interval (number of steps) for evaluating and writing the dipole.

Examples

Example 1

To use one NEP potential to propagate the MD (*nep0*), and another (*nep1*) to compute and write the dipole every 100 steps, write:

```
potential nep0
potential nep1
...
dump_dipole 100
```

before the *run keyword*. This will generate a *dipole.out* output files containing the time step and predicted dipole at that time step.

Caveats

- This keyword is not propagating. That means, its effect will not be passed from one run to the next.

5.2.57 dump_polarizability

Predicts the polarizability for the current configuration of atoms during MD, using the second supplied NEP. The first potential should be a regular NEP potential model and is used to run the MD, whilst the second NEP should be a *nep*_polarizability* model.

Syntax

```
dump_polarizability  <interval>
```

`interval` parameter is the output interval (number of steps) for evaluating and writing the polarizability.

Examples

Example 1

To use one NEP potential to propagate the MD (*nep0*), and another (*nep1*) to compute and write the polarizability every 100 steps, write:

```
potential nep0
potential nep1
...
dump_polarizability 100
```

before the *run keyword*. This will generate a *polarizability.out* output files containing the time step and predicted polarizability at that time step.

Caveats

- This keyword is not propagating. That means, its effect will not be passed from one run to the next.

5.2.58 dump_force

Write the atomic forces to *force.out output file*.

Syntax

```
dump_force <interval> {<optional_args>}
```

The `interval` parameter is the output interval (number of steps) of the atom forces. At the moment, the optional arguments (`optional_args`) can only assume the value `group`. The option `group` shoud have two parameters:

```
group <grouping_method> <group_id>
```

which means only dumping forces of atoms in group `group_id` within the grouping method `grouping_method`. If this option is not used, the forces will be written for all the atoms.

Examples

Example 1

To dump all the forces every 10 steps for a run, one can add:

```
dump_force 10
```

before the *run keyword*.

Example 2

Similar to the above example, but only for atoms in group 1 within grouping method 2:

```
dump_force 10 group 2 1
```

5.2.59 dump_netcdf

Write the atomic positions (coordinates) and (optionally) velocities in NetCDF format to a movie.nc file.

Syntax

This keyword has the following format:

```
dump_netcdf <interval> <has_velocity> [{optional_args}]
```

The `interval` parameter is the output interval (number of steps) of the atom positions. *has_velocity* can be 1 or 0, which means the velocities will or will not be included in the output. The optinal arguments (`optional_args`) provide additional functionality. Currently, the following optional argument is accepted:

- `precision`
  - If `value` is `single`, the output data are 32-bit floating point numbers.
  - If `value` is `double`, the output data are 64-bit floating point numbers.

  The default value is `double`.

Requirements and specifications

- This keyword requires an external package to operate. Instructions for how to set up the NetCDF package can be found *here*.
- The NetCDF format follows the AMBER specifications.
- The `single` option is good for saving space, but does not follow the official AMBER 1.0 conventions.
- NetCDF output files can be read for example by VMD or OVITO for visualization.
- The NetCDF files also contain cell lengths and angles, which can be used in the visualization software to illustrate boundaries and show periodic copies of the structure.

Examples

Single precision without veolocities

To dump the positions every 1000 steps to a NetCDF file with 32-bit floating point values, one can add:

```
dump_netcdf 1000 0 precision single
```

before the *run command*.

Double precision with velocities

To dump the positions and velocities every 1000 steps to a NetCDF file with 64-bit floating point values, one can add:

```
dump_netcdf 1000 1
```

before the *run command*.

Caveats

- Following the AMBER 1.0 conventions, length is in units of Ångström and velocity is in units of Ångström/picosecond.
- This keyword is not propagating. That means, its effect will not be passed from one run to the next.
- The output appends to the same file for different runs in the same simulation. Re-running the simulation will create a new output file.
- If the file "movie.nc" already exists in the current directory, it will not be overwritten. Instead, files will be generated with names "movie_2.nc", "movie_3.nc", ..., "movie_n.nc".
- If the `precision` changes between different runs, the first defined precision will still be used (i.e., changes in precision are ignored during a simulation).

5.2.60 dump_position

Write the atomic positions (coordinates) to the *movie.xyz output file*.

Syntax

```
dump_position <interval> [{optional_args}]
```

The `interval` parameter is the output interval (number of steps) of the atom positions.

The optional arguments (`<optional_args>`) can be `group` or `precision`, which can be in any order.

The option `group` shoud have two parameters:

```
group <grouping_method> <group_id>
```

which means only dumping positions of atoms in group `group_id` within the grouping method `grouping_method`. If this option is not used, positions will be dumped for all the atoms.

The option `precision` should have one parameter which can only be `single` or `double`:

```
precision single # output data with %0.9f format
precision double # output data with %0.17f format
```

If this option is not used, data will be output with the `%g` format.

Examples

Example 1

To dump all the positions every 1000 steps for a run, one can add:

```
dump_position 1000
```

before the *run keyword*.

Example 2

Similar to the above example, but only for atoms in group 1 within grouping method 2:

```
dump_position 1000 group 2 1
```

Example 3

Similar to the above example, but using double precision:

```
dump_position 1000 group 2 1 precision double
```

or equivalently:

```
dump_position 1000 precision double group 2 1
```

Caveats

This keyword is not propagating. That means, its effect will not be passed from one run to the next.

The output file has an appending behavior and will result in a single *movie.xyz output file* file no matter how many times the simulation is run.

5.2.61 dump_restart

Write a restart file.

Syntax

This keyword only requires a single parameter, which is the output interval (number of steps) of updating the restart file:

```
dump_restart <interval>
```

Example

To update the restart file every 100000 steps for a run, one can add:

```
dump_restart 100000
```

before the *run keyword*.

Caveats

This keyword is not propagating. That means, its effect will not be passed from one run to the next.

5.2.62 dump_thermo

This keyword controls the writing of global thermodynamic quantities to the *thermo.out output file*.

Syntax

This keyword only requires a single parameter, which is the output interval (number of steps) of the global thermodynamic quantities:

```
dump_thermo <interval>
```

Example

To dump the global thermodynamic quantities every 1000 steps for a run, one can add:

```
dump_thermo 1000
```

before the *run keyword*.

Caveats

This keyword is not propagating. That means, its effect will not be passed from one run to the next.

5.2.63 dump_velocity

Dump the atomic velocities to the *velocity.out output file*.

Syntax

```
dump_velocity <interval> [{optional_args}]
```

Here, the `interval` parameter is the output interval (number of steps) of the atomic velocities. At the moment, the only optional argument (`optional_args`) is `group`. The option `group` shoud have two parameters:

```
group <grouping_method> <group_id>
```

which means only dumping velocities of atoms in group `group_id` within the grouping method `grouping_method`. If this option is not used, velocities will be dumped for all the atoms.

Examples

Example 1

To dump all the velocities every 10 steps for a run, one can add:

```
dump_velocity 10
```

before the *run keyword*.

Example 2

Similar to the above example, but only for atoms in group 1 within grouping method 2:

```
dump_velocity 10 group 2 1
```

5.2.64 dump_shock_nemd

In shock wave piston simulations, it's often crucial to compute thermo information at different regions, both before and after the shock wave passage.

Piston simulations commonly involve millions of atoms. Dumping all the virial and velocity data for each atom can lead to excessively large output files, making data processing cumbersome. The *dump_shock_nemd* command addresses this by calculating spatial thermo information during the simulation.

This feature calculates the spatial distribution of partical velocity (km/h), stress (GPa), temperature and density (g/cm3) in x-direction.

Syntax

```
dump_shock_nemd interval <time_interval> bin_size <size of each bin>
```

- The `interval` parameter sets the output interval (number of steps).
- The `bin_size` parameter, optional with a default value of 10, defines the thickness of each histogram bin, measured in Angstroms.

Examples

To output spatial thermo information every 1000 steps for a run in the x-direction, with a bin size of 20 Angstroms, include the following before the *run keyword*:

```
dump_shock_nemd interval 1000 bin_size 20
```

5.3 Output files

File name	Generating keyword	Brief description	Outp
thermo.out	*dump_thermo*	Global thermodynamic quantities	Appe
movie.xyz	*dump_position*	Trajectory (atomic positions, velocities etc)	Appe
restart.xyz	*dump_restart*	The restart file	Over
dump.xyz	*dump_exyz*	Atomistic positions, velocities and forces.	Appe
observer.xyz	*dump_observer*	Atomistic positions, velocities and forces as evaluated with observing potentials.	Appe
observer.out	*dump_observer*	Thermodynamic quantities evaluated with observing potentials.	Appe
active.xyz	*active*	Structures selected through active learning.	Appe
active.out	*active*	Simulation time and uncertainty during active learning.	Appe
dipole.out	*dump_dipole*	Predicted dipole.	Appe
polarizability.out	*dump_polarizability*	Predicted polarizability.	Appe
velocity.out	*dump_velocity*	Contains the atomic velocities	Appe
force.out	*dump_force*	Contains the atomic forces	Appe
compute.out	*compute*	Time and space (group) averaged quantities	Appe
hac.out	*compute_hac*	Thermal conductivity data from the *EMD* method	Appe
kappa.out	*compute_hnemd*	Thermal conductivity data from the *HNEMD* method	Appe
shc.out	*compute_shc*	Spectral heat current (*SHC*) data	Appe
heatmode.out	*compute_gkma*	Modal heat current data from *GKMA* method	Appe
kappamode.out	*compute_hnema*	Modal thermal conductivity data from *HNEMA* method	Appe
dos.out, *mvac.out*	*compute_dos*	Phonon density of states (*PDOS*) data	Appe
sdc.out	*compute_sdc*	Self-diffusion coefficient (*SDC*) data	Appe
msd.out	*compute_msd*	Mean-square displacement (*MSD*) data	Appe
cohesive.out	*compute_cohesive*	Cohesive energy curve	Over
D.out	*compute_phonon*	Dynamical matrices $D(\boldsymbol{k})$ for the input k points	Over
omega2.out	*compute_phonon*	Phonon frequency squared $\omega^2(\boldsymbol{k})$ for the input $\boldsymbol{k}$-points	Over
viscosity_out	*compute_viscosity*	Viscosity and stress auto-correlation function	Appe
onsager.out	*compute_hnemdec*	Onsager coefficients	Appe
rdf.out	*compute_rdf*	Radial distribution function (*RDF*)	Appe
adf.out	*compute_adf*	Angular distribution function (*ADF*)	Appe
angular_rdf.out	*compute_angular_rdf*	Angular-dependent radial distribution function (*ARDF*)	Appe
mcmd.out	*mc*	Acceptance ratio and species concentrations	Appe
lsqt_dos.out	*compute_lsqt*	Electronic density of states	Appe
lsqt_velocity.out	*compute_lsqt*	Electron group velocity	Appe
lsqt_sigma.out	*compute_lsqt*	Electrical conductivity	Appe
orientorder.out	*compute_orientorder*	Steinhardt bond-orientational order parameters	Appe

5.3.1 `cohesive.out`

This file contains the cohesive energy data produced when invoking the *compute_cohesive keyword*.

File format

There are two columns, the first column gives the isotropic scaling factors (dimensionless) and the second column gives the corresponding potential energies (in units of eV) after energy minimization.

5.3.2 `compute.out`

This file contains space and time averaged quantities. It is produced when invoking the *compute keyword*.

File format

Assuming that the system is divided into M groups according to the grouping method used by the *compute keyword* keyword, then:

- if temperature is computed, there are M columns of group temperatures (in units of K) from left to right
- if potential is computed, there are M columns of group potentials (in units of eV) from left to right
- if force is computed: there are $3M$ columns of group forces (in units of eV/A) from left to right in the following form:

```
fx_1 ... fx_M fy_1 ... fy_M fz_1 ... fz_M
```

- if virial is computed, there are $9M$ columns of group virials (in units of eV) from left to right in the order of xx, xy, xz, yx, yy, yz, zx, zy, zz.
- if the potential part of the heat current is computed, there are $3M$ columns of group potential heat currents (in units of $\text{eV}^{3/2}$ $\text{amu}^{-1/2}$) from left to right in a form similar to that for force
- if the kinetic part of the heat current is computed, there are $3M$ columns of group kinetic heat currents (in units of $\text{eV}^{3/2}$ $\text{amu}^{-1/2}$) from left to right in a form similar to that for force
- if momentum is computed, there are $3M$ columns of group momenta (in units of $\text{amu}^{1/2}$ $\text{eV}^{1/2}$) from left to right in a form similar to that for force
- if temperature is computed, the last second column is the total energy of the thermostat coupling to the heat source region (in units of eV) and the last column is the total energy of the thermostat coupling to the heat sink region (in units of eV)

Note that regardless of the order of properties in the *compute keyword*, the order of the output data is fixed as given above.

5.3.3 `D.out`

This file contains the dynamical matrices $D(\boldsymbol{k})$ at different $\boldsymbol{k}$-points.

File format

The file contains `3 * N_basis * N_kpoints` rows and `6 * N_basis` columns, where `N_basis` is the number of basis atoms in the unit cell defined in the *basis.in input file* and `N_kpoints` is the number of $\boldsymbol{k}$-points in the *kpoints.in input file*.

Each consecutive `3 * N_basis` rows correspond to one $\boldsymbol{k}$-point. For each $\boldsymbol{k}$-point, the first `3*N_basis` columns correspond to the real part and the second `3*N_basis` columns correspond to the imaginary part. Schematically, the file is organized as:

```
D_real(k_0) D_imag(k_0)
D_real(k_1) D_imag(k_1)
...
```

The matrix elements are in units of eV Å^{-2} amu^{-1}.

5.3.4 `dos.out`

This file contains the data of the phonon density of states (*PDOS*). It is produced when invoking *compute_dos keyword* in the *run.in input file*.

File format

The file is organized as follows:

- column 1: angular frequency ω (in units of THz)
- column 2: *DOS* (in units of 1/THz) in the x direction
- column 3: *DOS* (in units of 1/THz) in the y direction
- column 4: *DOS* (in units of 1/THz) in the z direction

If there are multiple groups to be calculated as specified in the *compute_dos keyword*, data will be output group by group, each occupying the same number of rows (the number of frequency points).

5.3.5 `force.out`

This file contains the per-atom forces of the atoms sampled at a given frequency. It is produced when invoking *dump_force keyword*.

File format

```
fx_1(1) fy_1(1) fz_1(1)
fx_2(1) fy_2(1) fz_2(1)
...
fx_N(1) fy_N(1) fz_N(1)
fx_1(2) fy_1(2) fz_1(2)
vx_2(2) fy_2(2) fz_2(2)
...
fx_N(2) fy_N(2) fz_N(2)
...
```

- There are three columns, corresponding to the x, y, and z components of the force.
- Each `N` consecutive lines correspond to one frame at a given time point.
- The number of lines equals the number of frames times `N`.
- `fx_m(n)` is the x component of the `m`-th atom in the `n`-th frame.
- `fy_m(n)` is the y component of the `m`-th atom in the `n`-th frame.
- `fz_m(n)` is the z component of the `m`-th atom in the `n`-th frame.
- Force components are in units of eV/Å.

5.3.6 `hac.out`

This file contains the heat current auto-correlation (*HAC*) function and the running thermal conductivity (*RTC*) from the the *EMD method for heat transport* method. It is produced when invoking the *compute_hac keyword* in the *run.in input file*.

File format

This file reads

- column 1: correlation time (in units of ps)
- column 2: $\langle J_x^{\rm in}(0) J_x^{\rm tot}(t)\rangle$ (in units of eV^3/amu)
- column 3: $\langle J_x^{\rm out}(0) J_x^{\rm tot}(t)\rangle$ (in units of eV^3/amu)

- column 4: $\langle J_y^{\text{in}}(0)J_y^{\text{tot}}(t)\rangle$ (in units of eV^3/amu)
- column 5: $\langle J_y^{\text{out}}(0)J_y^{\text{tot}}(t)\rangle$ (in units of eV^3/amu)
- column 6: $\langle J_z^{\text{tot}}(0)J_z^{\text{tot}}(t)\rangle$ (in units of eV^3/amu)
- column 7: $\kappa_x^{\text{in}}(t)$ (in units of W/mK)
- column 8: $\kappa_x^{\text{out}}(t)$ (in units of W/mK)
- column 9: $\kappa_y^{\text{in}}(t)$ (in units of W/mK)
- column 10: $\kappa_y^{\text{out}}(t)$ (in units of W/mK)
- column 11: $\kappa_z^{\text{tot}}(t)$ (in units of W/mK)

Note that the *HAC* and the *RTC* are decomposed as described in [Fan2017]. This decomposition is useful for 2D materials but not necessary for 3D materials. For 3D materials, one can sum up some columns to get the conventional data. For example:

$$\langle J_x^{\text{tot}}(0)J_x^{\text{tot}}(t)\rangle = \langle J_x^{\text{in}}(0)J_x^{\text{tot}}(t)\rangle + \langle J_x^{\text{out}}(0)J_x^{\text{tot}}(t)\rangle$$

and

$$\kappa_x^{\text{tot}}(t) = \kappa_x^{\text{in}}(t) + \kappa_x^{\text{out}}(t).$$

Note that the cross term introduced in [Fan2017] has been evenly attributed to the in-plane and out-of-plane components. This has been justified in [Fan2019].

Only the potential part of the heat current is included. If the convective part of the heat current is important in your system, you can use the *compute keyword* to calculate and output the heat current data and post-process it by yourself.

5.3.7 `heatmode.out`

This file contains the modal heat current generated by the Green-Kubo Modal Analysis (*GKMA*) method. It is produced when invoking the *compute_gkma keyword* in the *run.in input file*.

File format

This file reads:

$$\begin{matrix}
J_{1,1}^{x,in} & J_{1,1}^{x,out} & J_{1,1}^{y,in} & J_{1,1}^{y,out} & J_{1,1}^{z} & \cdots \\
J_{1,2}^{x,in} & J_{1,2}^{x,out} & J_{1,2}^{y,in} & J_{1,2}^{y,out} & J_{1,2}^{z} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
J_{1,n}^{x,in} & J_{1,n}^{x,out} & J_{1,n}^{y,in} & J_{1,n}^{y,out} & J_{1,n}^{z} & \cdots \\
J_{2,1}^{x,in} & J_{2,1}^{x,out} & J_{2,1}^{y,in} & J_{2,1}^{y,out} & J_{2,1}^{z} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{matrix}$$

- Each output is in units of eV$^{3/2}$amu$^{-1/2}$.
- The indices denote the output number (left) and the bin (right).
- Each output will have n bins, which are determined by the arguments of the *compute_gkma keyword*.

5.3.8 `kappa.out`

This file contains some components of the running thermal conductivity (*RTC*) tensor from the homogeneous nonequilibrium molecular dynamics (*HNEMD*) method. It is generated when invoking the *compute_hnemd keyword*.

File format

If the driving force is in the μ (μ can be x, y, or z) direction, this file reads:

- column 1: $\kappa_{\mu x}^{\text{in}}(t)$ (in units of W/mK)
- column 2: $\kappa_{\mu x}^{\text{out}}(t)$ (in units of W/mK)
- column 3: $\kappa_{\mu y}^{\text{in}}(t)$ (in units of W/mK)
- column 4: $\kappa_{\mu y}^{\text{out}}(t)$ (in units of W/mK)
- column 5: $\kappa_{\mu z}^{\text{tot}}(t)$ (in units of W/mK)

Both $\kappa_{\mu x}(t)$ and $\kappa_{\mu y}(t)$ have been decomposed into comtributions from in-plane (hence superscript in) and out-of-plane (hence superscript out) vibrational modes, as described in [Fan2019]. This decomposition is useful for 2D (or layered) materials but is not necessary for 3D materials. For 3D materials, one can sum up some columns to get the conventional data. That is:

$$\kappa_{\mu x}^{\text{tot}}(t) = \kappa_{\mu x}^{\text{in}}(t) + \kappa_{\mu x}^{\text{out}}(t)$$
$$\kappa_{\mu y}^{\text{tot}}(t) = \kappa_{\mu y}^{\text{in}}(t) + \kappa_{\mu y}^{\text{out}}(t).$$

Only the potential part of the heat current has been considered. To simulation systems in which the convective heat current is important, one would have to modify the source code.

5.3.9 `kappamode.out`

This file contains the modal thermal conductivity generated by the homogeneous nonequilibrium modal analysis (*HNEMA*) method. It is generated when invoking the *compute_hnema keyword*.

File format

This file reads:

$$\begin{array}{ccccc}
\kappa_{1,1}^{x,in} & \kappa_{1,1}^{x,out} & \kappa_{1,1}^{y,in} & \kappa_{1,1}^{y,out} & \kappa_{1,1}^{z} \\
\kappa_{1,2}^{x,in} & \kappa_{1,2}^{x,out} & \kappa_{1,2}^{y,in} & \kappa_{1,2}^{y,out} & \kappa_{1,2}^{z} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\kappa_{1,n}^{x,in} & \kappa_{1,n}^{x,out} & \kappa_{1,n}^{y,in} & \kappa_{1,n}^{y,out} & \kappa_{1,n}^{z} \\
\kappa_{2,1}^{x,in} & \kappa_{2,1}^{x,out} & \kappa_{2,1}^{y,in} & \kappa_{2,1}^{y,out} & \kappa_{2,1}^{z} \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{array}$$

- The output is in units of $\text{Wm}^{-1}\text{K}^{-1/2}$.
- The indices denote the output number (left) and the bin (right).
- Each output comprises n bins, which are determined by the arguments of the *compute_hnema keyword*.

5.3.10 `mvac.out`

This file contains the data of mass-weighted velocity autocorrelation (*VAC*). The file is generated when invoking the *compute_dos keyword*.

File format

The file is organized as follows:

- column 1: correlation time (in units of ps)
- column 2: VAC (in units of $\text{Å}^2/\text{s}^2$) in the math:x direction

- column 3: VAC (in units of $Å^2/s^2$) in the math:y direction
- column 4: VAC (in units of $Å^2/s^2$) in the math:z direction

Only the group selected via the *compute_dos keyword* is included in the output.

5.3.11 movie.xyz

This file contains the positions (trajectory) of the atoms sampled at a given frequency. It is geenrated when invoking the *dump_position keyword*.

File format

This file is written in extended xyz format. Specifically, it reads:

```
number_of_atoms
frame_label
type_1 x1 y1 z1
...
number_of_atoms
frame_label
type_1 x1 y1 z1
...
```

- The first line: `number_of_atoms` is the number of atoms for one frame.
- The second line: `frame_label` is the frame label (starting from 0) within a run.
- The third line: `type_1` is the atom type for atom 1 and `x1 y1 z1` are the position components (in units of Ångstrom) for this atom.
- Then there are `number_of_atoms - 1` additional lines for the current frame.
- Then the pattern above is repeated.

5.3.12 omega2.out

This file contains the squared frequencies $\omega^2(\boldsymbol{k})$ at different $\boldsymbol{k}$-points.

File format

There are `N_kpoints` rows and `3 * N_basis` columns, where `N_basis` is the number of basis atoms in the unit cell defined in the *basis.in input file* and `N_kpoints` is the number of $\boldsymbol{k}$-points in the *kpoints.in input file*.

Each line corresponds to one $\boldsymbol{k}$-point. Each line contains `3 * N_basis` numbers, corresponding to the `3 * N_basis` values for $\omega^2(\boldsymbol{k})$ (in units of THz^2) in the order of increasing magnitude.

5.3.13 restart.xyz

This is the restart file. It is generated when invoking the *dump_restart keyword*.

File format

This file has the same format as the *simulation model input file*.

- The output mode for this file is overwrite.
- By renaming (or copying) this file to `model.xyz` one can restart a simulation.

5.3.14 dump.xyz

File containing atomistic positions, velocities and forces. It is generated when invoking the *dump_exyz keyword*.

File format

This file is in the extended XYZ format. The output mode for this file is append.

5.3.15 observer.xyz

File containing atomistic positions, velocities and forces. It is generated when invoking the *dump_observer keyword*.

- In the case of *dump_observer* being in *observe* mode, an XYZ-file for each potential will be written with the potential index in the *run.in*-file appended to the filename. For example, *observer0.xyz* corresponding to the first NEP potential specified in *run.in*, *observer1.xyz* to the second, and so forth.
- In the case of *mode* being *average*, only a single file will be written, corresponding to the average of the supplied NEP potentials.

File format

This file is in the extended XYZ format. The output mode for this file is append.

5.3.16 observer.out

This file contains the global thermodynamic quantities sampled at a given frequency, for each of the specified potentials. This file is generated when the *dump_observer keyword* is invoked, which also controls the frequency of the output.

- If *mode* in *dump_observer* is set to *observe*, then one file will be written for each of the *N* specified potentials, with the name *observer*.out*.
- If *mode* in *dump_observer* is set to *average*, then only a single file will be written, correspond to the thermodynamic quantities computed with the average potential.

Refer to *thermo.out* for the format of this file.

5.3.17 dipole.out

This file contains the global thermodynamic quantities sampled at a given frequency, for each of the specified potentials. This file is generated when the *dump_dipole keyword* is invoked, which also controls the frequency of the output.

File format

The output mode for this file is append. The file format is as follows:

```
column    1         2    3    4
quantity  time_step mu_x mu_y mu_z
```

5.3.18 polarizability.out

This file contains the global thermodynamic quantities sampled at a given frequency, for each of the specified potentials. This file is generated when the *dump_polarizability keyword* is invoked, which also controls the frequency of the output.

File format

The output mode for this file is append. The file format is as follows:

```
column    1         2    3    4    5    6    7
quantity  time_step p_xx p_yy p_zz p_xy p_yz p_zx
```

5.3.19 active.xyz

File containing atomistic positions, velocities, forces and uncertainties for structures written during on-the-fly active learning. It is generated when invoking the *active* keyword. Only structures with uncertainty exceeding the threshold δ will be written; thus, if no such structure is encountered during the MD simulation, this file will be missing.

File format

This file is in the extended XYZ format. The output mode for this file is append.

5.3.20 active.out

This file contains the simulation time t and uncertainty σ_f for each step of the MD simulation that has been checked during active learning. This file is generated when the *active keyword* is invoked, which also controls the frequency with which to check the uncertainty.

Note that the time and uncertainty will be written to this file regardless of if the structure exceeds the threshold δ.

File format

There are two columns in this file:

```
column   1 2
quantity t s
```

where the first column is the time in fs, and the second is the observed uncertainty in eV/Å.

5.3.21 sdc.out

This file contains the velocity autocorrelation (*VAC*) and self diffusion coefficient (*SDC*). It is generated when invoking the *compute_sdc keyword*.

File format

The data in this file are organized as follows:

- column 1: correlation time (in units of ps)
- column 2: VAC (in units of Å^2/ps^2) in the x direction
- column 3: VAC (in units of Å^2/ps^2) in the y direction
- column 4: VAC (in units of Å^2/ps^2) in the z direction
- column 5: SDC (in units of Å^2/ps) in the x direction
- column 6: SDC (in units of Å^2/ps) in the y direction
- column 7: SDC (in units of Å^2/ps) in the z direction

Only the group selected via the arguments of the *compute_sdc_keyword* is included in this output.

5.3.22 `msd.out`

This file contains the mean-square displacement (*MSD*) and self diffusion coefficient (*SDC*). It is generated when invoking the *compute_msd keyword*.

File format

The data in this file are organized as follows:

- column 1: correlation time (in units of ps)
- column 2: MSD (in units of Å^2) in the x direction
- column 3: MSD (in units of Å^2) in the y direction
- column 4: MSD (in units of Å^2) in the z direction
- column 5: SDC (in units of Å^2/ps) in the x direction
- column 6: SDC (in units of Å^2/ps) in the y direction
- column 7: SDC (in units of Å^2/ps) in the z direction

Only the group selected via the arguments of the *compute_msd_keyword* is included in this output.

In the case of `all_groups` having been specified, a set of columns ordered like above will be written for each group in the selected grouping method. For example, in a system with three groups, a total of 19 columns will be written. The first column is the time, columns 2-7 are the *MSD* and *SDC* for the first group, columns 8-13 for the second group, and columns 14-19 for group 3.

5.3.23 `shc.out`

This file contains the non-equilibrium virial-velocity correlation function $K(t)$ and the spectral heat current (*SHC*) $J_q(\omega)$, in a given direction, for a group of atoms, as defined in Eq. (18) and the left part of Eq. (20) of [Fan2019]. It is generated when invoking the *compute_shc keyword*.

File format

For each run, there are 3 columns and `2*Nc-1 + num_omega` rows. Here, `Nc` is the number of correlation steps and `num_omega` is the number of frequency points.

In the first `2*Nc-1` rows:

- column 1: correlation time t from negative to positive, in units of ps
- column 2: $K^{\rm in}(t)$ in units of Å eV/ps
- column 3: $K^{\rm out}(t)$ in units of Å eV/ps

$K^{\rm in}(t) + K^{\rm out}(t) = K(t)$ is exactly the expression in Eq. (18) of [Fan2019]. The in-out decomposition follows the definition in [Fan2017], which is useful for 2D materials but is not necessary for 3D materials.

In the next `num_omega` rows:

- column 1: angular frequency ω in units of THz
- column 2: $J_q^{\rm in}(\omega)$ in units of Å eV/ps/THz
- column 3: $J_q^{\rm out}(\omega)$ in units of Å eV/ps/THz

$J_q^{\rm in}(\omega) + J_q^{\rm out}(\omega) = J_q(\omega)$ is exactly the left expression in Eq. (20) of [Fan2019].

Only the potential part of the heat current has been included.

If :attr: 'group_id' is -1, then the file follows the above rules and will contain $K(t)$ and $J_q(\omega)$ for each group id except for group id 0. And the contents of the :attr: 'group_id' are arranged from smallest to largest.

5.3.24 thermo.out

This file contains the global thermodynamic quantities sampled at a given frequency. The frequency of the output is controlled via the *dump_thermo keyword*.

File format

There are 18 columns in this output file, each containing the values of a quantity at increasing time points:

```
column   1 2 3 4  5  6  7   8   9   10 11 12 13 14 15 16 17 18
quantity T K U Pxx Pyy Pzz Pyz Pxz Pxy ax ay az bx by bz cx cy cz
```

- `T` is the temperature (in units of K)
- `K` is the kinetic energy (in units of eV) of the system
- `U` is the potential energy (in units of eV) of the system
- `Pxx` is the pressure (in units of GPa) in the xx direction
- `Pyy` is the pressure (in units of GPa) in the yy direction
- `Pzz` is the pressure (in units of GPa) in the zz direction
- `Pyz` is the pressure (in units of GPa) in the yz direction
- `Pxz` is the pressure (in units of GPa) in the xz direction
- `Pxy` is the pressure (in units of GPa) in the xy direction
- `ax ay az bx by bz cx cy cz` are the components (in units of Ångstrom) of the triclinic box matrix formed by the following vectors:

$$\boldsymbol{a} = a_x \boldsymbol{e}_x + a_y \boldsymbol{e}_y + a_z \boldsymbol{e}_z$$
$$\boldsymbol{b} = b_x \boldsymbol{e}_x + b_y \boldsymbol{e}_y + b_z \boldsymbol{e}_z$$
$$\boldsymbol{c} = c_x \boldsymbol{e}_x + c_y \boldsymbol{e}_y + c_z \boldsymbol{e}_z$$

Caveats

- The data in this file are also valid for PIMD-related runs, but note that in this case the output temperature is just the target one. The energy and pressure contain the virial-estimator contributions.

5.3.25 velocity.out

This file contains the velocities of the atoms sampled at a given frequency. It produced when invoking the *dump_velocity keyword*.

File format

```
vx_1(1) vy_1(1) vz_1(1)
vx_2(1) vy_2(1) vz_2(1)
...
vx_N(1) vy_N(1) vz_N(1)
vx_1(2) vy_1(2) vz_1(2)
vx_2(2) vy_2(2) vz_2(2)
...
vx_N(2) vy_N(2) vz_N(2)
...
```

- There are three columns, corresponding to the x, y, and z components of the velocity.

- Each `N` consecutive lines correspond to one frame at a given time point.
- The number of lines equals the number the number of frames times `N`.
- `vx_m(n)` is the x component of the `m`-th atom in the `n`-th frame.
- `vy_m(n)` is the y component of the `m`-th atom in the `n`-th frame.
- `vz_m(n)` is the z component of the `m`-th atom in the `n`-th frame.
- Velocity components are in units of Å/fs.

5.3.26 `viscosity.out`

This file contains the stress auto-correlation function and the running viscosity from the Green-Kubo method. It is produced when invoking the *compute_viscosity keyword* in the *run.in input file*.

File format

This file reads

- column 1: correlation time (in units of ps)
- column 2: $\langle S_{xx}(0)S_{xx}(t)\rangle$ (in units of eV^2)
- column 3: $\langle S_{yy}(0)S_{yy}(t)\rangle$ (in units of eV^2)
- column 4: $\langle S_{zz}(0)S_{zz}(t)\rangle$ (in units of eV^2)
- column 5: $\langle S_{xy}(0)S_{xy}(t)\rangle$ (in units of eV^2)
- column 6: $\langle S_{xz}(0)S_{xz}(t)\rangle$ (in units of eV^2)
- column 7: $\langle S_{yz}(0)S_{yz}(t)\rangle$ (in units of eV^2)
- column 8: $\langle S_{yx}(0)S_{yx}(t)\rangle$ (in units of eV^2)
- column 9: $\langle S_{zx}(0)S_{zx}(t)\rangle$ (in units of eV^2)
- column 10: $\langle S_{zy}(0)S_{zy}(t)\rangle$ (in units of eV^2)
- column 11: $\eta_{xx} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{xx}(0)S_{xx}(t')\rangle dt'$ (in units of Pa s)
- column 12: $\eta_{yy} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{yy}(0)S_{yy}(t')\rangle dt'$ (in units of Pa s)
- column 13: $\eta_{zz} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{zz}(0)S_{zz}(t')\rangle dt'$ (in units of Pa s)
- column 14: $\eta_{xy} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{xy}(0)S_{xy}(t')\rangle dt'$ (in units of Pa s)
- column 15: $\eta_{xz} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{xz}(0)S_{xz}(t')\rangle dt'$ (in units of Pa s)
- column 16: $\eta_{yz} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{yz}(0)S_{yz}(t')\rangle dt'$ (in units of Pa s)
- column 17: $\eta_{yx} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{yx}(0)S_{yx}(t')\rangle dt'$ (in units of Pa s)
- column 18: $\eta_{zx} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{zx}(0)S_{zx}(t')\rangle dt'$ (in units of Pa s)
- column 19: $\eta_{zy} = \frac{1}{k_{\rm B}TV}\int_0^t \langle S_{zy}(0)S_{zy}(t')\rangle dt'$ (in units of Pa s)

Note:

- The shear viscosity can be calculated as $\eta_{\rm S} = \frac{1}{3}\left(\eta_{xy} + \eta_{xz} + \eta_{yz}\right)$.
- The longitudinal viscosity can be calculated as $\eta_{\rm L} = \frac{1}{3}\left(\eta_{xx} + \eta_{yy} + \eta_{zz}\right)$.
- The bulk viscosity $\eta_{\rm B}$ can be calculated from $\eta_{\rm B} + \frac{4}{3}\eta_{\rm S} = \eta_{\rm L}$.

- We have the following symmetric property, $\eta_{\alpha\beta} = \eta_{\beta\alpha}$, which should be confirmed by the output data.

5.3.27 `onsager.out`

This file contains some components of the running onsager coefficients tensor from the homogeneous non-equilibrium molecular dynamics Evans-Cummings algorithm (*HNEMDEC*) method. It is generated when invoking the *compute_hnemdec keyword*.

File format

If the driving force is in the μ (μ can be x, y, or z) direction and the dissipative flux is set as heat flux, for a system with M elements, this file reads:

- column 1: $L^{\mathrm{qq}}_{x\mu}(t)$ (in units of W/mK)
- column 2: $L^{\mathrm{qq}}_{y\mu}(t)$ (in units of W/mK)
- column 3: $L^{\mathrm{qq}}_{z\mu}(t)$ (in units of W/mK)
- column 4: $L^{\mathrm{1q}}_{x\mu}(t)$ (in units of $10^{-6}kg/smK$)
- column 5: $L^{\mathrm{1q}}_{y\mu}(t)$ (in units of $10^{-6}kg/s/m/K$)
- column 6: $L^{\mathrm{1q}}_{z\mu}(t)$ (in units of $10^{-6}kg/s/m/K$)
- ...
- column 3M+1: $L^{\mathrm{Mq}}_{x\mu}(t)$ (in units of $10^{-6}kg/smK$)
- column 3M+2: $L^{\mathrm{Mq}}_{y\mu}(t)$ (in units of $10^{-6}kg/smK$)
- column 3M+3: $L^{\mathrm{Mq}}_{z\mu}(t)$ (in units of $10^{-6}kg/smK$)

If the dissipative flux is changed to momentum flux of component α, this file reads:

- column 1: $L^{q\alpha}_{x\mu}(t)$ (in units of $10^{-6}kg/smK$)
- column 2: $L^{q\alpha}_{y\mu}(t)$ (in units of $10^{-6}kg/smK$)
- column 3: $L^{q\alpha}_{z\mu}(t)$ (in units of $10^{-6}kg/smK$)
- column 4: $L^{1\alpha}_{x\mu}(t)$ (in units of $10^{-12}kgs/m^3K$)
- column 5: $L^{1\alpha}_{y\mu}(t)$ (in units of $10^{-12}kgs/m^3K$)
- column 6: $L^{1\alpha}_{z\mu}(t)$ (in units of $10^{-12}kgs/m^3K$)
- ...
- column $3M+1$: $L^{M\alpha}_{x\mu}(t)$ (in units of $10^{-12}kgs/m^3K$)
- column $3M+2$: $L^{M\alpha}_{y\mu}(t)$ (in units of $10^{-12}kgs/m^3K$)
- column $3M+3$: $L^{M\alpha}_{z\mu}(t)$ (in units of $10^{-12}kgs/m^3K$)

Both the potential part and the kinetic part of the heat current have been considered. One can obtain the onsager coefficent Λ^{ml}_{ij} by:

$$\Lambda^{ml}_{ij} = T^2 L^{ml}_{ij}$$

The thermal conductivity can be derived from the onsager matrix Λ, that is:

$$\kappa = \frac{1}{T^2(\Lambda^{-1})_{00}}$$

5.3.28 rdf.out

This file contains the radial distribution function (*RDF*). It is generated when invoking the *compute_rdf keyword*.

File format

The data in this file are organized as follows:

- column 1: radius (in units of Å)
- column 2: The (*RDF*) for the whole system
- column 3 and more: The (*RDF*) for specific atom pairs if specified

5.3.29 adf.out

This file contains the Angular Distribution Function (*ADF*). It is generated when the *compute_adf keyword* is used.

File Format

For global ADF, the data in this file are organized as follows:

- Column 1: Angles (in degrees)
- Column 2: The (*ADF*) for the entire system

For local ADF, the data are organized as follows:

- Column 1: Angles (in degrees)
- The next Ntriples columns: The (*ADF*) for each specific atom triple

5.3.30 mcmd.out

This file contains some data related to (*MC*) simulation. It is generated when invoking the *mc keyword*.

File format

The data in this file are organized as follows:

- column 1: number of *MD* steps
- column 2: acceptance ratio of the *MC* trials
- column 3 and more: species concentrations in the specified order

5.3.31 lsqt_dos.out

This file contains the electronic density of states (*DOS*) from linear-scaling quantum transport (*LSQT*) calculations. It is produced when invoking the *compute_lsqt keyword* in the *run.in input file*.

File format

- Each row contains the *DOS* values (in units of states/atom/eV) for the specified energies.
- The number of columns equals the number of energy points.
- Different rows are from different time points in the *MD* simulation (supposed to be an equilibrium one), which can be averaged.

5.3.32 lsqt_velocity.out

This file contains the electron group velocity from linear-scaling quantum transport (*LSQT*) calculations. It is produced when invoking the *compute_lsqt keyword* in the *run.in input file*.

File format

- Each row contains the electron group velocity values (in units of m/s) for the specified energies.
- The number of columns equals the number of energy points.
- Different rows are from different time points in the *MD* simulation (supposed to be an equilibrium one), which can be averaged.
- Data within band gaps are not reliable and should not be trusted.

5.3.33 lsqt_sigma.out

This file contains the running electrical conductivity $\Sigma(E, t)$ as a function of energy E and correlation time t, from linear-scaling quantum transport (*LSQT*) calculations. It is produced when invoking the *compute_lsqt keyword* in the *run.in input file*.

File format

- The number of columns equals the number of energy points. The energy values can be inferred from the parameters to the *compute_lsqt keyword*.
- The number of rows equals the product of the number of *MD* steps for one run and the number of independent runs (supposed to be equilibrium ones), and the results from different runs can thus be averaged to reduce the statistical uncertainties.
- The electrical conductivity values are in units of S/m.

5.3.34 angular_rdf.out

This file contains the angular-dependent radial distribution function (*ARDF*) data.

File Format

The file has the following columns:

1. `radius`: The radial distance r (in Å)
2. `theta`: The angle θ (in radians, from -π to π)
3. `total`: The total ARDF $g(r, \theta)$ for all atom pairs
4. `type_i_j`: The partial ARDF $g(r, \theta)$ for atom pairs of type i and j (if specified)

For each radius value, there will be multiple rows corresponding to different angle values.

Example

Here is an example of the file content:

```
#radius theta total type_0_0 type_1_1 type_0_1
0.05 -3.14159 0.00000 0.00000 0.00000 0.00000
0.05 -3.07959 0.00000 0.00000 0.00000 0.00000
...
```

5.3.35 `orientorder.out`

This file stores the **Steinhardt order parameters**. It is automatically generated when the *compute_orientorder keyword* is invoked.

File format

The data are written in a repeating block structure:

- **First line**: the current simulation step.
- **Second line**: column headers indicating the computed degrees. For example, if degrees 4 and 6 are calculated together with their w_l versions, the column names will be: `ql4 ql6 wl4 wl6`
- **Subsequent lines**: per-atom order parameter values for the given step.

CHAPTER

SIX

NEP EXECUTABLE

6.1 Input files

To run a *NEP* construction using the `nep` executable, one has to prepare three input files that respectively specificy the parameters of the *NEP model* (`nep.in`), the *training dataset* (`train.xyz`), and the *test dataset* (`test.xyz`).

6.1.1 `nep.in`

This file specifies hyperparameters used for training neuroevolution potential (*NEP*) models, the functional form of which is outline *here*. The *NEP* approach was proposed in [Fan2021] (NEP1) and later improved in [Fan2022a] (NEP2), [Fan2022b] (NEP3), and [Song2024] (NEP4). Currently, we support NEP3 and NEP4, which can be chosen by the *version keyword*.

File format

In this input file, blank lines and lines starting with # are ignored. One can thus write comments after #.

All other lines need to be of the following form:

```
keyword parameter_1 parameter_2 ...
```

Keywords can appear in any order with the exception of the *type_weight keyword*, which cannot appear before the *type keyword*.

The *type keyword* does not have default parameters and *must* be set. All other keywords have default values.

Keywords

Keyword	Brief description
version	select the NEP version
type	number of atom types and list of chemical species
type_weight	force weights for different atom types
model_type	select to train potential, dipole, or polarizability
prediction	select between training and prediction (inference)
zbl	outer cutoff for the universal *ZBL* potential [Ziegler1985]
cutoff	radial ($r_{\rm c}^{\rm R}$) and angular ($r_{\rm c}^{\rm A}$) cutoffs
n_max	size of radial ($n_{\rm max}^{\rm R}$) and angular ($n_{\rm max}^{\rm A}$) basis
basis_size	number of radial ($N_{\rm bas}^{\rm R}$) and angular ($N_{\rm bas}^{\rm A}$) basis functions
l_max	expansion order for angular terms
neuron	number of neurons in the hidden layer ($N_{\rm neu}$)
lambda_1	weight of $\mathcal{L}_1$-norm regularization term
lambda_2	weight of $\mathcal{L}_2$-norm regularization term
lambda_e	weight of energy loss term
lambda_f	weight of force loss term
lambda_v	weight of virial loss term
atomic_v	fit atomic or global virial
force_delta	bias term that can be used to make smaller forces more accurate
batch	batch size for training
population	population size used in the *SNES* algorithm [Schaul2011]
generation	number of generations used by the *SNES* algorithm [Schaul2011]

Example

Here is an example `nep.in` file using all the default parameters:

```
type          2 Te Pb # this is a mandatory keyword
version       4       # default
cutoff        8 4     # default
n_max         4 4     # default
basis_size    8 8     # default
l_max         4 2 0   # default
neuron        30      # default
lambda_e      1.0     # default
lambda_f      1.0     # default
lambda_v      0.1     # default
batch         1000    # default
population    50      # default
generation    100000  # default
```

The NEP tutorial illustrates the construction of a *NEP* model. More examples can be found in this repository.

6.1.2 `train.xyz` and `test.xyz`

The `train.xyz` file, which contains the training data for the construction of a *NEP* model, and the `test.xyz` file, which contains the corresponding test data, both need to be provided in extended xyz file format. Each structure (or configuration or frame) occupies $N + 2$ lines, where N is the number of atoms in the structure.

Format for a single structure

Line 1

The first line should only contain one field, which is the number of atoms in the structure N.

Line 2

This line consists of a number of `keyword=value` pairs separated by spaces. Spaces before and after = are allowed. All the characters are case-insensitive. `value` can be a single item or a number of items enclosed by double quotes, such as `keyword="value_1 value_2 value_3"`. Here, the different values are separated by spaces and spaces after the left " and before the right " are allowed. For example, one can write `keyword=" value_1 value_2 value_3 "`.

Essentially any keyword is allowed, but we only read the following ones:

- `lattice="ax ay az bx by bz cx cy cz"` is mandatory and gives the cell vectors:

$$\boldsymbol{a} = a_x \boldsymbol{e}_x + a_y \boldsymbol{e}_y + a_z \boldsymbol{e}_z$$
$$\boldsymbol{b} = b_x \boldsymbol{e}_x + b_y \boldsymbol{e}_y + b_z \boldsymbol{e}_z$$
$$\boldsymbol{c} = c_x \boldsymbol{e}_x + c_y \boldsymbol{e}_y + c_z \boldsymbol{e}_z$$

- `energy=energy_value` such as `energy=-123.4` is mandatory and gives the target energy of the structure, which is -123.4 eV in this example.
- `virial="vxx vxy vxz vyx vyy vyz vzx vzy vzz"` is optional and gives the 3×3 virial tensor of the structure in eV. Positive and negative values represent compressed and stretched states, respectively.
- `stress="sxx sxy sxz syx syy syz szx szy szz"` is optional and gives the 3×3 stress tensor of the structure in eV/Å^3. Positive and negative values represent stretched and compressed states, respectively. If both `virial` and `stress` are present the former is used.
- `weight=relative_weight` is optional and gives the relative weight for the current structure in the total loss function.
- `properties=property_name:data_type:number_of_columns` is mandatory but only read the following items:
  - `species:S:1` chemical symbol in the periodic table (case-sensitive)
  - `pos:R:3` position vector
  - `force:R:3` or `forces:R:3` target force vector
- If a dipole model is to be trained, energy, virial, stress, and force will be ignored and one should additionally provide `dipole="dx dy dz"`, which is the dipole vector of the structure.
- If a polarizability model is to be trained, energy, virial, stress, force, and dipole will be ignored and one should additionally provide `pol="pxx pxy pxz pyx pyy pyz pzx pzy pzz"`, which is the polarizability tensor of the structure.

Starting from line 3

Each line should contain the same number of items, which are determined by the `property` keywords on line 2.

Units

- Length and position are expected in units of Ångstrom.
- The energy is expected in units of eV.

- Forces are expected in units of eV/Å.
- Virials are expected in units of eV (such that the virial divided by the volume yields the stress).
- Dipole and polarizability can be in arbitrary units (such as the Hartree atomic units) as liked (and remembered) by the user.

Tips

- Periodic boundary conditions are always assumed for all directions in each configuration. When the box thickness in a direction is smaller than twice of the radial cutoff distance, the code will internally replicate the box in that direction.
- The minimal number of atoms in a configuration is 1. The user is responsible for choosing a sensible reference energy when preparing the energy data. But this is not crucial as the absolute energies are not relevant in the present context. However, because NEP training uses single precision, accuracy will be lost if any reference energy is smaller than -100 eV/atom. The code will give a warning message in this case.
- The energy and virial data refer to the total energy and virial for the system. They are not per-atom but per-cell quantities.

6.2 Input parameters

Below you can find a listing of keywords for the `nep.in` input file.

6.2.1 `version`

This keyword selects which *NEP* version should be used. The syntax is:

```
version <version_number>
```

Here, `<version_number>` must be an integer, which can be 3 or 4, corresponding to NEP3 and NEP4, respectively. The default is 4.

More information about the *NEP* formalism can be found *here*.

6.2.2 `prediction`

This keyword instructs **nep** to evaluate a model against a set of structures without starting an optimization. This requires a `nep.txt` file, in addition to the `nep.in` file, to be present. Note that only the structures in `train.xyz` are included in the prediction. The syntax is:

```
prediction <mode>
```

where `<mode>` must be an integer that can assume one of the following values.

Value	Mode
0	optimization mode (default)
1	prediction mode

6.2.3 `model_type`

This keyword allows one to specify the type of model that is being trained. The syntax is:

```
model_type <type_value>
```

where <type_value> must be an integer that can assume one of the following values.

Value	Type of model
0	potential (default)
1	dipole
2	polarizability

6.2.4 type

This is a *mandatory* keyword that specifies the chemical species, for which a model is to be constructed.

The syntax is:

```
type <number_of_species> [<species>]
```

where <number_of_species> must be an integer and [<species>] must be a list of <number_of_species> chemical symbols. The latter are case-sensitive and must correspond to species in the periodic table.

6.2.5 type_weight

This keyword allows one to specify different weights for different chemical species. These weights are employed in the calculation of the force term in the loss function. Different weights for different species can be useful if the number of atoms per species are very different, e.g., for a few impurity atoms embedded in a host.

The syntax is as follows:

```
type_weight [<weight>]
```

Here, [<weight>] must be a list of $N_{\rm typ}$ non-negative real numbers representing the relative force weights for the different atomic species. By default all species carry the same weight (1.0).

6.2.6 zbl

This keyword can be used to spline the pair potential to the universal *ZBL* potential [Ziegler1985] at short distances. This is useful, for example, for models to be used in simulations of ion irradiation or extreme compression, when the interatomic distances can become very short.

The syntax is as follows:

```
zbl <cutoff>
```

Here, <cutoff> is a real number that specifies the "outer" cutoff $r_{\rm c}^{\rm ZBL-outer}$, below which the *NEP* pair potential is being splined to the *ZBL* potential. The "inner" cutoff of the *ZBL* potential, below which value the pair interaction is completely given by the *ZBL* potential, is fixed to half of the outer cutoff, $r_{\rm c}^{\rm ZBL-inner} = r_{\rm c}^{\rm ZBL-outer}/2$, which we have empirically found to be a reasonable choice.

When this keyword is absent, the *ZBL* potential will not be enabled and the value of $r_{\rm c}^{\rm ZBL-outer}$ is irrelevant.

Permissible values are 1 Å $\leq r_{\rm c}^{\rm ZBL-outer} \leq$ 2.5 Å

One can also use flexible ZBL parameters by providing a *zbl.in* file in the working directory, in which case the <cutoff> parameter is still needed but will not be used. For a n-species system, there should be $n(n+1)/2$ lines in the *zbl.in* files. Each line represents a unique pair of species. Counting from 1, the order of the lines is 1-1, 1-2, …, 1-n, 2-2, 2-3, …, 2-n, n-n. For each pair of species, there are 10 parameters to be listed from left to right: the first

two are the inner and outer cutoff radii, respectively; the next 8 are the parameters a_1 to a_8 in the ZBL function $\phi(x) = a_1 e^{-a_2 x} + a_3 e^{-a_4 x} + a_5 e^{-a_6 x} + a_7 e^{-a_8 x}$.

6.2.7 use_typewise_cutoff_zbl

This keyword enables one to use typewise cutoff radii for the ZBL part of the *NEP* model. The syntax is:

```
use_typewise_cutoff_zbl [<factor>]
```

with one optional (dimensionless) parameter <factor> that defaults to 0.65.

If this keyword is present, the outer ZBL cutoff between two elements is the minimum between the global outer ZBL cutoff $r_{\rm outer}^{\rm ZBL}$ and <factor> times of the sum of the covalent radii of the two elements, and the inner ZBL cutoff is half of the outer one.

By default, this keyword is not in effect.

6.2.8 cutoff

This keyword enables one to specify the radial ($r_{\rm c}^{\rm R}$) and angular ($r_{\rm c}^{\rm A}$) cutoffs of the *NEP* model. The syntax is:

```
cutoff <radial_cutoff> <angular_cutoff>
```

where <radial_cutoff> and <angular_cutoff> correspond to $r_{\rm c}^{\rm R}$ and $r_{\rm c}^{\rm A}$, respectively. The cutoffs must satisfy the conditions 2.5 Å $\leq r_{\rm c}^{\rm A} \leq r_{\rm c}^{\rm R} \leq$ 10 Å.

The defaults are $r_{\rm c}^{\rm R}$ = 8 Å and $r_{\rm c}^{\rm A}$ = 4 Å. It can be computationally beneficial to use (possibly much) smaller $r_{\rm c}^{\rm R}$ but the default values should be reasonable in most cases.

6.2.9 use_typewise_cutoff

This keyword enables one to use typewise cutoff radii for the radial and angular descriptors of the *NEP* model. The syntax is:

```
use_typewise_cutoff [<radial_factor> <angular_factor>]
```

with two optional (dimensionless) parameters, <radial_factor> and <angular_factor>, which default to 2.5 and 2, respectively.

If this keyword is present, the radial cutoff between two elements is the minimum between the global radial cutoff $r_{\rm c}^{\rm R}$ and <radial_factor> times of the sum of the covalent radii of the two elements, and the angular cutoff between two elements is the minimum between the global angular cutoff $r_{\rm c}^{\rm A}$ and <angular_factor> times of the sum of the covalent radii of the two elements.

By default, this keyword is not in effect.

6.2.10 n_max

This keyword sets the number of radial ($n_{\rm max}^{\rm R}$) and angular ($n_{\rm max}^{\rm A}$) descriptor components as introduced in Sect. II.B and Eq. (2) of [Fan2022b]. The syntax is:

```
n_max <n_max_R> <n_max_A>
```

where n_max_R and n_max_A are $n_{\rm max}^{\rm R}$ and $n_{\rm max}^{\rm A}$, respectively. The two parameters must satisfy $0 \leq n_{\rm max}^{\rm R} \leq 12$ and $0 \leq n_{\rm max}^{\rm A} \leq 8$.

The default values of $n_{\rm max}^{\rm R} = 4$ and $n_{\rm max}^{\rm A} = 4$ are relatively small but typically yield high speed.

Note: These parameters should not be confused with $N_{\rm bas}^{\rm R}$ and $N_{\rm bas}^{\rm A}$, which are set via the *basis_size keyword*.

6.2.11 basis_size

It sets the number of basis functions that are used to build the radial and angular descriptor functions, see Sects. II.B and II.C as well as Eq. (3) in [Fan2022b]. The syntax is:

```
basis_size <N_bas_R> <N_bas_A>
```

where <N_bas_R> and <N_bas_A> set $N_{\rm bas}^{\rm R}$ and $N_{\rm bas}^{\rm A}$, respectively. The parameters must satisfy $0 \leq N_{\rm bas}^{\rm R} \leq 16$ and $0 \leq N_{\rm bas}^{\rm A} \leq 12$.

The default values of $N_{\rm bas}^{\rm R} = 8$ and $N_{\rm bas}^{\rm A} = 8$ are usually sufficient.

Note: These parameters should not be confused with $n_{\rm max}^{\rm R}$ and $n_{\rm max}^{\rm A}$, which are set via the *n_max keyword*.

6.2.12 l_max

It sets the maximum expansion order for the angular terms, see Sect. II.C of [Fan2022b]. The syntax is:

```
l_max <l_max_3b> {<l_max_4b> {<l_max_5b>}}
```

where l_max_3b, l_max_4b, and l_max_5b set the limits for three, four, and five-body terms, respectively. The latter two arguments are optional (as indicated by the curly brackets).

If there is one value $l_{\rm max}^{\rm 4b} = l_{\rm max}^{\rm 5b} = 0$. If there are two values $l_{\rm max}^{\rm 5b} = 0$.

$l_{\rm max}^{\rm 3b}$ can take values from 0 to 8, $l_{\rm max}^{\rm 4b}$ can be 0 or 2, and $l_{\rm max}^{\rm 5b}$ can be 0 or 1. It is also required to have $l_{\rm max}^{\rm 3b} \geq l_{\rm max}^{\rm 4b} \geq l_{\rm max}^{\rm 5b}$.

The default values are $l_{\rm max}^{\rm 3b} = 4$, $l_{\rm max}^{\rm 4b} = 2$, and $l_{\rm max}^{\rm 5b} = 0$.

6.2.13 neuron

This keyword sets the number of neurons in the hidden layer of the *NN*. The syntax is:

```
neuron <number_of_neurons>
```

where <number_of_neurons> corresponds to $N_{\rm neu}$ in the *NEP formalism* [Fan2022b]. Values must satisfy $1 \leq N_{\rm neu} \leq 120$.

The default value is 30, which is relatively small but can lead to relatively high speed. Larger values rarely lead to a notable improvement.

6.2.14 lambda_1

This keyword sets the weight λ_1 of the $\mathcal{L}_1$-norm regularization term in the *loss function*. The syntax is:

```
lambda_1 <weight>
```

Here, <weight> represents λ_1, which can be set to any non-negative values. The default value is $\lambda_1 = \sqrt{N}/1000$, where N is the total number of training parameters.

6.2.15 lambda_2

This keyword sets the weight λ_2 of the $\mathcal{L}_2$-norm regularization term in the *loss function*. The syntax is:

```
lambda_2 <weight>
```

Here, <weight> represents λ_2, which can be set to any non-negative values. The default value is $\lambda_2 = \sqrt{N}/1000$, where N is the total number of training parameters.

6.2.16 lambda_e

This keyword sets the weight λ_e of the loss term associated with the **energy** in the *loss function*. The syntax is:

```
lambda_e <weight>
```

Here, <weight> represents λ_e, which must satisfy $\lambda_e \geq 0$ and defaults to $\lambda_e = 1.0$.

It might be beneficial to use a two-step training scheme, where λ_e takes a small value (such as 0.1) in the first training and a large value (such as 10) in the second (restarting from the first). During the second training, the energy error might decrease appreciably, while the force and virial errors are not much affected.

6.2.17 lambda_f

This keyword sets the weight λ_f of the loss term associated with the **forces** in the *loss function*. The syntax is:

```
lambda_f <weight>
```

Here, <weight> represents λ_f, which must satisfy $\lambda_f \geq 0$ and defaults to $\lambda_f = 1.0$.

6.2.18 lambda_v

This keyword sets the weight λ_v of the loss term associated with the **virials** in the *loss function*. The syntax is:

```
lambda_v <weight>
```

Here, <weight> represents λ_v, which must satisfy $\lambda_v \geq 0$ and defaults to $\lambda_v = 0.1$.

6.2.19 atomic_v

This keyword sets the mode *atomic_v* of whether to fit atomic or global quantities for dipole (*model_type = 1*) or polarizability (*model_type = 2*). Only one of atomic and global can be fitted at a time. Fitting both simultaneously is not supported. For the virial tensor (*model_type = 0*), only the global model is supported. The syntax is:

```
atomic_v <mode>
```

where <mode> must be an integer that can assume one of the following values.

Value	Mode
0	fit global tensor (default)
1	fit atomic tensor

6.2.20 lambda_shear

This keyword sets the extra weight λ_s of the loss term associated with the *shear* virials in the *loss function*. The syntax is:

```
lambda_shear <weight>
```

Here, <weight> represents λ_s, which must satisfy $\lambda_s \geq 0$ and defaults to $\lambda_s = 1$. The weight for the shear virials is thus $\lambda_v \lambda_s$, where λ_v is set by the *lambda_v keyword*. We have tested that a value of $\lambda_s = 5$ is sometimes helpful to make the prediction of shear moduli more accurate.

6.2.21 force_delta

This keyword can be used to bias the loss function to put more emphasis on obtaining accurate predictions for smaller forces. The syntax is:

```
force_delta <delta>
```

where <delta> sets the parameter δ, which must satisfy $\delta \geq 0$ eV/A and defaults to $\delta = 0$ eV/A (i.e., no bias).

When $\delta = 0$ eV/A, the *loss term associated with the forces* is proportional to the *RMSE* of the forces:

$$\sqrt{\frac{1}{3N}\sum_{i=1}^{N}\left(\boldsymbol{F}_i^{\mathrm{NEP}}-\boldsymbol{F}_i^{\mathrm{tar}}\right)^2}$$

When $\delta > 0$ eV/Å, this expression is modified to read:

$$\sqrt{\frac{1}{3N}\sum_{i=1}^{N}\left(\boldsymbol{F}_i^{\mathrm{NEP}}-\boldsymbol{F}_i^{\mathrm{tar}}\right)^2\frac{1}{1+\|\boldsymbol{F}_i^{\mathrm{tar}}\|/\delta}}$$

In this case, a smaller δ implies a larger weight on smaller forces.

6.2.22 batch

This keyword sets the size of each batch used during the *optimization procedure*. The syntax is:

```
batch <batch_size>
```

Here, <batch_size> sets the batch size N_{bat}, which must satisfy $N_{\mathrm{bat}} \geq 1$ and defaults to $N_{\mathrm{bat}} = 1000$.

In principle one can train against the entire training set during every iteration of the optimization procedure (equivalent to N_{bat} being identical to the number of structures in the training set). It is, however, often beneficial for computational speed and potentially necessary for memory reasons to consider only a subset of the training data at any given iteration. N_{bat} sets the size of this subset.

Usually, training sets with more diverse structures require using larger batch sizes to achieve maximal accuracy. In many cases, a batch size between $N_{\mathrm{bat}} = 100$ and $N_{\mathrm{bat}} = 1000$ should be sufficient to achieve good results. If you have a powerful GPU (such as a Tesla A100), you can use a large batch size (such as $N_{\mathrm{bat}} = 1000$) or the full-batch ($N_{\mathrm{bat}} \geq$ number of configurations in the training set). If you use a small batch size for a powerful GPU, you will simply waste the GPU resources. If you have a weaker GPU, you can use a smaller batch size.

If you observe oscillations in the *RMSE* values for energies, forces, and virials even for generations $\gtrsim 10^5$, it is a sign that the batch size is too small.

6.2.23 population

This keyword sets the size of the population used by the *SNES* algorithm [Schaul2011]. The syntax is:

```
population <population_size>
```

Here, <population_size> sets the population size N_{pop}, which must satisfy $10 \leq N_{\mathrm{pop}} \leq 100$ and defaults to $N_{\mathrm{pop}} = 50$.

6.2.24 generation

This keyword sets the number of generations N_{gen} for the *SNES* algorithm [Schaul2011]. The syntax is:

```
generation <number_of_generations>
```

Here, <number_of_generations> sets $N_{\rm gen}$, which must satisfy $0 \leq N_{\rm gen} \leq 10^7$ and defaults to $N_{\rm gen} = 10^5$. For simple systems, $N_{\rm gen} = 10^4 \sim 10^5$ is enough. For more complicated systems, values in the range $N_{\rm gen} = 10^5 \sim 10^6$ can be required to obtain a well converged model.

6.2.25 save_potential

This keyword sets the number of of generations between writing a nep.txt checkpoint file. If <format> is set to 0 the output file name is formatted as nep_gen[generation].txt. If <format> is set to 1 the output file name is formatted as nep_y[year]_m[month]_d[day]_h[hour]_m[minute]_s[second]_generation[generation].txt. These model files can be used to monitor the training progress of your model. Additionally, if <save_restart> is set to 1 the *nep.restart file* is also written, following the naming format set by <format>. Note that the *nep.restart file* is the file that is required to continue training.

The syntax is:

```
save_potential <number_of_generations_between_save_potential> <format> <save_restart>
```

The default number of generations between saved model files is $N = 10^5$ and the output file names use the extended format.

6.2.26 output_descriptor

This keyword instructs **nep** to output the (normalized) descriptors for all the atoms in the train.xyz file. It is only in effect for the prediction mode (see the *prediction keyword*). The syntax is:

```
output_descriptor <mode>
```

where <mode> must be an integer that can assume one of the following values.

Value	Mode
0	not to output descriptors during prediction (default)
1	output per-structure descriptors during prediction
2	output per-atom descriptors during prediction

6.2.27 fine_tune

This keyword instructs **nep** to fine tune a model starting from a foundation model. The syntax is:

```
fine_tune <nep_model_file> <nep_restart_file>
```

where <nep_model_file> is the potential file for the foundation model and <nep_restart_file> is the restart file for the foundation model.

Currently, we provide one foundation model in GPUMD/potentials/nep/nep89_20250409.

For more details, see the following exemplary nep.in file:

```
# for prediction
#type       89 H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Sc Ti V Cr Mn Fe Co Ni␣
↪Cu Zn Ga Ge As Se Br Kr Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe Cs Ba La␣
↪Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Ac Th Pa␣
↪U Np Pu
```

(continues on next page)

(continued from previous page)

```
#prediction 1

# for fine-tuning
fine_tune nep89_20250409.txt nep89_20250409.restart
type <your types>

# These cannot be changed:
version    4
zbl        2
cutoff     6 5
n_max      4 4
basis_size 8 8
l_max      4 2 1
neuron     80

# These can be changed:
lambda_1   0
lambda_e   1
lambda_f   1
lambda_v   1
batch      5000
population 50
save_potential 1000 0
generation 5000
```

6.3 Output files

The nep executable produces several output files. The *loss.out file* is written in "append mode", while all other files are continuously overwritten.

The contents of the *energy_train.out*, *force_train.out*, *virial_train.out*, *stress_train.out*, *dipole_train.out*, and *polarizability_train.out* files are updated every 1000 steps, while the contents of the other output files are updated every 100 steps.

With the exception of the *nep.txt file*, the output files contain only numbers (no text) in matrix form. All the files are plain text files.

File name	Brief description
loss.out	loss function, regularization terms, and *RMSE* data as a function of generation
nep.txt	*NEP* potential parameters
nep.restart	restart file
energy_train.out	target and predicted energies for training data set
energy_test.out	target and predicted energies for test data set
force_train.out	target and predicted forces for training data set
force_test.out	target and predicted forces for test data set
virial_train.out	target and predicted virials for training data set
virial_test.out	target and predicted virials for test data set
stress_train.out	target and predicted stress values for training data set
stress_test.out	target and predicted stress values for test data set
dipole_train.out	target and predicted dipole values for training data set
dipole_test.out	target and predicted dipole values for test data set
polarizability_train.out	target and predicted polarizability values for training data set
polarizability_test.out	target and predicted polarizability values for test data set
descriptor.out	descriptor values for training data set in prediction mode

6.3.1 `loss.out`

This files contains the terms that enter the *loss function* for every 100-th generation.

If a potential model is trained, each row contains the following fields:

```
gen L_t L_1 L_2 L_e_train L_f_train L_v_train L_e_test L_f_test L_v_test
```

where

- `gen` is the current generation.
- `L_t` is the total loss function.
- `L_1` is the loss function related to the $\mathcal{L}_1$ regularization.
- `L_2` is the loss function related to the $\mathcal{L}_2$ regularization.
- `L_e_train` is the energy RMSE (in units of eV/atom) for the training set.
- `L_f_train` is the force RMSE (in units of eV/Å) for the training set.
- `L_v_train` is the virial RMSE (in units of eV/atom) for the training set.
- `L_e_test` is the energy RMSE (in units of eV/atom) for the test set.
- `L_f_test` is the force RMSE (in units of eV/Å) for the test set.
- `L_v_test` is the virial RMSE (in units of eV/atom) for the test set.

If a dipole model is trained, each row contains the following fields:

```
gen L_t L_1 L_2 L_mu_train L_mu_test
```

where

- `L_mu_train` is the dipole RMSE (per atom) for the training set.
- `L_mu_test` is the dipole RMSE (per atom) for the test set.

If a polarizability model is trained, each row contains the following fields:

```
gen L_t L_1 L_2 L_alpha_train L_alpha_test
```

where

- `L_alpha_train` is the polarizability RMSE (per atom) for the training set.
- `L_alpha_test` is the polarizability RMSE (per atom) for the test set.

6.3.2 nep.txt

This file contains the parameters of the *NEP* model generated by the training. It can be used by the `gpumd` executable to run *MD* simulations.

6.3.3 nep.restart

This file enables restarting an optimization run. If the file is present, training will start from the state saved in this file. The file is continuously updated during training.

The user does not need to understand the contents of this file. One must, however, ensure that the hyperparameters in the *nep.in input file* related to the descriptor are the same as those used to generate the restart file.

6.3.4 energy_*.out

The `energy_train.out` and `energy_test.out` files contain the predicted and target energies for the training and test sets, respectively. Each file contains 2 columns. The first column gives the energy in units of eV/atom calculated using the *NEP* model. The second column gives the corresponding target energies. Each row corresponds to the configuration at the same position in the *train.xyz (if using the full-batch) and test.xyz files*, respectively.

6.3.5 force_*.out

The `force_train.out` and `force_test.out` files contain the predicted and target force components of the configurations provided in the *train.xyz and test.xyz input files*.

There are 6 columns. The first three columns are the x, y, and z force components in units of eV/Å computed using the *NEP* model. The last three columns are the corresponding target forces. Each row corresponds to one atom.

6.3.6 virial_*.out

The `virial_train.out` and `virial_test.out` files contain the predicted and target virials.

There are N_c rows, where N_c is the number of configurations in the *train.xyz and test.xyz input files*.

There are 12 columns. The first 6 columns give the xx, yy, zz, xy, yz, and zx virial components calculated using the *NEP* model. The last 6 columns give the corresponding target virials.

For a structure without target virial or stress, a target value of -1e6 will be output to remind the user about this.

The virial values are in units of eV/atom.

6.3.7 stress_*.out

The `stress_train.out` and `stress_test.out` files contain the predicted and target stress components.

There are N_c rows, where N_c is the number of configurations in the *train.xyz and test.xyz input files*.

There are 12 columns. The first 6 columns give the xx, yy, zz, xy, yz, and zx stress components calculated using the *NEP* model. The last 6 columns give the corresponding target stress components.

For a structure without target virial or stress, a target value of -1e6 will be output to remind the user about this.

The stress values are in units of GPa.

6.3.8 dipole_*.out

The `dipole_train.out` and `dipole_test.out` files contain the predicted and target dipole values.

There are N_c rows, where N_c is the number of configurations in the *train.xyz and test.xyz input files*.

There are 6 columns. The first 3 columns give the x, y, and z dipole components calculated using the *NEP* model. The last 3 columns give the corresponding target values.

The dipole values are normalized by the number of atoms.

6.3.9 polarizability_*.out

The `polarizability_train.out` and `polarizability_test.out` files contain the predicted and target polarizability values.

There are N_c rows, where N_c is the number of configurations in the *train.xyz and test.xyz input files*.

There are 12 columns. The first 6 columns give the xx, yy, zz, xy, yz, and zx polarizability components calculated using the *NEP* model. The last 6 columns give the corresponding target values.

The polarizability values are normalized by the number of atoms.

6.3.10 descriptor.out

The `descriptor.out` file contains the per-structure or per-atom descriptor values for the input `train.xyz` file.

There are N rows and N_des columns, where N is the number of structures or atoms in `train.xyz` and N_des is the dimension for the descriptor vector.

The row index is consistent with the global structure or atom index in the `train.xyz` file.

For each row, there are N_des descriptor components, arranged in a particular order (radial components, three-body angular components, four-body angular components, five-body angular components).

CHAPTER

SEVEN

CREDITS

TBC

7.1 Acknowledgments

- NSFC with project numbers 11404033 and 11974059
- Aalto Science-IT project
- Finland's IT Center for Science (CSC)

CHAPTER

EIGHT

BIBLIOGRAPHY

CHAPTER

NINE

PUBLICATIONS

9.1 2025

1. Yunjia Bao, Tao Chen, Zhuo Miao, Weidong Zheng, Puqing Jiang, Kunfeng Chen, Ruiqiang Guo, and Dongfeng Xue. Machine-learning-assisted understanding of depth-dependent thermal conductivity in lithium niobate induced by point defects. *Advanced Electronic Materials*, pages 2400944, 2025. doi:10.1002/aelm.202400944.

2. Peng Bi, Yushen Wan, Yong Yi, and Songhu Bi. Impact of surface phonons on the thermal conductivity of triply periodic minimal surface silicon. *Phys. Rev. B*, 112:155404, Oct 2025. doi:10.1103/ctfs-fx4m.

3. Wengang Bu, Pengfei He, Jiamao Hao, Xiangyang Wang, Rong Wang, Zhenfeng Hu, Jinyong Mo, and Xiubing Liang. Exploring the formation and mechanical significance of short-range ordering in w-mo-v-based on neuroevolution potential. *Materials Today Physics*, pages 101854, 2025. doi:10.1016/j.mtphys.2025.101854.

4. Chenyang Cao, Liyi Bai, Shuo Cao, Ye Su, Yanzhou Wang, Zheyong Fan, and Ping Qian. Structural and transport properties of litfsi/g3 electrolyte with machine-learned molecular dynamics. *The Journal of Physical Chemistry C*, 129(28):13030–13039, 2025. doi:10.1021/acs.jpcc.5c02287.

5. Jian Cao, Chang Liu, Zian Chen, Haichao Li, Lina Xu, Hongping Xiao, Shun Wang, Xiao He, and Guoyong Fang. Accelerated discovery of refractory high-entropy alloys via interpretable machine learning. *The Journal of Physical Chemistry Letters*, pages 8806–8814, 2025. doi:10.1021/acs.jpclett.5c01616.

6. Shuo Cao, Ao Wang, Zheyong Fan, Hua Bao, Ping Qian, Ye Su, and Yu Yan. Lattice thermal conductivity of 16 elemental metals from molecular dynamics simulations with a unified neuroevolution potential. *Journal of Applied Physics*, 137(22):225101, 06 2025. doi:10.1063/5.0275820.

7. Fei Chen, Han Wang, Yanan Jiang, Lihua Zhan, and Youliang Yang. Development of a neuroevolution machine learning potential of al-cu-li alloys. *Metals*, 2025. doi:10.3390/met15010048.

8. Yuanyuan Chen, Zihao Song, Shuhan Lv, Libin Shi, and Ping Qian. Phase diagram and thermoelectric performance of lead-free perovskite using machine learning potentials and density functional theory. *Computational Materials Science*, 258:114015, 2025. doi:10.1016/j.commatsci.2025.114015.

9. Jia-Peng Dai, Xiang Liu, and Dong Li. Insight into structural-functional relationship for conductive-radiative heat transfer in multi-scale thermal insulating carbon aerogels. *Carbon*, 243:120467, 2025. doi:10.1016/j.carbon.2025.120467.

10. Davide Donadio, Margaret L. Berrens, Wanyu Zhao, Shunda Chen, and Tianshu Li. Metastability and ostwald step rule in the crystallisation of diamond and graphite from molten carbon. *Nature Communications*, 16(1):6324, 2025. doi:10.1038/s41467-025-61674-5.

11. Haikuan Dong, Yuqi Liu, Zihan Tan, Qing Li, Xiaoye Zhou, Shujun Zhou, and Xiaoming Xiu. Coherent heat transport in graphene grain boundary superlattices. *Physica Scripta*, 100(8):085975, aug 2025. doi:10.1088/1402-4896/adf400.

12. Peng-Hu Du, Qian Wang, Qiang Sun, and Puru Jena. Thermal rectification of sierpiński tetrahedron fractals assembled from supertetrahedral T_2-type tin selenide clusters. *Phys. Rev. B*, 111:L121406, Mar 2025. doi:10.1103/PhysRevB.111.L121406.

13. Sangita Dutta, Erik Fransson, Tobias Hainer, Benjamin M. Gallant, Dominik J. Kubicki, Paul Erhart, and Julia Wiktor. Revealing the low-temperature phase of $FAPbI_3$ using a machine-learned potential. *Journal of the American Chemical Society*, 2025. doi:10.1021/jacs.5c05265.

14. Mandi Fang, Yinqiao Zhang, Zheyong Fan, Daquan Tan, Xiaoyong Cao, Chunlei Wei, Nan Xu, and Yi He. Enhancing transferability of machine learning-based polarizability models in condensed-phase systems via atomic polarizability constraint. *npj Computational Materials*, 2025. doi:10.1038/s41524-025-01705-3.

15. Xiaokang Feng, Shuning Pan, Kento Katagiri, Jiuyang Shi, Jia Qu, Keita Nonaka, Cong Liu, Liang Sun, Pinwen Zhu, Norimasa Ozaki, Takayoshi Sano, Yuichi Inubushi, Kohei Miyanishi, Keiichi Sueda, Tadashi Togashi, Makina Yabashi, Toshinori Yabuuchi, Hirotaka Nakamura, Yoichiro Hironaka, Yuhei Umeda, Yusuke Seto, Takuo Okuchi, Jian Sun, Toshimori Sekine, and Wenge Yang. Nanosecond structural evolution in shocked coesite. *Science Advances*, 11(17):eads3139, 2025. doi:10.1126/sciadv.ads3139.

16. Xiaoqian Gao, Shu Lin, Chuankui Xiao, Jing Wan, Yilun Liu, and Huasong Qin. Achieving ultralow and highly isotropic thermal conductivity in coherent si3n4 ceramics heterostructure: a machine learning potential based molecular dynamics simulation study. *Thin-Walled Structures*, pages 113751, 2025. doi:10.1016/j.tws.2025.113751.

17. Zeyu Guo, Guanting Li, Mingyue Lv, and Jing Wan. Phonon thermal transport in diamond-graphene superlattices. *Diamond and Related Materials*, pages 112880, 2025. doi:10.1016/j.diamond.2025.112880.

18. Tobias Hainer, Erik Fransson, Sangita Dutta, Julia Wiktor, and Paul Erhart. A morphotropic phase boundary in $MA_{1-x}FA_xPbI_3$: linking structure, dynamics, and electronic properties. *Nature Communications*, pages 8775, 2025. doi:10.1038/s41467-025-64526-4.

19. Jinge Han, Jun Tang, Hehuan Bai, Yanru Guo, Haochen Tong, Zhigang Zang, and Ru Li. Lattice thermal conductivity of cssnbr3/cs2snbr6 interface from ab initio based neuroevolution potential simulations. *The Journal of Chemical Physics*, 163(4):044705, 07 2025. doi:10.1063/5.0275050.

20. Song Hu and Xiaokun Gu. Approaching to low thermal conductivity limit in layered materials through full-spectrum phonon band engineering. *Materials Today Physics*, pages 101669, 2025. doi:10.1016/j.mtphys.2025.101669.

21. Zhi-Qiang Hu, Yi-Fan Xie, Rui Liu, Jian-Li Shao, and Peng-Wan Chen. Prediction of reaction kinetics for cl-20 and host–guest crystals under high temperature and pressure using neuroevolution potential. *The Journal of Chemical Physics*, 162(17):174503, 05 2025. doi:10.1063/5.0258001.

22. Xinfang Jia, Yu Bao, Shuo Cao, Ye Su, and Ping Qian. Temperature effects on radiation damage in hcp-zirconium: a molecular dynamics study using a fine-tuned machine-learned potential. *Journal of Nuclear Materials*, 616:156025, 2025. doi:10.1016/j.jnucmat.2025.156025.

23. Wenwu Jiang, Ting Liang, Jianbin Xu, and Wengen Ouyang. Strain-engineered anisotropic thermal transport in layered MoS_2 structures. *ACS Appl. Mater. Interfaces*, 17(23):34833–34844, 2025. doi:10.1021/acsami.5c06264.

24. Dingyi Jin, Guo Wei, and Haidong Wang. Sensitivity of short-range order prediction to machine learning potential formalisms: a case study on nbmotaw high-entropy alloy. *Chinese Physics B*, 2025. doi:10.1088/1674-1056/ae039a.

25. Prakriti Kayastha, Erik Fransson, Paul Erhart, and Lucy Whalley. Octahedral tilt-driven phase transitions in $bazrs_3$ chalcogenide perovskite. *The Journal of Physical Chemistry Letters*, 16(8):2064–2071, 2025. doi:10.1021/acs.jpclett.4c03517.

26. Rasmus Lavén, Erik Fransson, Paul Erhart, Fanni Juranyi, Garrett E. Granroth, and Maths Karlsson. Unraveling the nature of vibrational dynamics in $cspbi_3$ by inelastic neutron scattering and molecular dynamics simulations. *J. Phys. Chem. Lett.*, pages 4812–4818, 2025. doi:10.1021/acs.jpclett.5c00778.

27. Ke Li and Hao Ma. Decoding the thermal conductivity of ionic covalent organic frameworks: optical phonons as key determinants revealed by neuroevolution potential. *Materials Today Physics*, 54:101724, 2025. doi:10.1016/j.mtphys.2025.101724.

28. Rui Li, Zijun Qi, Zhanpeng Sun, Biao Meng, Wei Shen, Zhaofu Zhang, and Gai Wu. Defect evolution in gallium oxide during stretching process: a molecular dynamics simulation. *Materials Science in Semiconductor Processing*, 192:109463, 2025. doi:10.1016/j.mssp.2025.109463.

29. Shun Li, Lantao Fang, Tao Liu, Xuping Wang, Bing Liu, Yuanyuan Zhang, Xianshun Lv, and Lei Wei. Machine learning-accelerated molecular dynamics calculations for investigating the thermal modulation by ferroelectric domain wall in ktn single crystals. *Computational Materials Science*, 249:113674, 2025. doi:10.1016/j.commatsci.2025.113674.

30. Wenlong Li, Yu Liu, Zhendong Li, Pei Zhang, Xinghua Li, and Tao Ouyang. Thermal transport properties of 2d narrow bandgap semiconductor ca3n2, ba3p2, and ba3as2: machine learning potential study. *Chinese Physics B*, 2025. doi:10.1088/1674-1056/ade5a0.

31. X.T. Li, R. Liu, J.P. Hou, Z.J. Zhang, and Z.F. Zhang. Trade-off model for strength-ductility relationship of metallic materials. *Acta Materialia*, pages 120942, 2025. doi:10.1016/j.actamat.2025.120942.

32. Zhendong Li, Longwei Han, Tao Ouyang, Juexian Cao, Yongshen Yao, and Xiaolin Wei. Thermal and mechanical properties of deep-ultraviolet light sources candidate materials $BeGeN_2$ by machine-learning molecular dynamics simulations. *Phys. Rev. Mater.*, 9:033804, Mar 2025. doi:10.1103/PhysRevMaterials.9.033804.

33. Zhendong Li, Tao Ouyang, Longwei Han, Zhunyun Tang, Xiaoxia Wang, Juexian Cao, Pei Zhang, Yongshen Yao, and Xiaolin Wei. Impact of high-order anharmonicity and nitrogen vacancy defects on the thermal conductivity of $MoSi_2N_4$. *Phys. Rev. B*, 112:064309, Aug 2025. doi:10.1103/gmh7-gsk8.

34. Junyu Lian, Shuping Jiao, Wanjian Yin, and Ke Zhou. Machine-learning potential molecular dynamics reveals the critical role of flexibility in solid–liquid nanofluidic friction. *ACS Nano*, 2025. doi:10.1021/acsnano.5c08363.

35. Nianjie Liang, Alfredo Fiorentino, and Bai Song. Thermal transport in amorphous carbon nanotubes. *Phys. Rev. B*, 112:094205, Sep 2025. doi:10.1103/klhj-x3f2.

36. Ting Liang, Wenwu Jiang, Ke Xu, Hekai Bu, Zheyong Fan, Wengen Ouyang, and Jianbin Xu. Pysed: a tool for extracting kinetic-energy-weighted phonon dispersion and lifetime from molecular dynamics simulations. *Journal of Applied Physics*, 138(7):075101, 08 2025. doi:10.1063/5.0278798.

37. Christopher Linderälv, Nicklas Österbacka, Julia Wiktor, and Paul Erhart. Optical line shapes of color centers in solids from classical autocorrelation functions. *npj Computational Materials*, 11(1):101, Apr 2025. doi:10.1038/s41524-025-01565-x.

38. Fachen Liu, Ruilin Mao, Zhiqiang Liu, Jinlong Du, and Peng Gao. Probing phonon transport dynamics across an interface by electron microscopy. *Nature*, 2025. doi:10.1038/s41586-025-09108-6.

39. Junlan Liu, Qian Yin, Mengshu He, and Jun Zhou. Constructing accurate machine learned potential and performing highly efficient atomistic simulation to predict structural and thermal properties: the case of Cu_7PS_6. *Computational Materials Science*, 251:113686, 2025. doi:10.1016/j.commatsci.2025.113686.

40. Shixian Liu, Ge Zhang, Fei Yin, A. A. Barinov, V. I. Khvesyuk, and Nuo Yang. Temperature dependence of specific heat capacity of nanostructures via neuroevolution machine-learned potential. *Journal of Applied Physics*, 138(10):104301, 09 2025. doi:10.1063/5.0284002.

41. Yiwen Liu, Yaming Fu, Fangchao Gu, Hulei Yu, Lei Zhuang, and Yanhui Chu. Lattice-distortion-driven reduced lattice thermal conductivity in high-entropy ceramics. *Advanced Science*, pages 2501157, 2025. doi:10.1002/advs.202501157.

42. Yu-Qi Liu, Hai-Kuan Dong, Ying Ren, Wei-Gang Zhang, and Wei Chen. Crystallization of h-bn by molecular dynamics simulation using a machine learning interatomic potential. *Computational Materials Science*, 249:113621, feb 2025. doi:10.1016/j.commatsci.2024.113621.

43. Yutao Liu, Tinghong Gao, Qingquan Xiao, Yunjun Ruan, Qian Chen, Bei Wang, and Jin Huang. Generalized modeling of carbon film deposition growth via hybrid md/mc simulations with machine-learning potentials. *npj Computational Materials*, 11(1):285, 2025. doi:10.1038/s41524-025-01781-5.

44. Zhoulin Liu, Jianchun Sha, Guang-Ling Song, Ziliang Wang, and Yinghe Zhang. Understanding magnesium dissolution through machine learning molecular dynamics. *Chemical Engineering Journal*, 516:163578, 2025. doi:10.1016/j.cej.2025.163578.

45. Chenchen Lu, Zhen Li, Xinxin Sang, Zheyong Fan, Xujun Xu, Yingyan Zhang, Ke Xu, Yanhua Cheng, Junhua Zhao, Jin-Cheng Zheng, and Ning Wei. Stress-driven grain boundary structural transition in diamond by machine learning potential. *Small*, pages 2409092, 2025. doi:10.1002/smll.202409092.

46. Chenchen Lu, Xujun Xu, Yanhua Cheng, Zheyong Fan, Zhen Li, Junhua Zhao, and Ning Wei. Interfacial thermal conductance at the gas-solid interface: microscopic energy transport mechanisms and the thermal rectification phenomenon. *International Communications in Heat and Mass Transfer*, 166:109153, 2025. doi:10.1016/j.icheatmasstransfer.2025.109153.

47. Wenzhu Luo, Ershuai Yin, Lei Wang, Wenlei Lian, Neng Wang, and Qiang Li. Heat transfer enhancement of n-ga-al semiconductors heterogeneous interfaces. *International Journal of Heat and Mass Transfer*, 244:126902, 2025. doi:10.1016/j.ijheatmasstransfer.2025.126902.

48. Shuang Lyu, Xingzhong Cao, Yanguang Zhou, and Yue Chen. Phonon scattering and thermal transport in pbse and medium entropy thermoelectric pbse0.5te0.25s0.25 with defects of different dimensions. *The Journal of Physical Chemistry Letters*, pages 5429–5434, 2025. doi:10.1021/acs.jpclett.5c00740.

49. Kaiqiang Ma, Lan Zhang, and Huizhong Ma. Machine learning potentials decode the thermophysical properties and two-stage phase transition in lanbo4. *Ceramics International*, 2025. doi:10.1016/j.ceramint.2025.07.328.

50. Jaeyun Moon and Zhiting Tian. Crystal-like thermal transport in amorphous carbon. *npj Computational Materials*, 11(1):1–8, 2025. doi:10.1038/s41524-025-01625-2.

51. Myung-Hoon Oh, Hyun-Seok Kim, and Seonho Cho. Effects of wide phononic bandgaps on thermal conductance in silicon nanomeshes. *The Journal of Physical Chemistry C*, Feb 2025. doi:10.1021/acs.jpcc.4c07490.

52. Niuchang Ouyang, Dongyi Shen, Chen Wang, Ruihuan Cheng, Qi Wang, and Yue Chen. Positive temperature-dependent thermal conductivity induced by wavelike phonons in complex ag-based argyrodites. *Phys. Rev. B*, 111:064307, Feb 2025. doi:10.1103/PhysRevB.111.064307.

53. Fengzijun Pan, Zhoulin Liu, Guangyao Li, Yuhui Zhang, Zepeng Fan, Yang Gang, and Dawei Wang. Hydration behavior of quartz surfaces revealed by molecular dynamics simulations with a novel machine learning potential. *The Journal of Physical Chemistry C*, 2025. doi:10.1021/acs.jpcc.5c03759.

54. Paolo Pegolo, Enrico Drigo, Federico Grasselli, and Stefano Baroni. Transport coefficients from equilibrium molecular dynamics. *The Journal of Chemical Physics*, 162(6):064111, 02 2025. doi:10.1063/5.0249677.

55. Zijun Qi, Rui Li, Kun Ma, Haoyuan Chen, Yunfei Song, Qijun Wang, Xiang Sun, Lijie Li, Sheng Liu, Wei Shen, Dekun Yang, and Gai Wu. In-depth insights into crystal orientation and temperature dependence of interfacial thermal transport in diamond/sic heterostructures by machine learning molecular dynamics. *International Communications in Heat and Mass Transfer*, 166:109231, 2025. doi:10.1016/j.icheatmasstransfer.2025.109231.

56. Zhijia Qin, Shirong Liang, Linli Zhu, Penghua Ying, Dongfeng Li, and Ligang Sun. Developing a machine learning ni-co-al ternary interatomic potential for temperature and strain-rate dependent mechanical behaviors of ni-based single crystal superalloys. *Materials & Design*, pages 114535, 2025. doi:10.1016/j.matdes.2025.114535.

57. Ariana Quek, Niuchang Ouyang, Hung-Min Lin, Olivier Delaire, and Johann Guilleminot. Enhancing robustness in machine-learning-accelerated molecular dynamics: a multi-model nonparametric probabilistic approach. *Mechanics of Materials*, pages 105237, 2025. doi:10.1016/j.mechmat.2024.105237.

58. Petter Rosander, Erik Fransson, Nicklas Österbacka, Paul Erhart, and Göran Wahnström. Untangling the raman spectra of cubic and tetragonal $BaZrO_3$. *Phys. Rev. B*, 111:064107, Feb 2025. doi:10.1103/PhysRevB.111.064107.

59. Alireza Seifi, Mahyar Ghasemi, Movaffaq Kateb, and Pirooz Marashi. Challenges in determining the thermal conductivity of core–shell nanowires by atomistic simulation. *The Journal of Chemical Physics*, 162(12):124706, 03 2025. doi:10.1063/5.0246759.

60. Shuyue Shan, Zhongwei Zhang, Shuang Lu, Sebastian Volz, and Jie Chen. Generation of interfacial phonon modes and their contribution to thermal transport across the gan/zno interface. *Phys. Rev. B*, 112:155302, Oct 2025. doi:10.1103/ym8w-gwr1.

61. GuoFa Shen, Tinghong Gao, Qian Chen, Qingquan Xiao, Bei Wang, Jin Huang, Yunjun Ruan, Shipeng Zhang, and Shuang Li. Neuroevolution potential-driven accurate and efficient discovery of mechanical behavior and nanocluster dynamics in tial alloys. *J. Mater. Chem. C*, pages –, 2025. doi:10.1039/D5TC02960K.

62. Dan C. Sorescu, Terumasa Tadano, and Wissam A. Saidi. Theoretical investigations of anharmonic effects and phonon transport in the cubic phase of crystalline perovskite cspbcl3. *The Journal of Physical Chemistry C*, 2025. doi:10.1021/acs.jpcc.5c02871.

63. Xuhui Sun, Shuang Lu, Shuyue Shan, Zhongwei Zhang, and Jie Chen. Origin of superdiffusive thermal transport in one-dimensional van der waals atomic chains. *Phys. Rev. B*, 111:205404, May 2025. doi:10.1103/PhysRevB.111.205404.

64. Zhanpeng Sun, Yunfei Song, Zijun Qi, Xiang Sun, Meiyong Liao, Rui Li, Qijun Wang, Lijie Li, Gai Wu, Wei Shen, and Sheng Liu. Heat transport exploration through the gan/diamond interfaces using machine learning potential. *International Journal of Heat and Mass Transfer*, 241:126724, 2025. doi:10.1016/j.ijheatmasstransfer.2025.126724.

65. Zihan Tan, Shuo Wang, Yuqi Liu, Yang Xiao, Xiaoye Zhou, Shujun Zhou, Xiaoming Xiu, and Haikuan Dong. Coherent and incoherent phonon transport in graphene/h-bn superlattice: a machine learning potential. *Physica E: Low-dimensional Systems and Nanostructures*, pages 116259, 2025. doi:10.1016/j.physe.2025.116259.

66. Xiao Tang, Liangcai Wu, Ziang Xu, Lei Liu, Zhitang Song, and Wenxiong Song. Thermal conductivity of selenium crystals based on machine learning potentials. *Physical Chemistry Chemical Physics*, 2025. doi:10.1039/D5CP02310F.

67. Zhunyun Tang, Xiaoxia Wang, Jin Li, Chaoyu He, Mingxing Chen, Chao Tang, and Tao Ouyang. Weak s-d orbital bonding induces strong phonon anharmonicity and unconventional mass-dependent behavior of lattice thermal conductivity in $CaTiF_6$. *Phys. Rev. Mater.*, 9:093401, Sep 2025. doi:10.1103/ybjz-t8q1.

68. Nutth Tuchinda and Christopher A. Schuh. Grain boundary segregation spectra from a generalized machine-learning potential. *Scripta Materialia*, 264:116682, 2025. doi:10.1016/j.scriptamat.2025.116682.

69. Lilian M. Vogl, Shunda Chen, Peter Schweizer, Xiaochen Jin, Shui-Qing Yu, Jifeng Liu, Tianshu Li, and Andrew M. Minor. Identification of short-range ordering motifs in semiconductors. *Science*, pages 1342–1346, 2025. doi:10.1126/science.adu0719.

70. Bing Wang, Kaiqi Li, Weiming Zhang, Yuqi Sun, Jian Zhou, and Zhimei Sun. Thermal transport of GeTe/Sb_2Te_3 superlattice by large-scale molecular dynamics with machine-learned potential. *The Journal of Physical Chemistry C*, 129(13):6386–6396, 2025. doi:10.1021/acs.jpcc.4c07570.

71. Bing Wang, Zenan Shi, Penghua Ying, Libo Li, and Jin Zhang. Strain-induced conformation transition in two-dimensional covalent organic frameworks. *Phys. Rev. B*, 112:144105, Oct 2025. doi:10.1103/jmx6-rpvl.

72. Bowen Wang, Baowen Wang, Hejin Yan, and Yongqing Cai. Oxygen vacancy-driven interfacial alloying and mixing for enhanced heat transfer in gallium oxide. *Materials Today Physics*, 54:101714, 2025. doi:10.1016/j.mtphys.2025.101714.

73. Jingwen Wang, Zheng Zhu, Tianran Jiang, and Ke Chen. Machine learning revealed giant thermal conductivity reduction by strong phonon localization in two-angle disordered twisted multilayer graphene. *npj Computational Materials*, 11(1):195, 2025. doi:10.1038/s41524-025-01678-3.

74. Lin-Di Wang, Ying-Bin Cheng, and Jian Zhou. Molecular dynamics study on phonon coherent transport in iii–v semiconductor superlattices. *Journal of Applied Physics*, 137(11):115103, 03 2025. doi:10.1063/5.0253919.

75. Yan Wang, Lin Xie, Haobo Yang, Mingyuan Hu, Xin Qian, Ronggui Yang, and Jiaqing He. Strong orbital-lattice coupling induces glassy thermal conductivity in high-symmetry single crystal $BaTiS_3$. *Phys. Rev. X*, 15:011066, Mar 2025. doi:10.1103/PhysRevX.15.011066.

76. Yanzhou Wang, Zheyong Fan, Ping Qian, Miguel A. Caro, and Tapio Ala-Nissila. Density dependence of thermal conductivity in nanoporous and amorphous carbon with machine-learned molecular dynamics. *Phys. Rev. B*, 111:094205, Mar 2025. doi:10.1103/PhysRevB.111.094205.

77. Yong Wang, Junjie Wang, Ge Yao, Zheyong Fan, Enzo Granato, Michael Kosterlitz, Tapio Ala-Nissila, Roberto Car, and Jian Sun. Phase transitions and dimensional cross-over in layered confined solids. *Proceedings of the National Academy of Sciences*, 122(17):e2502980122, 2025. doi:10.1073/pnas.2502980122.

78. Yongheng Wang, Xiangzheng Jia, Ruixiang Chen, and Enlai Gao. Data-driven extreme-value statistics for fracture size effects. *Journal of Applied Physics*, 138(11):114303, 09 2025. doi:10.1063/5.0282021.

79. Yunlong Wang, Zhixin Liang, Chi Ding, Junjie Wang, Zheyong Fan, Hui-Tian Wang, Dingyu Xing, and Jian Sun. Gputb: efficient machine learning tight-binding method for large-scale electronic properties calculations. *Computational Materials Today*, pages 100039, 2025. doi:10.1016/j.commt.2025.100039.

80. Yunlong Wang, Jiuyang Shi, Zhixin Liang, Tianheng Huang, Junjie Wang, Chi Ding, Chris J. Pickard, Hui-Tian Wang, Dingyu Xing, Dongdong Ni, and Jian Sun. Machine learning simulations reveal oxygen's phase diagram and thermal properties at conditions relevant to white dwarfs. *Nature Communications*, 16(1):5504, 2025. doi:10.1038/s41467-025-61390-0.

81. Guo Wei, Zichen Song, Jie Hou, Jiaqi Wang, Akksay Singh, and Lei Li. Resolving vacancy clustering mechanisms in tungsten via machine-learning-potential-based simulations at experimental time scales. *Acta Materialia*, pages 121529, 2025. doi:10.1016/j.actamat.2025.121529.

82. Xin Wu, Zhang Wu, Ting Liang, Zheyong Fan, Jianbin Xu, Masahiro Nomura, and Penghua Ying. Phonon coherence and minimum thermal conductivity in disordered superlattices. *Phys. Rev. B*, 111:085413, Feb 2025. doi:10.1103/PhysRevB.111.085413.

83. Yong-Chao Wu, Xiaoya Chang, Zhi Gen Yu, Yong-Wei Zhang, and Jian-Li Shao. Revealing the role of al4c3 in the mechanical behavior of aluminum/graphene composites through machine learning potential-driven atomistic simulations. *Mechanics of Materials*, 209:105428, 2025. doi:10.1016/j.mechmat.2025.105428.

84. Yuzhu Wu, Zhifeng Sun, Ningning Liu, Zhong Wang, Yueming Hu, Tianqi Bai, Tao Wang, Jingyang Chen, Xiaopan Qiu, Xudong Zhang, Fushun Liang, Dongcheng Jiao, Dan Li, Lishuo Han, Wenhu Wang, Qin Xie, Ronghua Zhang, Ali Cai, Yuqi Xia, Haonan Zhai, Zhong-zhen Yu, Yue Qi, Chu Wang, Peng Gao, Xiucai Sun, Bingyang Cao, Yuqing Song, and Zhongfan Liu. Controlled growth of graphene-skinned al2o3 powders by fluidized bed-chemical vapor deposition for heat dissipation. *Advanced Science*, pages e03388, 2025. doi:10.1002/advs.202503388.

85. Xing Xiang and Yanguang Zhou. Comparable thermal transport between $Li_{15}Si_4$ and lisi. *Phys. Rev. Mater.*, 9:085401, Aug 2025. doi:10.1103/ndlz-hxwq.

86. Feng Xiao, Qing-Yu Xie, Ru Yu, Huashan Li, Junrong Zhang, and Bao-Tian Wang. Impact of lattice distortions and overdamped vibrations on the structural properties and thermal transport of strongly anharmonic $AgSnSbTe_3$ crystals. *Phys. Rev. B*, 111:184304, May 2025. doi:10.1103/PhysRevB.111.184304.

87. Yang Xiao, Yuqi Liu, Zihan Tan, Bohan Zhang, Ke Xu, Zheyong Fan, Shunda Chen, Shiyun Xiong, and Haikuan Dong. Optimizing thermoelectric performance of graphene antidot lattices via quantum transport and machine-learning molecular dynamics simulations. *Phys. Rev. Mater.*, 9:084603, 2025. doi:10.1103/hbzs-bzb8.

88. Boyuan Xu, Liyi Bai, Shenzhen Xu, and Qisheng Wu. Observing nucleation and crystallization of rock salt lif from molten state through molecular dynamics simulations with refined machine-learned force field. *The Journal of Chemical Physics*, 162(23):234705, 06 2025. doi:10.1063/5.0276535.

89. Ke Xu, Hekai Bu, Shuning Pan, Eric Lindgren, Yongchao Wu, Yong Wang, Jiahui Liu, Keke Song, Bin Xu, Yifan Li, Tobias Hainer, Lucas Svensson, Julia Wiktor, Rui Zhao, Hongfu Huang, Cheng Qian, Shuo Zhang, Zezhu Zeng, Bohan Zhang, Benrui Tang, Yang Xiao, Zihan Yan, Jiuyang Shi, Zhixin Liang, Junjie Wang, Ting

Liang, Shuo Cao, Yanzhou Wang, Penghua Ying, Nan Xu, Chengbing Chen, Yuwen Zhang, Zherui Chen, Xin Wu, Wenwu Jiang, Esme Berger, Yanlong Li, Shunda Chen, Alexander J. Gabourie, Haikuan Dong, Shiyun Xiong, Ning Wei, Yue Chen, Jianbin Xu, Feng Ding, Zhimei Sun, Tapio Ala-Nissila, Ari Harju, Jincheng Zheng, Pengfei Guan, Paul Erhart, Jian Sun, Wengen Ouyang, Yanjing Su, and Zheyong Fan. Gpumd 4.0: a high-performance molecular dynamics package for versatile materials simulations with machine-learned potentials. *Materials Genome Engineering Advances*, pages e70028, 2025. doi:10.1002/mgea.70028.

90. Ke Xu, Yuan Li, Dongliang Ding, Ting Liang, Jianyang Wu, and Jianbin Xu. Critical size transitions in silicon nanowires: amorphization, phonon hydrodynamics, and thermal conductivity. *The Journal of Physical Chemistry Letters*, 2025. doi:10.1021/acs.jpclett.5c01802.

91. Ke Xu, Ting Liang, Nan Xu, Penghua Ying, Shunda Chen, Ning Wei, Jianbin Xu, and Zheyong Fan. Nep-mb-pol: a unified machine-learned framework for fast and accurate prediction of water's thermodynamic and transport properties. *npj Computational Materials*, 11(1):279, 2025. doi:10.1038/s41524-025-01777-1.

92. Xuyang Xue, Jian Yuan, Junyi Yang, and Huashan Li. Unraveling the rare ferroelectric-antiferroelectric transition in the two-dimensional hybrid perovskites $(PMA)_2PbBr_4$. *Phys. Rev. B*, 112:024106, Jul 2025. doi:10.1103/42l1-fmkt.

93. Fuwei Yang, Wenjiang Zhou, Tian Gu, Zhibin Zhang, Kexin Zhang, Wujuan Yan, Yuxi Wang, Kaihui Liu, and Bai Song. Thermal transport in rhombohedral boron nitride. *Phys. Rev. B*, pages 115422, Sep 2025. doi:10.1103/wj6y-bsjq.

94. Jin Yang, Xueyan Zhu, Alan J. H. McGaughey, Yee Sin Ang, and Wee-Liat Ong. Two-mode terms in wigner transport equation elucidate anomalous thermal transport in amorphous silicon. *Phys. Rev. B*, 111:094206, Mar 2025. doi:10.1103/PhysRevB.111.094206.

95. Rui Yang, Qiaoling Si, Qiang Sheng, Mingyang Yang, Mu Du, Hu Zhang, Xiao-Lei Shi, Guihua Tang, and Zhi-Gang Chen. Atomic armor for thermal stability in nanoporous structures. *Proceedings of the National Academy of Sciences*, 122(36):e2510746122, 2025. doi:10.1073/pnas.2510746122.

96. Penghua Ying, Cheng Qian, Rui Zhao, Yanzhou Wang, Ke Xu, Feng Ding, Shunda Chen, and Zheyong Fan. Advances in modeling complex materials: the rise of neuroevolution potentials. *Chemical Physics Reviews*, 6(1):011310, 03 2025. doi:10.1063/5.0259061.

97. Penghua Ying, Wenjiang Zhou, Lucas Svensson, Esmée Berger, Erik Fransson, Fredrik Eriksson, Ke Xu, Ting Liang, Jianbin Xu, Bai Song, Shunda Chen, Paul Erhart, and Zheyong Fan. Highly efficient path-integral molecular dynamics simulations with gpumd using neuroevolution potentials: case studies on thermal properties of materials. *The Journal of Chemical Physics*, 162(6):064109, 02 2025. doi:10.1063/5.0241006.

98. Shaobo Yu, Junjie Wang, Yu Han, Qiuhan Jia, Zhixin Liang, Ziyang Yang, Yujian Pan, Hao Gao, and Jian Sun. A generalized method for refining and selecting random crystal structures using graph theory. *The Journal of Chemical Physics*, 163(9):094106, 09 2025. doi:10.1063/5.0278803.

99. Li Yuan, Zheng Weidong, Pu Jin Huan, Wang Qi, and Jiang Jian-Hua. Strong lattice anharmonicity and glass-like lattice thermal conductivity in nitrohalide double antiperovskites: a case study based on machine-learning potentials. *Thermo-X*, 2025. doi:10.70401/tx.2025.0001.

100. Jincheng Yue, Rongkun Chen, Dengke Ma, and Shiqian Hu. Nanoparticle-assisted phonon transport modulation in si/ge heterostructures using neuroevolution potential machine learning models. *Chinese Physics Letters*, 2025. URL: http://iopscience.iop.org/article/10.1088/0256-307X/42/3/036301, doi:10.1088/0256-307X/42/3/036301.

101. Yuxuan Zeng, Wei Cao, Yijing Zuo, Tan Peng, Yue Hou, Ling Miao, Ziyu Wang, and Jing Shi. Accelerating the discovery of materials with expected thermal conductivity via a synergistic strategy of dft and interpretable deep learning. *Materials Futures*, 4(4):045602, oct 2025. doi:10.1088/2752-5724/ae08d0.

102. Zezhu Zeng, Zheyong Fan, Michele Simoncelli, Chen Chen, Ting Liang, Yue Chen, Geoff Thornton, and Bingqing Cheng. Lattice distortion leads to glassy thermal transport in crystalline $cs_3bi_2i_6cl_3$. *Proceedings of the National Academy of Sciences*, pages e2415664122, 2025. doi:10.1073/pnas.2415664122.

103. Zezhu Zeng, Xia Liang, Zheyong Fan, Yue Chen, Michele Simoncelli, and Bingqing Cheng. Thermal transport of amorphous hafnia across the glass transition. *ACS Materials Letters*, pages 2695–2701, 2025. doi:10.1021/acsmaterialslett.5c00263.

104. Majid Zeraati, Artem R. Oganov, Alexey P. Maltsev, and Sergey F. Solodovnikov. Computational screening of complex oxides for next-generation thermal barrier coatings. *Journal of Applied Physics*, 137(6):065106, 02 2025. doi:10.1063/5.0253010.

105. Bowen Zhang, Peiyao Zhang, Changguo Wang, Huifeng Tan, Liwei Dong, Jia-Yan Liang, and Yuanpeng Liu. Atomistic insights into stress-driven lithiation at silicon anode crack tips. *ACS Applied Materials & Interfaces*, 2025. doi:10.1021/acsami.5c11237.

106. Chenxin Zhang, Yanyan Chen, Qian Wang, and Puru Jena. Phonon localization and a boson-peak-like anomaly in twisted penta-pdse2 bilayer. *Nano Lett.*, 2025. doi:10.1021/acs.nanolett.5c01621.

107. Guoqiang Zhang, Huasong Qin, and Yilun Liu. Strength of 2d materials with multiple defects. *Thin-Walled Structures*, 217:113897, 2025. doi:10.1016/j.tws.2025.113897.

108. Haoming Zhang, Mo Cheng, Xuanyu Jiang, Hui Zhang, Xiaodong Pi, Deren Yang, and Tianqi Deng. Neuroevolution potential for thermal transport in silicon carbide. *Journal of Materials Informatics*, 2025. doi:10.20517/jmi.2025.06.

109. Jian Zhang, Zhuo Zhao, Miao Jiang, Yuan Cheng, and Gang Zhang. Thermal properties of hexagonal diamond: machine learning potential and molecular dynamics study. *Phys. Rev. Mater.*, pages 094603, Sep 2025. doi:10.1103/z9gl-yb41.

110. Lei Zhang, Wenhao Luo, Renxi Liu, Mohan Chen, Zhongbo Yan, and Kun Cao. Exploring the energy landscape of aluminas through machine learning interatomic potential. *Phys. Rev. Mater.*, 9:023801, Feb 2025. doi:10.1103/PhysRevMaterials.9.023801.

111. Pingyang Zhang, Shaodong Zhang, Yihan Qin, Tingting Du, Lei Wei, and Xiangyu Li. Machine learning-driven molecular dynamics decodes thermal tuning in graphene foam composites. *npj Computational Materials*, 11(1):214, 2025. doi:10.1038/s41524-025-01710-6.

112. Shaodong Zhang, Pan Chen, Lei Wei, Pingyang Zhang, Xuping Wang, Bing Liu, Yuanyuan Zhang, Xianshun Lv, Xiangyu Li, and Tingting Du. Theoretical investigation on the dynamic thermal transport properties of graphene foam by machine-learning molecular dynamics simulations. *International Journal of Thermal Sciences*, 210:109631, 2025. doi:10.1016/j.ijthermalsci.2024.109631.

113. ZhongTing Zhang, Jian Luo, HengAn Wu, Hao Ma, and YinBo Zhu. Unveiling the microscopic origin of anomalous thermal conductivity in amorphous carbon. *Science Advances*, 11(23):eadx5007, 2025. doi:10.1126/sciadv.adx5007.

114. Wenjiang Zhou, Nianjie Liang, Xiguang Wu, Shiyun Xiong, Zheyong Fan, and Bai Song. Insight into the effect of force error on the thermal conductivity from machine-learned potentials. *Materials Today Physics*, 50:101638, 2025. doi:10.1016/j.mtphys.2024.101638.

115. Xianteng Zhou, Yuanji Xu, Yue Chen, and Fuyang Tian. Mechanism on lattice thermal conductivity of carbon-vacancy and porous medium entropy ceramics. *Scripta Materialia*, 259:116568, 2025. doi:10.1016/j.scriptamat.2025.116568.

116. Xiaoye Zhou, Yuqi Liu, Benrui Tang, Junyuan Wang, Haikuan Dong, Xiaoming Xiu, Shunda Chen, and Zheyong Fan. Million-atom heat transport simulations of polycrystalline graphene approaching first-principles accuracy enabled by neuroevolution potential on desktop gpus. *Journal of Applied Physics*, 137(1):014305, 01 2025. doi:10.1063/5.0244987.

117. Zijie Zhu, Yiwen Liu, Yuanbin Qin, Fangchao Gu, Lei Zhuang, Hulei Yu, and Yanhui Chu. Tough and strong bioinspired high-entropy all-ceramics with a contiguous network structure. *Nature Communications*, 16(1):4587, 2025. doi:10.1038/s41467-025-59914-9.

9.2 2024

1. Ethan Berger and Hannu-Pekka Komsa. Polarizability models for simulations of finite temperature raman spectra from machine learning molecular dynamics. *Phys. Rev. Mater.*, 8:043802, Apr 2024. doi:10.1103/PhysRevMaterials.8.043802.

2. Ethan Berger, Juha Niemelä, Outi Lampela, André H. Juffer, and Hannu-Pekka Komsa. Raman spectra of amino acids and peptides from machine learning polarizabilities. *Journal of Chemical Information and Modeling*, 64(12):4601–4612, 2024. doi:10.1021/acs.jcim.4c00077.

3. Margaret L. Berrens, Arpan Kundu, Marcos F. Calegari Andrade, Tuan Anh Pham, Giulia Galli, and Davide Donadio. Nuclear quantum effects on the electronic structure of water and ice. *The Journal of Physical Chemistry Letters*, 15(26):6818–6825, 2024. doi:10.1021/acs.jpclett.4c01315.

4. Chenyang Cao, Shuo Cao, YuanXu Zhu, Haikuan Dong, Yanzhou Wang, and Ping Qian. Thermal transports of 2d phosphorous carbides by machine learning molecular dynamics simulations. *International Journal of Heat and Mass Transfer*, 224:125359, 2024. doi:10.1016/j.ijheatmasstransfer.2024.125359.

5. Rongkun Chen, Yu Tian, Jiayi Cao, Weina Ren, Shiqian Hu, and Chunhua Zeng. Unified deep learning network for enhanced accuracy in predicting thermal conductivity of bilayer graphene, hexagonal boron nitride, and their heterostructures. *Journal of Applied Physics*, 135(14):145106, 04 2024. doi:10.1063/5.0201698.

6. Shunda Chen, Xiaochen Jin, Wanyu Zhao, and Tianshu Li. Intricate short-range order in gesn alloys revealed by atomistic simulations with highly accurate and efficient machine-learning potentials. *Phys. Rev. Mater.*, 8:043805, Apr 2024. doi:10.1103/PhysRevMaterials.8.043805.

7. Zekun Chen, Margaret L. Berrens, Kam-Tung Chan, Zheyong Fan, and Davide Donadio. Thermodynamics of water and ice from a fast and scalable first-principles neuroevolution potential. *Journal of Chemical & Engineering Data*, 69(1):128–140, 2024. doi:10.1021/acs.jced.3c00561.

8. Ruihuan Cheng, Zezhu Zeng, Chen Wang, Niuchang Ouyang, and Yue Chen. Impact of strain-insensitive low-frequency phonon modes on lattice thermal transport in A_2XB_6-type perovskites. *Phys. Rev. B*, 109:054305, Feb 2024. doi:10.1103/PhysRevB.109.054305.

9. Yajuan Cheng, Honggang Zhang, Shiyun Xiong, Sebastian Volz, and Tao Zhang. Tuning the thermal resistance of sige phononic interfaces across ballistic and diffusive regimes. *International Journal of Heat and Mass Transfer*, 235:126144, 2024. doi:10.1016/j.ijheatmasstransfer.2024.126144.

10. Higo de Araujo Oliveira, Zheyong Fan, Ari Harju, and Luiz Felipe C. Pereira. Tuning the thermal conductivity of silicon phononic crystals via defect motifs: implications for thermoelectric devices and photovoltaics. *ACS Applied Nano Materials*, 0(0):0, 2024. doi:10.1021/acsanm.4c01875.

11. Quan Deng, Qiang Liu, Ming Yuan, Xiaohui Duan, Lin Gan, Jinzhe Yang, Wenlai Zhao, Zhenxiang Zhang, Guiming Wu, Wayne Luk, Haohuan Fu, and Guangwen Yang. Acceleration of multi-body molecular dynamics with customized parallel dataflow. *IEEE Transactions on Parallel & Distributed Systems*, 35(12):2297–2314, 2024. doi:10.1109/TPDS.2024.3420441.

12. Haikuan Dong, Yongbo Shi, Penghua Ying, Ke Xu, Ting Liang, Yanzhou Wang, Zezhu Zeng, Xin Wu, Wenjiang Zhou, Shiyun Xiong, Shunda Chen, and Zheyong Fan. Molecular dynamics simulations of heat transport using machine-learned potentials: a mini-review and tutorial on gpumd with neuroevolution potentials. *Journal of Applied Physics*, 135(16):161101, 04 2024. doi:10.1063/5.0200833.

13. Haoyu Dong, Zhiqiang Li, Baole Sun, Yanguang Zhou, Linhua Liu, and Jia-Yue Yang. Thermal transport in disordered wurtzite scaln alloys using machine learning interatomic potentials. *Materials Today Communications*, 39:109213, 2024. doi:10.1016/j.mtcomm.2024.109213.

14. Hongzhao Fan, Penghua Ying, Zheyong Fan, Yue Chen, Zhigang Li, and Yanguang Zhou. Anomalous strain-dependent thermal conductivity in the metal-organic framework hkust-1. *Phys. Rev. B*, 109:045424, Jan 2024. doi:10.1103/PhysRevB.109.045424.

15. Zheyong Fan, Yang Xiao, Yanzhou Wang, Penghua Ying, Shunda Chen, and Haikuan Dong. Combining linear-scaling quantum transport and machine-learning molecular dynamics to study thermal and electronic transports in complex materials. *Journal of Physics: Condensed Matter*, 36(24):245901, mar 2024. doi:10.1088/1361-648X/ad31c2.

16. Mandi Fang, Shi Tang, Zheyong Fan, Yao Shi, Nan Xu, and Yi He. Transferability of machine learning models for predicting raman spectra. *The Journal of Physical Chemistry A*, 128(12):2286–2294, 2024. doi:10.1021/acs.jpca.3c07109.

17. H. F. Feng, B. Liu, J. L. Bai, X. Zhang, Z. X. Song, and Zhi-Xin Guo. Nontrivial impact of interlayer coupling on thermal conductivity: opposing trends in in-plane and out-of-plane phonons. *Phys. Rev. B*, 110:214304, Dec 2024. doi:10.1103/PhysRevB.110.214304.

18. Lucas Fine, Rasmus Lavén, Zefeng Wei, Tatsuya Tsumori, Hiroshi Kageyama, Ryoichi Kajimoto, Mónica Jimenéz-Ruiz, Michael Marek Koza, and Maths Karlsson. Configuration and dynamics of hydride ions in the nitride-hydride catalyst ca3crn3h. *Chemistry of Materials*, Dec 2024. doi:10.1021/acs.chemmater.4c02897.

19. Dylan A. Folkner, Zekun Chen, Giuseppe Barbalinardo, Florian Knoop, and Davide Donadio. Elastic moduli and thermal conductivity of quantum materials at finite temperature. *Journal of Applied Physics*, 136(22):221101, 12 2024. doi:10.1063/5.0238723.

20. Erik Fransson, Petter Rosander, Paul Erhart, and Göran Wahnström. Understanding correlations in $BaZrO_3$: structure and dynamics on the nanoscale. *Chemistry of Materials*, 36(1):514–523, 2024. doi:10.1021/acs.chemmater.3c02548.

21. Erik Fransson, Julia Wiktor, and Paul Erhart. Impact of organic spacers and dimensionality on templating of halide perovskites. *ACS Energy Letters*, 9(8):3947–3954, 2024. doi:10.1021/acsenergylett.4c01283.

22. Alexander J. Gabourie, Carlos A. Polanco, Connor J. McClellan, Haotian Su, Mohamadali Malakoutian, Çağil Köroğlu, Srabanti Chowdhury, Davide Donadio, and Eric Pop. Ai-accelerated atoms-to-circuits thermal simulation pipeline for integrated circuit design. *2024 IEEE International Electron Devices Meeting (IEDM)*, pages 1–4, Dec 2024. doi:10.1109/IEDM50854.2024.10873564.

23. Guan Huang, Lichuan Zhang, Shibing Chu, Yuee Xie, and Yuanping Chen. A highly ductile carbon material made of triangle rings: a study of machine learning. *Applied Physics Letters*, 124(4):043103, 01 2024. doi:10.1063/5.0189906.

24. Xiaofan Huang, Chengzhi Li, Minhui Yuan, Jing Shuai, Xiang-Guo Li, and Yanglong Hou. Unphysical grain size dependence of lattice thermal conductivity in $Mg_3(Sb, Bi)_2$: an atomistic view of concentration dependent segregation effects. *Materials Today Physics*, pages 101386, 2024. doi:10.1016/j.mtphys.2024.101386.

25. Guotai Li, Jialin Tang, Jiongzhi Zheng, Qi Wang, Zheng Cui, Ke Xu, Jianbin Xu, Te-Huan Liu, Guimei Zhu, Ruiqiang Guo, and Baowen Li. Convergent thermal conductivity in strained monolayer graphene. *Phys. Rev. B*, 109:035420, Jan 2024. doi:10.1103/PhysRevB.109.035420.

26. Hongfei Li, Yuanxu Zhu, MengFan Chu, Haikuan Dong, and Guohua Zhang. Thermal conductivity of irregularly shaped nanoparticles from equilibrium molecular dynamics. *Journal of Physics: Condensed Matter*, 36(34):345703, may 2024. doi:10.1088/1361-648X/ad44f9.

27. Kaiqi Li, Bin Liu, Jian Zhou, and Zhimei Sun. Revealing the crystallization dynamics of sb–te phase change materials by large-scale simulations. *J. Mater. Chem. C*, pages –, 2024. doi:10.1039/D3TC04586B.

28. Youtian Li, Yangyu Guo, Shiyun Xiong, and Hongliang Yi. Enhanced heat transport in amorphous silicon via microstructure modulation. *International Journal of Heat and Mass Transfer*, 222:125167, 2024. doi:10.1016/j.ijheatmasstransfer.2023.125167.

29. Zhiqiang Li, Haoyu Dong, Jian Wang, Linhua Liu, and Jia-Yue Yang. Active learning molecular dynamics-assisted insights into ultralow thermal conductivity of two-dimensional covalent organic frameworks. *International Journal of Heat and Mass Transfer*, 225:125404, 2024. doi:10.1016/j.ijheatmasstransfer.2024.125404.

30. Zhiqiang Li, Jian Wang, Haoyu Dong, Yanguang Zhou, Linhua Liu, and Jia-Yue Yang. Mechanistic insights into water filling effects on thermal transport of carbon nanotubes from machine learning molecular dynamics. *International Journal of Heat and Mass Transfer*, 235:126152, 2024. doi:10.1016/j.ijheatmasstransfer.2024.126152.

31. Yiwen Liu, Hong Meng, Zijie Zhu, Hulei Yu, Lei Zhuang, and Yanhui Chu. Predicting mechanical and thermal properties of high-entropy ceramics via transferable machine-learning-potential-based molecular dynamics. *Advanced Functional Materials*, pages 2418802, Dec 2024. doi:10.1002/adfm.202418802.

32. Shuang Lyu, Ruihuan Cheng, Haiqi Li, and Yue Chen. Effects of local chemical ordering on the thermal transport in entropy-regulated pbse-based thermoelectric materials. *Applied Physics Letters*, 124(23):232202, 06 2024. doi:10.1063/5.0213996.

33. Mufasila Mumthaz Muhammed and Junais Habeeb Mokkath. Thermal characteristics of $CsPbX_3$ (x =cl/br/i) halide perovskites. *Materials Today Communications*, 41:110628, 2024. doi:10.1016/j.mtcomm.2024.110628.

34. Shuning Pan, Jiuyang Shi, Zhixin Liang, Cong Liu, Junjie Wang, Yong Wang, Hui-Tian Wang, Dingyu Xing, and Jian Sun. Shock compression pathways to pyrite silica from machine learning simulations. *Phys. Rev. B*, 110:224101, Dec 2024. doi:10.1103/PhysRevB.110.224101.

35. Paolo Pegolo and Federico Grasselli. Thermal transport of glasses via machine learning driven simulations. *Frontiers in Materials*, 2024. doi:10.3389/fmats.2024.1369034.

36. Zijun Qi, Xiang Sun, Zhanpeng Sun, Qijun Wang, Dongliang Zhang, Kang Liang, Rui Li, Diwei Zou, Lijie Li, Gai Wu, Wei Shen, and Sheng Liu. Interfacial optimization for aln/diamond heterostructures via machine learning potential molecular dynamics investigation of the mechanical properties. *ACS Applied Materials & Interfaces*, 16(21):27998–28007, 2024. doi:10.1021/acsami.4c06055.

37. Guoliang Ru, Weihong Qi, Shu Sun, Kewei Tang, Chengfeng Du, and Weimin Liu. Interlayer friction and adhesion effects in penta-$pdse_2$-based van der waals heterostructures. *Advanced Science*, 11(34):2400395, 2024. doi:10.1002/advs.202400395.

38. Christian Schäfer, Jakub Fojt, Eric Lindgren, and Paul Erhart. Machine learning for polaritonic chemistry: accessing chemical kinetics. *Journal of the American Chemical Society*, 146(8):5402–5413, 2024. doi:10.1021/jacs.3c12829.

39. Pengjie Shi and Zhiping Xu. Exploring fracture of h-bn and graphene by neural network force fields. *Journal of Physics: Condensed Matter*, 36(41):415401, jul 2024. doi:10.1088/1361-648X/ad5c31.

40. Soonsung So and Joo-Hyoung Lee. Unraveling interfacial thermal transport in β -ga_2o_3/h-bn van der waals heterostructures. *Materials Today Physics*, 46:101506, 2024. doi:10.1016/j.mtphys.2024.101506.

41. Soonsung So, Jae Hun Seol, and Joo-Hyoung Lee. Quasiballistic thermal transport in submicron-scale graphene nanoribbons at room-temperature. *Nanoscale Adv.*, 6:2919–2927, 2024. doi:10.1039/D4NA00261J.

42. Keke Song, Rui Zhao, Jiahui Liu, Yanzhou Wang, Eric Lindgren, Yong Wang, Shunda Chen, Ke Xu, Ting Liang, Penghua Ying, Nan Xu, Zhiqiang Zhao, Jiuyang Shi, Junjie Wang, Shuang Lyu, Zezhu Zeng, Shirong Liang, Haikuan Dong, Ligang Sun, Yue Chen, Zhuhua Zhang, Wanlin Guo, Ping Qian, Jian Sun, Paul Erhart, Tapio Ala-Nissila, Yanjing Su, and Zheyong Fan. General-purpose machine-learned potential for 16 elemental metals and their alloys. *Nature Communications*, 15(1):10208, 2024. doi:10.1038/s41467-024-54554-x.

43. Siddharth Sonti, Chenghan Sun, Zekun Chen, Robert Michael Kowalski, Joseph S. Kowalski, Davide Donadio, Surl-Hee Ahn, and Ambarish R. Kulkarni. Stability and dynamics of zeolite-confined gold nanoclusters. *Journal of Chemical Theory and Computation*, 20(18):8261–8269, 2024. doi:10.1021/acs.jctc.4c00978.

44. Chenghan Sun, Rajat Goel, and Ambarish R. Kulkarni. Developing cheap but useful machine learning-based models for investigating high-entropy alloy catalysts. *Langmuir*, 40(7):3691–3701, 2024. doi:10.1021/acs.langmuir.3c03401.

45. Zhanpeng Sun, Xiang Sun, Zijun Qi, Qijun Wang, Rui Li, Lijie Li, Gai Wu, Wei Shen, and Sheng Liu. Investigating thermal transport across the aln/diamond interface via the machine learning potential. *Diamond and Related Materials*, 147:111303, 2024. doi:10.1016/j.diamond.2024.111303.

46. Zhanpeng Sun, Dongliang Zhang, Zijun Qi, Qijun Wang, Xiang Sun, Kang Liang, Fang Dong, Yuan Zhao, Diwei Zou, Lijie Li, Gai Wu, Wei Shen, and Sheng Liu. Insight into interfacial heat transfer of β -ga$_2$o$_3$/diamond heterostructures via the machine learning potential. *ACS Applied Materials & Interfaces*, 16(24):31666–31676, 2024. doi:10.1021/acsami.3c19588.

47. Jialin Tang, Jiongzhi Zheng, Xiaohan Song, Lin Cheng, and Ruiqiang Guo. In-plane thermal conductivity of hexagonal boron nitride from 2d to 3d. *Journal of Applied Physics*, 135(20):205105, 05 2024. doi:10.1063/5.0206028.

48. Zhunyun Tang, Xiaoxia Wang, Chaoyu He, Jin Li, Mingxing Chen, Chao Tang, and Tao Ouyang. Effects of thermal expansion and four-phonon interactions on the lattice thermal conductivity of the negative thermal expansion material ScF_3. *Phys. Rev. B*, 110:134320, Oct 2024. doi:10.1103/PhysRevB.110.134320.

49. Heqing Tian, Wenhao Dong, Wenguang Zhang, and Chaxiu Guo. Machine learning techniques to probe the properties of molten salt phase change materials for thermal energy storage. *Cell Reports Physical Science*, pages 102042, 2024. doi:10.1016/j.xcrp.2024.102042.

50. Wei Tian, Chenyu Wang, and Ke Zhou. The dynamic diversity and invariance of ab initio water. *J. Chem. Theory Comput.*, 20(23):10667–10675, 2024. doi:10.1021/acs.jctc.4c01191.

51. B. Timalsina, H. G. Nguyen, and K. Esfarjani. Neuroevolution machine learning potential to study high temperature deformation of entropy-stabilized oxide $MgNiCoCuZnO_5$. *Journal of Applied Physics*, 136(15):155109, 10 2024. doi:10.1063/5.0224282.

52. Jing Wan, Guanting Li, Zeyu Guo, and Huasong Qin. Thermal transport in C_6N_7 monolayer: a machine learning based molecular dynamics study. *Journal of Physics: Condensed Matter*, 37(2):025301, oct 2024. doi:10.1088/1361-648X/ad81a6.

53. Bing Wang, Penghua Ying, and Jin Zhang. The thermoelastic properties of monolayer covalent organic frameworks studied by machine-learning molecular dynamics. *Nanoscale*, 16(1):237–248, 2024. doi:10.1039/D3NR04509A.

54. Chenyu Wang, Wei Tian, and Ke Zhou. Ab initio simulation of liquid water without artificial high temperature. *Journal of Chemical Theory and Computation*, 20(18):8202–8213, 2024. doi:10.1021/acs.jctc.4c00650.

55. Rui Wang, Hongyu Yu, Yang Zhong, and Hongjun Xiang. Identifying direct bandgap silicon structures with high-throughput search and machine learning methods. *The Journal of Physical Chemistry C*, 128(30):12677–12685, 2024. doi:10.1021/acs.jpcc.4c02967.

56. Xiaonan Wang, Jinfeng Yang, Penghua Ying, Zheyong Fan, Jin Zhang, and Huarui Sun. Dissimilar thermal transport properties in κ -Ga_2O_3 and β -Ga_2O_3 revealed by homogeneous nonequilibrium molecular dynamics simulations using machine-learned potentials. *Journal of Applied Physics*, 135(6):065104, 2024. doi:10.1063/5.0185854.

57. Peng Wei, Yiwen Liu, Yang Liu, Lei Zhuang, Hulei Yu, and Yanhui Chu. High-throughput composition screening of high-entropy rare-earth monosilicates for superior cmas corrosion resistance up to 1873 k. *Corrosion Science*, 235:112172, 2024. doi:10.1016/j.corsci.2024.112172.

58. Shuai Wu, Dongdong Kang, Xiaoxiang Yu, and Jiayu Dai. Thermal transport across armchair–zigzag graphene homointerface. *Applied Physics Letters*, 125(14):142206, 10 2024. doi:10.1063/5.0229671.

59. Xiguang Wu, Yajuan Cheng, Shaoming Huang, and Shiyun Xiong. Phonon resonance effect and defect scattering in covalently bonded carbon nanotube networks. *Phys. Rev. Appl.*, 22:024038, Aug 2024. doi:10.1103/PhysRevApplied.22.024038.

60. Xiguang Wu, Wenjiang Zhou, Haikuan Dong, Penghua Ying, Yanzhou Wang, Bai Song, Zheyong Fan, and Shiyun Xiong. Correcting force error-induced underestimation of lattice thermal conductivity in machine learning molecular dynamics. *The Journal of Chemical Physics*, 161(1):014103, 07 2024. doi:10.1063/5.0213811.

61. Xin Wu, Yunhui Wu, Xin Huang, Zheyong Fan, Sebastian Volz, Qiang Han, and Masahiro Nomura. Isotope interface engineering for thermal transport suppression in cryogenic graphene. *Materials Today Physics*, 46:101500,

2024. doi:10.1016/j.mtphys.2024.101500.

62. Zhang Wu, Rumeng Liu, Ning Wei, and Lifeng Wang. Unexpected reduction in thermal conductivity observed in graphene/h-bn heterostructures. *Physical Chemistry Chemical Physics*, 26(5):3823–3831, 2024. doi:10.1039/D3CP05407A.

63. Feiyang Xu, Dong Wang, Zhiguo Li, Hongxing Song, Lei Liu, Huayun Geng, Jianbo Hu, and Xiangrong Chen. Large-scale simulation of thermal conductivity in $CaSiO_3$ perovskite with neuroevolution potential. *Applied Physics Letters*, 125(3):034104, 07 2024. doi:10.1063/5.0217468.

64. Nan Xu, Petter Rosander, Christian Schäfer, Eric Lindgren, Nicklas Österbacka, Mandi Fang, Wei Chen, Yi He, Zheyong Fan, and Paul Erhart. Tensorial properties via the neuroevolution potential framework: fast simulation of infrared and raman spectra. *Journal of Chemical Theory and Computation*, 20(8):3273–3284, 2024. doi:10.1021/acs.jctc.3c01343.

65. Zihan Yan and Yizhou Zhu. Impact of lithium nonstoichiometry on ionic diffusion in tetragonal garnet-type $Li_7La_3Zr_2O_{12}$. *Chemistry of Materials*, Nov 2024. doi:10.1021/acs.chemmater.4c02454.

66. Ningxi Yang, Rongkun Chen, Yinong Liu, Weina Ren, and Shiqian Hu. Numerical evaluation of the effect of the twist angle on phonon hydrodynamics in twisted bilayer graphene. *Phys. Rev. B*, 110:245305, Dec 2024. doi:10.1103/PhysRevB.110.245305.

67. Penghua Ying, Amir Natan, Oded Hod, and Michael Urbakh. Effect of interlayer bonding on superlubric sliding of graphene contacts: a machine-learning potential study. *ACS Nano*, 18(14):10133–10141, 2024. doi:10.1021/acsnano.3c13099.

68. Linfeng Yu, Kexin Dong, Qi Yang, Yi Zhang, Zheyong Fan, Xiong Zheng, Huimin Wang, Zhenzhen Qin, and Guangzhao Qin. Dynamic mesophase transition induces anomalous suppressed and anisotropic phonon thermal transport. *npj Computational Materials*, 10:280, dec 2024. doi:10.1038/s41524-024-01442-z.

69. Maolin Yu, Zhiqiang Zhao, Wanlin Guo, and Zhuhua Zhang. Fracture toughness of two-dimensional materials dominated by edge energy anisotropy. *Journal of the Mechanics and Physics of Solids*, 186:105579, 2024. doi:10.1016/j.jmps.2024.105579.

70. Jincheng Yue, Shiqian Hu, Bin Xu, Rongkun Chen, Long Xiong, Rulei Guo, Yuanzhe Li, Lei-Lei Nian, Junichiro Shiomi, and Bo Zheng. Unraveling the mechanisms of thermal boundary conductance at the graphene-silicon interface: insights from ballistic, diffusive, and localized phonon transport regimes. *Phys. Rev. B*, 109:115302, Mar 2024. doi:10.1103/PhysRevB.109.115302.

71. Majid Zeraati, Artem R. Oganov, Tao Fan, and Sergey F. Solodovnikov. Searching for low thermal conductivity materials for thermal barrier coatings: a theoretical approach. *Phys. Rev. Mater.*, 8:033601, 2024. doi:10.1103/PhysRevMaterials.8.033601.

72. Chenxin Zhang, Jie Sun, Jiewei Cheng, and Qian Wang. Ultralow lattice thermal conductivity in quasi-one-dimensional BiI_3 with suppressed phonon coherence. *Phys. Rev. B*, 110:174309, Nov 2024. doi:10.1103/PhysRevB.110.174309.

73. Guoqiang Zhang, Siwei Zhao, Huasong Qin, and Yilun Liu. A unified strength criterion of diamane grain boundaries. *Extreme Mechanics Letters*, 68:102146, 2024. doi:10.1016/j.eml.2024.102146.

74. Jian Zhang, Hao-Chun Zhang, Weifeng Li, and Gang Zhang. Thermal conductivity of gete crystals based on machine learning potentials. *Chinese Physics B*, 33(4):047402, apr 2024. doi:10.1088/1674-1056/ad1b42.

75. Jintu Zhang, Odin Zhang, Luigi Bonati, and TingJun Hou. Combining transition path sampling with data-driven collective variables through a reactivity-biased shooting algorithm. *Journal of Chemical Theory and Computation*, 20(11):4523–4532, 2024. doi:10.1021/acs.jctc.4c00423.

76. Zhongwei Zhang, Qing Lu, Chi Ding, Tianheng Huang, Yijie Zhu, Yu Han, Junjie Wang, Xiaomeng Wang, and Jian Sun. Crystal structure prediction of lithium-beryllium alloys under pressure with distinctive electronic and dynamic behaviors. *Phys. Rev. B*, 110:174101, Nov 2024. doi:10.1103/PhysRevB.110.174101.

77. Zhiqiang Zhao, Min Yi, Wanlin Guo, and Zhuhua Zhang. General-purpose neural network potential for ti-al-nb alloys towards large-scale molecular dynamics with ab initio accuracy. *Phys. Rev. B*, 110:184115, Nov 2024. doi:10.1103/PhysRevB.110.184115.

78. Wenjiang Zhou and Bai Song. Isotope effect on four-phonon interaction and lattice thermal transport: an atomistic study of lithium hydride. *Phys. Rev. B*, 110:205202, Nov 2024. doi:10.1103/PhysRevB.110.205202.

9.3 2023

1. EA Bea, A Mancardo Viotti, MF Carusela, AG Monastra, and A Soba. Assessment, improvement, and comparison of different computational tools used for the simulation of heat transport in nanostructures. *SIMULATION*, 99(3):237–244, 2023. doi:10.1177/00375497211009611.

2. Ruihuan Cheng, Xingchen Shen, Stefan Klotz, Zezhu Zeng, Zehua Li, Alexandre Ivanov, Yu Xiao, Li-Dong Zhao, Frank Weber, and Yue Chen. Lattice dynamics and thermal transport of pbte under high pressure. *Phys. Rev. B*, 108:104306, 2023. doi:10.1103/PhysRevB.108.104306.

3. Yajuan Cheng, Zheyong Fan, Tao Zhang, Masahiro Nomura, Sebastian Volz, Guimei Zhu, Baowen Li, and Shiyun Xiong. Magic angle in thermal conductivity of twisted bilayer graphene. *Materials Today Physics*, 35:101093, 2023. doi:10.1016/j.mtphys.2023.101093.

4. Insa F. de Vries, Helena Osthues, and Nikos L. Doltsinis. Thermal conductivity across transition metal dichalcogenide bilayers. *iScience*, 2023. doi:10.1016/j.isci.2023.106447.

5. Haikuan Dong, Chenyang Cao, Penghua Ying, Zheyong Fan, Ping Qian, and Yanjing Su. Anisotropic and high thermal conductivity in monolayer quasi-hexagonal fullerene: A comparative study against bulk phase fullerene. *International Journal of Heat and Mass Transfer*, 206:123943, 2023. doi:10.1016/j.ijheatmasstransfer.2023.123943.

6. Peng-Hu Du, Cunzhi Zhang, Tingwei Li, and Qiang Sun. Low lattice thermal conductivity with two-channel thermal transport in the superatomic crystal PH_4AlBr_4. *Physical Review B*, 107(15):155204, 2023. doi:10.1103/PhysRevB.107.155204.

7. Fredrik Eriksson, Erik Fransson, Christopher Linderälv, Zheyong Fan, and Paul Erhart. Tuning the through-plane lattice thermal conductivity in van der waals structures through rotational (dis)ordering. *ACS Nano*, 17:25565, 2023. doi:10.1021/acsnano.3c09717.

8. Erik Fransson, J. Magnus Rahm, Julia Wiktor, and Paul Erhart. Revealing the free energy landscape of halide perovskites: metastability and transition characters in $CsPbBr_3$ and $MAPbI_3$. *Chemistry of Materials*, 35:8229–8238, 2023. doi:10.1021/acs.chemmater.3c01740.

9. Erik Fransson, Petter Rosander, Fredrik Eriksson, J. Magnus Rahm, Terumasa Tadano, and Paul Erhart. Limits of the phonon quasi-particle picture at the cubic-to-tetragonal phase transition in halide perovskites. *Commun Phys*, 6:173, 2023. doi:10.1038/s42005-023-01297-8.

10. Erik Fransson, Julia Wiktor, and Paul Erhart. Phase transitions in inorganic halide perovskites from machine-learned potentials. *The Journal of Physical Chemistry C*, 127:13773–13781, 2023. doi:10.1021/acs.jpcc.3c01542.

11. Yu Li and Jin-Wu Jiang. Vacancy defects impede the transition from peapods to diamond: a neuroevolution machine learning study. *Phys. Chem. Chem. Phys.*, 25:25629, 2023. doi:10.1039/D3CP03862A.

12. Ting Liang, Penghua Ying, Ke Xu, Zhenqiang Ye, Chao Ling, Zheyong Fan, and Jianbin Xu. Mechanisms of temperature-dependent thermal transport in amorphous silica from machine-learning molecular dynamics. *Phys. Rev. B*, 108:184203, Nov 2023. doi:10.1103/PhysRevB.108.184203.

13. Jiahui Liu, Jesper Byggmästar, Zheyong Fan, Ping Qian, and Yanjing Su. Large-scale machine-learning molecular dynamics simulation of primary radiation damage in tungsten. *Phys. Rev. B*, 108:054312, 2023. doi:10.1103/PhysRevB.108.054312.

14. Yingzhou Liu, Yinong Liu, Jincheng Yue, Long Xiong, Lei-Lei Nian, and Shiqian Hu. Modulation of interface modes for resonance-induced enhancement of the interfacial thermal conductance in pillar-based si/ge nanowires. *Phys. Rev. B*, 108:235426, Dec 2023. doi:10.1103/PhysRevB.108.235426.

15. Chenchen Lu, Zhi-hui Li, Shanchen Li, Zhen Li, Yingyan Zhang, Junhua Zhao, and Ning Wei. Molecular dynamics study of thermal transport properties across covalently bonded graphite-nanodiamond interfaces. *Carbon*, 213:118250, 2023. doi:10.1016/j.carbon.2023.118250.

16. Yimu Lu, Yongbo Shi, Junyuan Wang, Haikuan Dong, and Jie Yu. Reduction of thermal conductivity in carbon nanotubes by fullerene encapsulation from machine-learning molecular dynamics simulations. *Journal of Applied Physics*, 134(24):244901, 12 2023. doi:10.1063/5.0176338.

17. Niuchang Ouyang, Zezhu Zeng, Chen Wang, Qi Wang, and Yue Chen. Role of high-order lattice anharmonicity in the phonon thermal transport of silver halide AgX (X=Cl,Br,I). *Phys. Rev. B*, 108:174302, Nov 2023. doi:10.1103/PhysRevB.108.174302.

18. Shuning Pan, Tianheng Huang, Allona Vazan, Zhixin Liang, Cong Liu, Junjie Wang, Chris J. Pickard, Hui-Tian Wang, Dingyu Xing, and Jian Sun. Magnesium oxide-water compounds at megabar pressure and implications on planetary interiors. *Nature Communications*, 14(1):1165, 2023. doi:10.1038/s41467-023-36802-8.

19. Petter Rosander, Erik Fransson, Cosme Milesi-Brault, Constance Toulouse, Frédéric Bourdarot, Andrea Piovano, Alexei Bossak, Mael Guennou, and Göran Wahnström. Anharmonicity of the antiferrodistortive soft mode in barium zirconate $BaZrO_3$. *Phys. Rev. B*, 108:014309, 2023. doi:10.1103/PhysRevB.108.014309.

20. Wenhao Sha, Xuan Dai, Siyu Chen, Binglun Yin, and Fenglin Guo. Phonon thermal transport in two-dimensional PbTe monolayers via extensive molecular dynamics simulations with a neuroevolution potential. *Materials Today Physics*, 34:101066, 2023. doi:10.1016/j.mtphys.2023.101066.

21. Jiuyang Shi, Zhixing Liang, Junjie Wang, Shuning Pan, Chi Ding, Yong Wang, Hui-Tian Wang, Dingyu Xing, and Jian Sun. Double-shock compression pathways from diamond to bc8 carbon. *Phys. Rev. Lett.*, 131:146101, 2023. doi:10.1103/PhysRevLett.131.146101.

22. Yong-Bo Shi, Yuan-Yuan Chen, Hao Wang, Shuo Cao, Yuan-Xu Zhu, Meng-Fan Chu, Zhu-Feng Shao, Hai-Kuan Dong, and Ping Qian. Investigation of the mechanical and transport properties of $InGeX_3$ (X = S, Se and Te) monolayers using density functional theory and machine learning. *Physical Chemistry Chemical Physics*, 2023. doi:10.1039/D3CP01441J.

23. Yongbo Shi, Yuanyuan Chen, Haikuan Dong, Hao Wang, and Ping Qian. Investigation of phase transition, mechanical behavior and lattice thermal conductivity of halogen perovskites using machine learning interatomic potentials. *Phys. Chem. Chem. Phys.*, 25:30644–30655, 2023. doi:10.1039/D3CP04657E.

24. Ye Su, Yuan-Yuan Chen, Hao Wang, Hai-Kuan Dong, Shuo Cao, Li-Bin Shi, and Ping Qian. Origin of low lattice thermal conductivity and mobility of lead-free halide double perovskites. *Journal of Alloys and Compounds*, 962:170988, 2023. doi:10.1016/j.jallcom.2023.170988.

25. Zhanpeng Sun, Zijun Qi, Kang Liang, Xiang Sun, Zhaofu Zhang, Lijie Li, Qijun Wang, Guoqing Zhang, Gai Wu, and Wei Shen. A neuroevolution potential for predicting the thermal conductivity of α, β, and ϵ -Ga_2O_3. *Applied Physics Letters*, 123(19):192202, 11 2023. doi:10.1063/5.0165320.

26. Qi Wang, Chen Wang, Cheng Chi, Niuchang Ouyang, Ruiqiang Guo, Nuo Yang, and Yue Chen. Phonon transport in freestanding $SrTiO_3$ down to the monolayer limit. *Phys. Rev. B*, 108:115435, 2023. doi:10.1103/PhysRevB.108.115435.

27. Yanzhou Wang, Zheyong Fan, Ping Qian, Miguel A. Caro, and Tapio Ala-Nissila. Quantum-corrected thickness-dependent thermal conductivity in amorphous silicon predicted by machine learning molecular dynamics simulations. *Physical Review B*, 107(5):054303, 2023. doi:10.1103/PhysRevB.107.054303.

28. Han Wei, Yue Hu, and Hua Bao. Influence of point defects and multiscale pores on the different phonon transport regimes. *Communications Materials*, 4(1):1–9, 2023. doi:10.1038/s43246-023-00330-1.

29. Julia Wiktor, Erik Fransson, Dominik Kubicki, and Paul Erhart. Quantifying dynamic tilting in halide perovskites: chemical trends and local correlations. *Chemistry of Materials*, 35:6737–6744, 2023. doi:10.1021/acs.chemmater.3c00933.

30. Xin Wu, Xin Huang, Lei Yang, Zhongwei Zhang, Yangyu Guo, Sebastian Volz, Qiang Han, and Masahiro Nomura. Suppressed thermal transport in mathematically inspired 2d heterosystems. *Carbon*, 213:118264, 2023. doi:10.1016/j.carbon.2023.118264.

31. Xin Wu, Penghua Ying, Chunlei Li, and Qiang Han. Dual effects of hetero-interfaces on phonon thermal transport across graphene/C_3N lateral superlattices. *International Journal of Heat and Mass Transfer*, 201:123643, 2023. doi:10.1016/j.ijheatmasstransfer.2022.123643.

32. Jia-Hao Xiong, Zi-Jun Qi, Kang Liang, Xiang Sun, Zhan-Peng Sun, Qi-Jun Wang, Li-Wei Chen, Gai Wu, and Wei Shen. Molecular dynamics study of thermal conductivities of cubic diamond, lonsdaleite, and nanotwinned diamond via machine-learned potential. *Chinese Physics B*, 32(12):128101, dec 2023. doi:10.1088/1674-1056/ace4b4.

33. Ke Xu, Yongchao Hao, Ting Liang, Penghua Ying, Jianbin Xu, Jianyang Wu, and Zheyong Fan. Accurate prediction of heat conductivity of water by a neuroevolution potential. *The Journal of Chemical Physics*, 158(20):204114, 05 2023. doi:10.1063/5.0147039.

34. Chao Yang, Jian Wang, Dezhi Ma, Zhiqiang Li, Zhiyuan He, Linhua Liu, Zhiwei Fu, and Jia-Yue Yang. Phonon transport across gan-diamond interface: the nontrivial role of pre-interface vacancy-phonon scattering. *International Journal of Heat and Mass Transfer*, 214:124433, 2023. doi:10.1016/j.ijheatmasstransfer.2023.124433.

35. Penghua Ying, Haikuan Dong, Ting Liang, Zheyong Fan, Zheng Zhong, and Jin Zhang. Atomistic insights into the mechanical anisotropy and fragility of monolayer fullerene networks using quantum mechanical calculations and machine-learning molecular dynamics simulations. *Extreme Mechanics Letters*, 58:101929, 2023. doi:10.1016/j.eml.2022.101929.

36. Penghua Ying and Zheyong Fan. Combining the d3 dispersion correction with the neuroevolution machine-learned potential. *Journal of Physics: Condensed Matter*, 36(12):125901, dec 2023. doi:10.1088/1361-648X/ad1278.

37. Penghua Ying, Ting Liang, Ke Xu, Jianbin Xu, Zheyong Fan, Tapio Ala-Nissila, and Zheng Zhong. Variable thermal transport in black, blue, and violet phosphorene from extensive atomistic simulations with a neuroevolution potential. *International Journal of Heat and Mass Transfer*, 202:123681, 2023. doi:10.1016/j.ijheatmasstransfer.2022.123681.

38. Penghua Ying, Ting Liang, Ke Xu, Jin Zhang, Jianbin Xu, Zheng Zhong, and Zheyong Fan. Sub-micrometer phonon mean free paths in metal–organic frameworks revealed by machine learning molecular dynamics simulations. *ACS Applied Materials & Interfaces*, 15:36412, 2023. doi:10.1021/acsami.3c07770.

39. Honggang Zhang, Xiaokun Gu, Zheyong Fan, and Hua Bao. Vibrational anharmonicity results in decreased thermal conductivity of amorphous HfO_2 at high temperature. *Phys. Rev. B*, 108:045422, 2023. doi:10.1103/PhysRevB.108.045422.

40. Rui Zhao, Shucheng Wang, Zhuangzhuang Kong, Yunlei Xu, Kuan Fu, Ping Peng, and Cuilan Wu. Development of a neuroevolution machine learning potential of pd-cu-ni-p alloys. *Materials & Design*, 231:112012, 2023. doi:10.1016/j.matdes.2023.112012.

41. Ziyue Zhou, Jincheng Zeng, Zixuan Song, Yanwen Lin, Qiao Shi, Yongchao Hao, Yuequn Fu, Zhisen Zhang, and Jianyang Wu. Thermal conductivity of fivefold twinned silicon-germanium heteronanowires. *Phys. Chem. Chem. Phys.*, 25:25368–25376, 2023. doi:10.1039/D3CP02926C.

9.4 2022

1. Joakim Brorsson, Arsalan Hashemi, Zheyong Fan, Erik Fransson, Fredrik Eriksson, Tapio Ala-Nissila, Arkady V. Krasheninnikov, Hannu-Pekka Komsa, and Paul Erhart. Efficient calculation of the lattice thermal conductivity by atomistic simulations with ab initio accuracy. *Advanced Theory and Simulations*, 5(2):2100217, 2022. doi:10.1002/adts.202100217.

2. Yajuan Cheng, Shiyun Xiong, and Tao Zhang. Enhancing the coherent phonon transport in SiGe nanowires with dense Si/Ge interfaces. *Nanomaterials*, 12(24):4373, 2022. doi:10.3390/nano12244373.

3. Haikuan Dong, Zheyong Fan, Ping Qian, and Yanjing Su. Exactly equivalent thermal conductivity in finite systems from equilibrium and nonequilibrium molecular dynamics simulations. *Physica E: Low-dimensional Systems and Nanostructures*, 144:115410, 2022. doi:10.1016/j.physe.2022.115410.

4. Zheyong Fan. Improving the accuracy of the neuroevolution machine learning potential for multi-component systems. *Journal of Physics: Condensed Matter*, 34(12):125902, 2022. doi:10.1088/1361-648X/ac462b.

5. Zheyong Fan, Yanzhou Wang, Penghua Ying, Keke Song, Junjie Wang, Yong Wang, Zezhu Zeng, Ke Xu, Eric Lindgren, J. Magnus Rahm, Alexander J. Gabourie, Jiahui Liu, Haikuan Dong, Jianyang Wu, Yue Chen, Zheng Zhong, Jian Sun, Paul Erhart, Yanjing Su, and Tapio Ala-Nissila. GPUMD: A package for constructing accurate machine-learned potentials and performing highly efficient atomistic simulations. *The Journal of Chemical Physics*, 157(11):114801, 2022. doi:10.1063/5.0106617.

6. Hao Feng, Kai Zhang, Xin Wang, Guiqing Zhang, and Xiaoyong Guo. Thermal transport of bilayer graphene: a homogeneous nonequilibrium molecular dynamics study. *Physica Scripta*, 97(4):045704, 2022. doi:10.1088/1402-4896/ac5af0.

7. Alexander J. Gabourie, Çağıl Köroğlu, and Eric Pop. Substrate-dependence of monolayer MoS_2 thermal conductivity and thermal boundary conductance. *Journal of Applied Physics*, 131(19):195103, 2022. doi:10.1063/5.0089247.

8. Shuo Jin, Zhongwei Zhang, Yangyu Guo, Jie Chen, Masahiro Nomura, and Sebastian Volz. Optimization of interfacial thermal transport in Si/Ge heterostructure driven by machine learning. *International Journal of Heat and Mass Transfer*, 182:122014, 2022. doi:10.1016/j.ijheatmasstransfer.2021.122014.

9. Keqiang Li, Yajuan Cheng, Maofeng Dou, Wang Zeng, Sebastian Volz, and Shiyun Xiong. Tuning the Anisotropic Thermal Transport in 110-Silicon Membranes with Surface Resonances. *Nanomaterials*, 12(1):123, 2022. doi:10.3390/nano12010123.

10. Keqiang Li, Yajuan Cheng, Hongying Wang, Yangyu Guo, Zhongwei Zhang, Marc Bescond, Massahiro Nomura, Sebastian Volz, Xiaohong Zhang, and Shiyun Xiong. Phonon resonant effect in silicon membranes with different crystallographic orientations. *International Journal of Heat and Mass Transfer*, 183:122144, 2022. doi:10.1016/j.ijheatmasstransfer.2021.122144.

11. Zhi-Hui Li, Chenchen Lu, Aiqiang Shi, Sihan Zhao, Bingxian Ou, and Ning Wei. A multi-scale study on deformation and failure process of metallic structures in extreme environment. *International Journal of Molecular Sciences*, 23(22):14437, 2022. doi:10.3390/ijms232214437.

12. T. Liang, K. Xu, M. Han, Y. Yao, Z. Zhang, X. Zeng, J. Xu, and J. Wu. Abnormally high thermal conductivity in fivefold twinned diamond nanowires. *Materials Today Physics*, 25:100705, 2022. doi:10.1016/j.mtphys.2022.100705.

13. Wenhao Sha, Xuan Dai, Siyu Chen, and Fenglin Guo. Phonon thermal transport in graphene/h-BN superlattice monolayers. *Diamond and Related Materials*, 129:109341, 2022. doi:10.1016/j.diamond.2022.109341.

14. Wenhao Sha and Fenglin Guo. Thermal transport in two-dimensional carbon nitrides: A comparative molecular dynamics study. *Carbon Trends*, 7:100161, 2022. doi:10.1016/j.cartre.2022.100161.

15. Xiaomeng Wang, Yong Wang, Junjie Wang, Shuning Pan, Qing Lu, Hui-Tian Wang, Dingyu Xing, and Jian Sun. Pressure Stabilized Lithium-Aluminum Compounds with Both Superconducting and Superionic Behaviors. *Physical Review Letters*, 129(24):246403, 2022. doi:10.1103/PhysRevLett.129.246403.

16. Xin Wu and Qiang Han. Maximum thermal conductivity of multilayer graphene with periodic two-dimensional empty space. *International Journal of Heat and Mass Transfer*, 191:122829, 2022. doi:10.1016/j.ijheatmasstransfer.2022.122829.

17. Xin Wu and Qiang Han. Transition from incoherent to coherent phonon thermal transport across graphene/h-BN van der Waals superlattices. *International Journal of Heat and Mass Transfer*, 184:122390, 2022. doi:10.1016/j.ijheatmasstransfer.2021.122390.

18. Xin Wu and Qiang Han. Tunable anisotropic in-plane thermal transport of multilayer graphene induced by 2D empty space: Insights from interfaces. *Surfaces and Interfaces*, 33:102296, 2022. doi:10.1016/j.surfin.2022.102296.

19. Ke Xu, Ting Liang, Yuequn Fu, Zhen Wang, Zheyong Fan, Ning Wei, Jianbin Xu, Zhisen Zhang, and Jianyang Wu. Gradient nano-grained graphene as 2D thermal rectifier: A molecular dynamics based machine learning study. *Applied Physics Letters*, 121(13):133501, 2022. doi:10.1063/5.0108746.

20. Penghua Ying, Ting Liang, Yao Du, Jin Zhang, Xiaoliang Zeng, and Zheng Zhong. Thermal transport in planar sp^2-hybridized carbon allotropes: A comparative study of biphenylene network, pentaheptite and graphene. *International Journal of Heat and Mass Transfer*, 183:122060, 2022. doi:10.1016/j.ijheatmasstransfer.2021.122060.

21. Ziyue Zhou, Ke Xu, Zixuan Song, Zhen Wang, Yanwen Lin, Qiao Shi, Yongchao Hao, Yuequn Fu, Zhisen Zhang, and Jianyang Wu. Isotope doping-induced crossover shift in the thermal conductivity of thin silicon nanowires. *Journal of Physics: Condensed Matter*, 35(8):085702, 2022. doi:10.1088/1361-648X/acab4a.

9.5 2021

1. Giuseppe Barbalinardo, Zekun Chen, Haikuan Dong, Zheyong Fan, and Davide Donadio. Ultrahigh Convergent Thermal Conductivity of Carbon Nanotubes from Comprehensive Atomistic Modeling. *Physical Review Letters*, 127(2):025902, 2021. doi:10.1103/PhysRevLett.127.025902.

2. Xue-Kun Chen, Xiao-Yan Hu, Peng Jia, Zhong-Xiang Xie, and Jun Liu. Tunable anisotropic thermal transport in porous carbon foams: the role of phonon coupling. *International Journal of Mechanical Sciences*, 206:106576, 2021. doi:10.1016/j.ijmecsci.2021.106576.

3. Haikuan Dong, Petri Hirvonen, Zheyong Fan, Ping Qian, Yanjing Su, and Tapio Ala-Nissila. Heat transport across graphene/hexagonal-BN tilted grain boundaries from phase-field crystal model and molecular dynamics simulations. *Journal of Applied Physics*, 130(23):235102, 12 2021. doi:10.1063/5.0069134.

4. Haikuan Dong, Shiyun Xiong, Zheyong Fan, Ping Qian, Yanjing Su, and Tapio Ala-Nissila. Interpretation of apparent thermal conductivity in finite systems from equilibrium molecular dynamics simulations. *Physical Review B*, 103(3):035417, 2021. doi:10.1103/PhysRevB.103.035417.

5. Yao Du, Penghua Ying, and Jin Zhang. Prediction and optimization of the thermal transport in hybrid carbon-boron nitride honeycombs using machine learning. *Carbon*, 184:492–503, 2021. doi:10.1016/j.carbon.2021.08.035.

6. Zheyong Fan, Zezhu Zeng, Cunzhi Zhang, Yanzhou Wang, Keke Song, Haikuan Dong, Yue Chen, and Tapio Ala-Nissila. Neuroevolution machine learning potentials: Combining high accuracy and low cost in atomistic simulations and application to heat transport. *Physical Review B*, 104(10):104309, 2021. doi:10.1103/PhysRevB.104.104309.

7. Alexander J. Gabourie, Zheyong Fan, Tapio Ala-Nissila, and Eric Pop. Spectral decomposition of thermal conductivity: comparing velocity decomposition methods in homogeneous molecular dynamics simulations. *Phys. Rev. B*, 103:205421, May 2021. doi:10.1103/PhysRevB.103.205421.

8. Shi En Kim, Fauzia Mujid, Akash Rai, Fredrik Eriksson, Joonki Suh, Preeti Poddar, Ariana Ray, Chibeom Park, Erik Fransson, Yu Zhong, David A. Muller, Paul Erhart, David G. Cahill, and Jiwoong Park. Extremely

anisotropic van der Waals thermal conductors. *Nature*, 597(7878):660–665, 2021. doi:10.1038/s41586-021-03867-8.

9. Nicholas W. Lundgren, Giuseppe Barbalinardo, and Davide Donadio. Mode localization and suppressed heat transport in amorphous alloys. *Physical Review B*, 103(2):024204, 2021. doi:10.1103/PhysRevB.103.024204.

10. Sang-Hyuk Park, Hun Lee, Sehyuk Lee, Austin J. Minnich, Woo-Lim Jeong, Dong-Seon Lee, Soon-Sung So, Joo-Hyoung Lee, Young Min Song, and Young-Dahl Jho. Annealing-based manipulation of thermal phonon transport from light-emitting diodes to graphene. *Journal of Applied Physics*, 130(24):244303, 12 2021. doi:10.1063/5.0069466.

11. Soonsung So, Jeong-Yun Kim, Duckjong Kim, and Joo-Hyoung Lee. Recovery of thermal transport in atomic-layer-deposition-healed defective graphene. *Carbon*, 180:77–84, 2021. doi:10.1016/j.carbon.2021.04.098.

12. Hongying Wang, Yajuan Cheng, Zheyong Fan, Yangyu Guo, Zhongwei Zhang, Marc Bescond, Massahiro Nomura, Tapio Ala-Nissila, Sebastian Volz, and Shiyun Xiong. Anomalous thermal conductivity enhancement in low dimensional resonant nanostructures due to imperfections. *Nanoscale*, 13(22):10010–10015, 2021. doi:10.1039/D1NR01679B.

13. Xin Wu and Qiang Han. Phonon Thermal Transport across Multilayer Graphene/Hexagonal Boron Nitride van der Waals Heterostructures. *ACS Applied Materials & Interfaces*, 13(27):32564–32578, 2021. doi:10.1021/acsami.1c08275.

14. Xin Wu and Qiang Han. Semidefective Graphene/h-BN In-Plane Heterostructures: Enhancing Interface Thermal Conductance by Topological Defects. *The Journal of Physical Chemistry C*, 125(4):2748–2760, 2021. doi:10.1021/acs.jpcc.0c10387.

15. Zhongwei Zhang, Yangyu Guo, Marc Bescond, Jie Chen, Masahiro Nomura, and Sebastian Volz. Generalized decay law for particlelike and wavelike thermal phonons. *Physical Review B*, 103(18):184307, 2021. doi:10.1103/PhysRevB.103.184307.

9.6 2020

1. E. A. Bea, M. F. Carusela, A. Soba, A. G. Monastra, and A. M. Mancardo Viotti. Thermal conductance of structured silicon nanocrystals. *Modelling and Simulation in Materials Science and Engineering*, 28(7):075004, 2020. doi:10.1088/1361-651X/aba8eb.

2. Haikuan Dong, Zheyong Fan, Ping Qian, Tapio Ala-Nissila, and Yanjing Su. Thermal conductivity reduction in carbon nanotube by fullerene encapsulation: A molecular dynamics study. *Carbon*, 161:800–808, 2020. doi:10.1016/j.carbon.2020.01.114.

3. Bo Fu, Kevin D. Parrish, Hyun-Young Kim, Guihua Tang, and Alan J. H. McGaughey. Phonon confinement and transport in ultrathin films. *Physical Review B*, 101(4):045417, 2020. doi:10.1103/PhysRevB.101.045417.

4. Alexander J. Gabourie, Saurabh V. Suryavanshi, Amir Barati Farimani, and Eric Pop. Reduced thermal conductivity of supported and encased monolayer and bilayer MoS_2. *2D Materials*, 8(1):011001, 2020. doi:10.1088/2053-1583/aba4ed.

5. Xin Wu and Qiang Han. Thermal conductivity of defective graphene: an efficient molecular dynamics study based on graphics processing units. *Nanotechnology*, 31(21):215708, 2020. doi:10.1088/1361-6528/ab73bc.

6. Xin Wu and Qiang Han. Thermal conductivity of monolayer hexagonal boron nitride: From defective to amorphous. *Computational Materials Science*, 184:109938, 2020. doi:10.1016/j.commatsci.2020.109938.

9.7 2019

1. Zheyong Fan, Haikuan Dong, Ari Harju, and Tapio Ala-Nissila. Homogeneous nonequilibrium molecular dynamics method for heat transport and spectral decomposition with many-body potentials. *Phys. Rev. B*, 99:064308, Feb 2019. doi:10.1103/PhysRevB.99.064308.

2. Zheyong Fan, Yanzhou Wang, Xiaokun Gu, Ping Qian, Yanjing Su, and Tapio Ala-Nissila. A minimal Tersoff potential for diamond silicon with improved descriptions of elastic and phonon transport properties. *Journal of Physics: Condensed Matter*, 32(13):135901, 2019. doi:10.1088/1361-648X/ab5c5f.

3. Xiaokun Gu, Zheyong Fan, Hua Bao, and C. Y. Zhao. Revisiting phonon-phonon scattering in single-layer graphene. *Phys. Rev. B*, 100:064306, Aug 2019. doi:10.1103/PhysRevB.100.064306.

4. Leyla Isaeva, Giuseppe Barbalinardo, Davide Donadio, and Stefano Baroni. Modeling heat transport in crystals and glasses from a unified lattice-dynamical approach. *Nature communications*, 10(1):3853, 2019. doi:10.1038/s41467-019-11572-4.

5. Zhen Li, Shiyun Xiong, Charles Sievers, Yue Hu, Zheyong Fan, Ning Wei, Hua Bao, Shunda Chen, Davide Donadio, and Tapio Ala-Nissila. Influence of thermostatting on nonequilibrium molecular dynamics simulations of heat conduction in solids. *The Journal of Chemical Physics*, 12 2019. doi:10.1063/1.5132543.

6. Ke Xu, Alexander J. Gabourie, Arsalan Hashemi, Zheyong Fan, Ning Wei, Amir Barati Farimani, Hannu-Pekka Komsa, Arkady V. Krasheninnikov, Eric Pop, and Tapio Ala-Nissila. Thermal transport in MoS_2 from molecular dynamics using different empirical potentials. *Physical Review B*, 99(5):054303, 2019. doi:10.1103/PhysRevB.99.054303.

9.8 2018

1. Haikuan Dong, Zheyong Fan, Libin Shi, Ari Harju, and Tapio Ala-Nissila. Equivalence of the equilibrium and the nonequilibrium molecular dynamics methods for thermal conductivity calculations: From bulk to nanowire silicon. *Physical Review B*, 97(9):094305, 2018. doi:10.1103/PhysRevB.97.094305.

2. Haikuan Dong, Petri Hirvonen, Zheyong Fan, and Tapio Ala-Nissila. Heat transport in pristine and polycrystalline single-layer hexagonal boron nitride. *Physical Chemistry Chemical Physics*, 20(38):24602–24612, 2018. doi:10.1039/C8CP05159C.

3. Zheyong Fan, Ville Vierimaa, and Ari Harju. GPUQT: An efficient linear-scaling quantum transport code fully implemented on graphics processing units. *Computer Physics Communications*, 230:113–120, 2018. doi:10.1016/j.cpc.2018.04.013.

4. Petri Hirvonen, Gabriel Martine La Boissonière, Zheyong Fan, Cristian Vasile Achim, Nikolas Provatas, Ken R. Elder, and Tapio Ala-Nissila. Grain extraction and microstructural analysis method for two-dimensional poly and quasicrystalline solids. *Physical Review Materials*, 2(10):103603, 2018. doi:10.1103/PhysRevMaterials.2.103603.

5. Bohayra Mortazavi, Meysam Makaremi, Masoud Shahrokhi, Zheyong Fan, and Timon Rabczuk. N-graphdiyne two-dimensional nanomaterials: Semiconductors with low thermal conductivity and high stretchability. *Carbon*, 137:57–67, 2018. doi:10.1016/j.carbon.2018.04.090.

6. Ali Rajabpour, Zheyong Fan, and S. Mehdi Vaez Allaei. Inter-layer and intra-layer heat transfer in bilayer/monolayer graphene van der Waals heterostructure: Is there a Kapitza resistance analogous? *Applied Physics Letters*, 112(23):233104, 2018. doi:10.1063/1.5025604.

7. Ke Xu, Zheyong Fan, Jicheng Zhang, Ning Wei, and Tapio Ala-Nissila. Thermal transport properties of single-layer black phosphorus from extensive molecular dynamics simulations. *Modelling and Simulation in Materials Science and Engineering*, 26(8):085001, 2018. doi:10.1088/1361-651X/aae180.

9.9 2017

1. Khatereh Azizi, Petri Hirvonen, Zheyong Fan, Ari Harju, Ken R. Elder, Tapio Ala-Nissila, and S. Mehdi Vaez Allaei. Kapitza thermal resistance across individual grain boundaries in graphene. *Carbon*, 125:384–390, 2017. doi:10.1016/j.carbon.2017.09.059.

2. Zheyong Fan, Wei Chen, Ville Vierimaa, and Ari Harju. Efficient molecular dynamics simulations with many-body potentials on graphics processing units. *Computer Physics Communications*, 218:10–16, 2017. doi:10.1016/j.cpc.2017.05.003.

3. Zheyong Fan, Petri Hirvonen, Luiz Felipe C. Pereira, Mikko M. Ervasti, Ken R. Elder, Davide Donadio, Ari Harju, and Tapio Ala-Nissila. Bimodal Grain-Size Scaling of Thermal Transport in Polycrystalline Graphene from Large-Scale Molecular Dynamics Simulations. *Nano Letters*, 17(10):5919–5924, 2017. doi:10.1021/acs.nanolett.7b01742.

4. Zheyong Fan, Luiz Felipe C. Pereira, Petri Hirvonen, Mikko M. Ervasti, Ken R. Elder, Davide Donadio, Tapio Ala-Nissila, and Ari Harju. Thermal conductivity decomposition in two-dimensional materials: application to graphene. *Phys. Rev. B*, 95:144309, Apr 2017. doi:10.1103/PhysRevB.95.144309.

5. Zheyong Fan, Andreas Uppstu, and Ari Harju. Dominant source of disorder in graphene: charged impurities or ripples? *2D Materials*, 4(2):025004, jan 2017. doi:10.1088/2053-1583/aa529b.

6. Petri Hirvonen, Zheyong Fan, Mikko M. Ervasti, Ari Harju, Ken R. Elder, and Tapio Ala-Nissila. Energetics and structure of grain boundary triple junctions in graphene. *Scientific Reports*, 7(1):4754, 2017. doi:10.1038/s41598-017-04852-w.

7. Bohayra Mortazavi, Aurélien Lherbier, Zheyong Fan, Ari Harju, Timon Rabczuk, and Jean-Christophe Charlier. Thermal and electronic transport characteristics of highly stretchable graphene kirigami. *Nanoscale*, 9(42):16329–16341, 2017. doi:10.1039/C7NR05231F.

9.10 2016

1. Petri Hirvonen, Mikko M. Ervasti, Zheyong Fan, Morteza Jalalvand, Matthew Seymour, S. Mehdi Vaez Allaei, Nikolas Provatas, Ari Harju, Ken R. Elder, and Tapio Ala-Nissila. Multiscale modeling of polycrystalline graphene: A comparison of structure and defect energies of realistic samples from phase field crystal models. *Physical Review B*, 94(3):035414, 2016. doi:10.1103/PhysRevB.94.035414.

2. Bohayra Mortazavi, Zheyong Fan, Luiz Felipe C. Pereira, Ari Harju, and Timon Rabczuk. Amorphized graphene: A stiff material with low thermal conductivity. *Carbon*, 103:318–326, 2016. doi:10.1016/j.carbon.2016.03.007.

9.11 2015

1. Zheyong Fan, Luiz Felipe C. Pereira, Hui-Qiong Wang, Jin-Cheng Zheng, Davide Donadio, and Ari Harju. Force and heat current formulas for many-body potentials in molecular dynamics simulations with applications to thermal conductivity calculations. *Physical Review B*, 92(9):094301, 2015. doi:10.1103/PhysRevB.92.094301.

9.12 2013

1. Zheyong Fan, Topi Siro, and Ari Harju. Accelerated molecular dynamics force evaluation on graphics processing units for thermal conductivity calculations. *Computer Physics Communications*, 184(5):1414–1425, 2013. doi:10.1016/j.cpc.2013.01.008.

CHAPTER

TEN

GLOSSARY

ACE
atomic cluster expansion [Drautz2019]

ADF
angular distribution function

ADP
angular-dependent potential [Mishin2005]

ARDF
angular-dependent radial distribution function

BDP
Bussi-Donadio-Parrinello thermostat [Bussi2007b]

DOS
density of states

EAM
embedded atom method

EMD
equilibrium molecular dynamics

FCP
force constant potential

GK
Green-Kubo

GPU
graphics processing unit

GKMA
Green-Kubo modal analysis [Lv2016]

HAC
heat current auto-correlation

HNEMA
homogeneous non-equilibrium modal analysis [Gabourie2021]

HNEMD
The homogeneous non-equilibrium molecular dynamics

HNEMDEC
homogeneous non-equilibrium molecular dynamics Evans-Cummings algorithm

ILP
interlayer potential for van der Waals materials [Ouyang2018] [Ouyang2020]

LJ
Lennard-Jones potential

LSQT
linear-scaling quantum transport

MC
Monte Carlo

MD
molecular dynamics

MSD
mean square displacement

MSST
Multi-scale shock technique [Reed2003]

MTTK
Martyna-Tuckerman-Tobias-Klein integrator [Martyna1994]

NEMD
non-equilibrium molecular dynamics

NEP
neuroevolution potential

NHC
Nose-Hoover chain thermostat [Tuckerman2010]

NN
neural network

PDOS
phonon density of states

PIMD
path-integral molecular dynamics

RDF
radial distribution function

RMSE
root-mean-square error

RPMD
ring-polymer molecular dynamics

RTC
running thermal conductivity

SCR
stochastic cell rescaling barostat [Bernetti2020]

SDC
self-diffusion coefficient

SGC
semi-grand canonical

SHC
spectral heat current

SNES
separable natural evolution strategy [Schaul2011]

SVR
stochastic velocity rescaling thermostat [Bussi2007b]

SW
Stillinger-Weber potential [Stillinger1985]

TRPMD
thermostatted ring-polymer molecular dynamics

VAC
velocity auto-correlation

VCSGC
variance-constrained semi-grand canonical [Sadigh2012a] [Sadigh2012b]

ZBL
universal potential by Ziegler, Biersack, and Littmark [Ziegler1985]

CHAPTER

ELEVEN

INDEX

BIBLIOGRAPHY

[Pun2015] G. P. Pun, K. A. Darling, L. J. Kecskes, and Y. Mishin, "Angular-dependent interatomic potential for the Cu–Ta system and its application to structural stability of nano-crystalline alloys," Acta Mater. 100, 377 (2015).

[Starikov2018] S. V. Starikov, L. N. Kolotova, A. Y. Kuksin, D. E. Smirnova, and V. I. Tseplyaev, "Atomistic simulation of cubic and tetragonal phases of U-Mo alloy: Structure and thermodynamic properties," J. Nucl. Mater. 499, 451 (2018).

[Bernetti2020] Mattia Bernetti and Giovanni Bussi
Pressure control using stochastic cell rescaling
J. Chem. Phys. **153**, 114107 (2020)
DOI: 10.1063/5.0020514

[Berendsen1984] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak
Molecular dynamics with coupling to an external bath
J. Chem. Phys. **81**, 3684 (1984)
DOI: 10.1063/1.448118

[Bitzek2006] Erik Bitzek, Pekka Koskinen, Franz Gähler, Michael Moseler, and Peter Gumbsch
Structural Relaxation Made Simple
Phys. Rev. Lett. **97**, 170201 (2006)
DOI: 10.1103/PhysRevLett.97.170201

[Brorsson2021] Joakim Brorsson, Arsalan Hashemi, Zheyong Fan, Erik Fransson, Fredrik Eriksson, Tapio Ala-Nissila, Arkady V. Krasheninnikov, Hannu-Pekka Komsa, and Paul Erhart
Efficient calculation of the lattice thermal conductivity by atomistic simulations with ab-initio accuracy
Advanced Theory and Simulations **4**, 2100217 (2021)
DOI: 10.1002/adts.202100217

[Bussi2007a] Giovanni Bussi and Michele Parrinello
Accurate sampling using Langevin dynamics
Phys. Rev. E **75**, 056707 (2007)
DOI: 10.1103/PhysRevE.75.056707

[Bussi2007b] Giovanni Bussi, Davide Donadio, and Michele Parrinello
Canonical sampling through velocity rescaling
J. Chem. Phys. **126**, 014101 (2007)
DOI: 10.1063/1.2408420

[Ceriotti2010] Michele Ceriotti, Michele Parrinello, Thomas E. Markland, and David E. Manolopoulos
Efficient stochastic thermostatting of path integral molecular dynamics

J. Chem. Phys. **133**, 124104 (2010)
DOI: 10.1063/1.3489925

[Craig2004] Ian R. Craig and David E. Manolopoulos
Quantum statistics and classical mechanics: Real time correlation functions from ring polymer molecular dynamics
J. Chem. Phys. **121**, 3368 (2004)
DOI: 10.1063/1.1777575

[Dai2006] X. D. Dai, Y. Kong, J. H. Li, and B. X. Liu
Extended Finnis–Sinclair potential for bcc and fcc metals and alloys
J. Phys.: Condens. Matter **18**, 4527 (2006)
DOI: 10.1088/0953-8984/18/19/008

[Drautz2019] Ralf Drautz
Atomic cluster expansion for accurate and transferable interatomic potentials
Phys. Rev. B **99**, 014104 (2019)
DOI: 10.1103/PhysRevB.99.014104

[Eriksson2019] Fredrik Eriksson, Erik Fransson, and Paul Erhart
The Hiphive Package for the Extraction of High-Order Force Constants by Machine Learning
Advanced Theory and Simulations, **2**, 1800184 (2019)
DOI: 10.1002/adts.201800184

[Fan2015] Zheyong Fan, Luiz Felipe C. Pereira, Hui-Qiong Wang, Jin-Cheng Zheng, Davide Donadio, and Ari Harju
Force and heat current formulas for many-body potentials in molecular dynamics simulations with applications to thermal conductivity calculations
Phys. Rev. B **92**, 094301 (2015)
DOI: 10.1103/PhysRevB.92.094301

[Fan2017] Zheyong Fan, Luiz Felipe C. Pereira, Petri Hirvonen, Mikko M. Ervasti, Ken R. Elder, Davide Donadio, Tapio Ala-Nissila, and Ari Harju
Thermal conductivity decomposition in two-dimensional materials: Application to graphene
Phys. Rev. B **95**, 144309 (2017)
DOI: 10.1103/PhysRevB.95.144309

[Fan2019] Zheyong Fan, Haikuan Dong, Ari Harju, and Tapio Ala-Nissila
Homogeneous nonequilibrium molecular dynamics method for heat transport and spectral decomposition with many-body potentials
Phys. Rev. B **99**, 064308 (2019)
DOI: 10.1103/PhysRevB.99.064308

[Fan2020] Zheyong Fan, Yanzhou Wang, Xiaokun Gu, Ping Qian, Yanjing Su, and Tapio Ala-Nissila
A minimal Tersoff potential for diamond silicon with improved descriptions of elastic and phonon transport properties
J. Phys.: Condens. Matter **32**, 135901 (2020)
DOI: 10.1088/1361-648X/ab5c5f

[Fan2021] Zheyong Fan, Zezhu Zeng, Cunzhi Zhang, Yanzhou Wang, Keke Song, Haikuan Dong, Yue Chen, and Tapio Ala-Nissila
Neuroevolution machine learning potentials: Combining high accuracy and low cost in atomistic simulations and application to heat transport
Phys. Rev. B. **104**, 104309 (2021)
DOI: 10.1103/PhysRevB.104.104309

[Fan2021b] Zheyong Fan, Jose Hugo Garcia, Aron W Cummings, Jose Eduardo Barrios-Vargas, Michel Panhans, Ari Harju, Frank Ortmann, and Stephan Roche
Linear scaling quantum transport methodologies
Physics Reports **903**, 1 (2021)
DOI: 10.1016/j.physrep.2020.12.001

[Fan2022a] Zheyong Fan
Improving the accuracy of the neuroevolution machine learning potentials for multi-component systems
Journal of Physics: Condensed Matter **34**, 125902 (2022)
DOI: 10.1088/1361-648X/ac462b

[Fan2022b] Zheyong Fan, Yanzhou Wang, Penghua Ying, Keke Song, Junjie Wang, Yong Wang, Zezhu Zeng, Ke Xu, Eric Lindgren, J. Magnus Rahm, Alexander J. Gabourie, Jiahui Liu, Haikuan Dong, Jianyang Wu, Yue Chen, Zheng Zhong, Jian Sun, Paul Erhart, Yanjing Su, and Tapio Ala-Nissila
GPUMD: A package for constructing accurate machine-learned potentials and performing highly efficient atomistic simulations
Journal of Chemical Physics **157**, 114801 (2022)
DOI: 10.1063/5.0106617

[Freitas2016] Rodrigo Freitas, Mark Asta, Maurice de Koning
Nonequilibrium free-energy calculation of solids using LAMMPS
Computational Materials Science, **112**, 333 (2016)
DOI: 10.1016/j.commatsci.2015.10.050

[Gabourie2021] Alexander J. Gabourie, Zheyong Fan, Tapio Ala-Nissila, and Eric Pop
Spectral Decomposition of Thermal Conductivity: Comparing Velocity Decomposition Methods in Homogeneous Molecular Dynamics Simulations
Phys. Rev. B **103**, 205421 (2021)
DOI: 10.1103/PhysRevB.103.205421

[Grimme2010] Stefan Grimme, Jens Antony, Stephan Ehrlich, and Helge Krieg
A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu
J. Chem. Phys. **132**, 154104 (2010)
DOI: 10.1063/1.3382344

[Grimme2011] Stefan Grimme, Stephan Ehrlich, and Lars Goerigk
Effect of the damping function in dispersion corrected density functional theory
Journal of Computational Chemistry **32**, 1456 (2011)
DOI: 10.1002/jcc.21759

[Guénolé2020] Julien Guénolé, Wolfram G. Nöhring, Aviral Vaid, Frédéric Houllé, Zhuocheng Xie, Aruna Prakash, and Erik Bitzek
Assessment and optimization of the fast inertial relaxation engine (fire) for energy minimization in atomistic simulations and its implementation in lammps
Computational Materials Science **175**, 109584 (2020)
DOI: 10.1016/j.commatsci.2020.109584

[Hoover1996] William G. Hoover and Brad Lee Holian
Kinetic moments method for the canonical ensemble distribution
Physics Letters A, **211**, 253-257 (1996)
DOI: *10.1016/0375-9601(95)00973-6 <https://doi.org/10.1016/0375-9601(95)00973-6>*

[Leimkuhler2013] Benedict Leimkuhler and Charles Matthews

Rational construction of stochastic numerical methods for molecular sampling
Applied Mathematics Research eXpress **2013**, 34 (2013)
DOI: 10.1093/amrx/abs010

[Li2019] Zhen Li, Shiyun Xiong, Charles Sievers, Yue Hu, Zheyong Fan, Ning Wei, Hua Bao, Shunda Chen, Davide Donadio, and Tapio Ala-Nissila
Influence of Thermostatting on Nonequilibrium Molecular Dynamics Simulations of Heat Conduction in Solids
J. Chem. Phys. **151**, 234105 (2019)
DOI: 10.1063/1.5132543

[Lv2016] Wei Lv and Asegun Henry
Direct calculation of modal contributions to thermal conductivity via Green-Kubo modal analysis
New J. Phys. **18**, 013028 (2016)
DOI: 10.1088/1367-2630/18/1/013028

[Martyna1994] Glenn J. Martyna, Douglas J. Tobias, and Michael L. Klein
Constant pressure molecular dynamics algorithms
The Journal of Chemical Physics, **101**, 4177-4189 (1994)
DOI: *10.1063/1.467468 <https://doi.org/10.1063/1.467468>*

[Mishin2005] Y. Mishin, M. J. Mehl, and D. A. Papaconstantopoulos
Phase stability in the Fe–Ni system: Investigation by first-principles calculations and atomistic simulations
Acta Materialia **53**, 4029 (2005)
DOI: *10.1016/j.actamat.2005.05.001 <https://doi.org/10.1016/j.actamat.2005.05.001>*

[Parrinello1981] M. Parrinello and A. Rahman
Polymorphic transitions in single crystals: A new molecular dynamics method
Journal of Applied Physics, **52**, 7182-7190 (1981)
DOI: *10.1063/1.328693 <https://doi.org/10.1063/1.328693>*

[Rahm2021] J. Magnus Rahm, Joakim Löfgren, Erik Fransson, and Paul Erhart
A tale of two phase diagrams: Interplay of ordering and hydrogen uptake in Pd–Au–H
Acta Materialia, **211**, 116893 (2021)
DOI: *10.1016/j.actamat.2021.116893 <https://doi.org/10.1016/j.actamat.2021.116893>*

[Ravelo2004] Ravelo, R. and Holian, B. L. and Germann, T. C. and Lomdahl, P. S.
Constant-stress Hugoniostat method for following the dynamical evolution of shocked matter
Phys. Rev. B. **70**, 014103 (2004)
DOI: 10.1103/PhysRevB.70.014103

[Reed2003] Evan J. Reed, Laurence E. Fried, and J. D. Joannopoulos
A Method for Tractable Dynamical Studies of Single and Double Shock Compression
Phys. Rev. Lett. **90**, 235503 (2003)
DOI: 10.1103/PhysRevLett.90.235503

[Rossi2014] Mariana Rossi, Michele Ceriotti, and David E. Manolopoulos
How to remove the spurious resonances from ring polymer molecular dynamics
J. Chem. Phys. **140**, 234116 (2014)
DOI: 10.1063/1.4883861

[Sadigh2012a] Babak Sadigh, Paul Erhart, Alexander Stukowski, Alfredo Caro, Enrique Martinez, and Luis Zepeda-Ruiz

Scalable parallel Monte Carlo algorithm for atomistic simulations of precipitation in alloys
Phys. Rev. B **85**, 184203 (2012)
DOI: 10.1103/PhysRevB.85.184203

[Sadigh2012b] Babak Sadigh and Paul Erhart
Calculation of excess free energies of precipitates via direct thermodynamic integration across phase boundaries
Phys. Rev. B **86**, 134204 (2012)
DOI: 10.1103/PhysRevB.86.134204

[Schaul2011] T. Schaul, T. Glasmachers, and J. Schmidhuber
High dimensions and heavy tails for natural evolution strategies
In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation
GECCO '11 (Association for Computing Machinery), New York, USA (2011), pp. 845–852
DOI: 10.1145/2001576.2001692

[Song2024] Keke Song, Rui Zhao, Jiahui Liu, Yanzhou Wang, Eric Lindgren, Yong Wang, Shunda Chen, Ke Xu, Ting Liang, Penghua Ying, Nan Xu, Zhiqiang Zhao, Jiuyang Shi, Junjie Wang, Shuang Lyu, Zezhu Zeng, Shirong Liang, Haikuan Dong, Ligang Sun, Yue Chen, Zhuhua Zhang, Wanlin Guo, Ping Qian, Jian Sun, Paul Erhart, Tapio Ala-Nissila, Yanjing Su, and Zheyong Fan
General-purpose machine-learned potential for 16 elemental metals and their alloys
Nature Communications *15*, 10208 (2024)
DOI: 10.1038/s41467-024-54554-x

[Tersoff1988] Jerry Tersoff
New empirical approach for the structure and energy of covalent systems
Phys. Rev. B *37*, 6991 (1988)
DOI: 10.1103/PhysRevB.37.6991

[Tersoff1989] Jerry Tersoff
Modeling solid-state chemistry: Interatomic potentials for multicomponent systems
Phys. Rev. B **39**, 5566(R) (1989)
DOI: 10.1103/PhysRevB.39.5566

[Tuckerman2010] Mark E. Tuckerman
Statistical Mechanics: Theory and Molecular Simulation (Oxford Graduate Texts)
1st Edition, Oxford University Press (2010)

[Zhou2004] X. W. Zhou, R. A. Johnson, and H. N. G. Wadley
Misfit-energy-increasing dislocations in vapor-deposited CoFe/NiFe multilayers
Phys. Rev. B **69**, 144113 (2004)
DOI: 10.1103/PhysRevB.69.144113

[Ziegler1985] J. F. Ziegler, J. P. Biersack, and U. Littmark
In *The Stopping and Range of Ions in Matter*, volume 1
New York, 1985. Pergamon. ISBN 0-08-022053-3

[Koning2001] Maurice de Koning, Alex Antonelli, and Sidney Yip
Single-simulation determination of phase boundaries: A dynamic Clausius–Clapeyron integration method
J. Chem. Phys. **115**, 11025–11035 (2001)
DOI: 10.1063/1.1420486

[Cajahuaringa2022] Samuel Cajahuaringa and Alex Antonelli
Non-equilibrium free-energy calculation of phase-boundaries using LAMMPS

Computational Materials Science **207**, 111275 (2022)
DOI: 10.1016/j.commatsci.2022.111275

[Podryabinkin2023] Evgeny Podryabinkin, Kamil Garifullin, Alexander Shapeev, and Ivan Novikov
MLIP-3: Active learning on atomic environments with moment tensor potentials
J. Chem. Phys. **159**, 084112 (2023)
DOI: 10.1063/5.0155887

[Lysogorskiy2023] Yury Lysogorskiy, Anton Bochkarev, Matous Mrovec, and Ralf Drautz
Active learning strategies for atomic cluster expansion models
Phys. Rev. M **7**, 043801 (2023)
DOI: 10.1103/PhysRevMaterials.7.043801

[Ouyang2018] Wengen Ouyang, Davide Mandelli, Michael Urbakh, and Oded Hod
Nanoserpents: graphene nanoribbon motion on two-dimensional hexagonal materials
Nano Lett. **18**, 6009-6016 (2018)
DOI: 10.1021/acs.nanolett.8b02848

[Ouyang2020] Wengen Ouyang, Ido Azuri, Davide Mandelli, Alexandre Tkatchenko, Leeor Kronik, Michael Urbakh, and Oded Hod
Mechanical and tribological properties of layered materials under high pressure: assessing the importance of many-body dispersion effects
J. Chem. Theory Comput. **16(1)**, 666-676 (2020)
DOI: 10.1021/acs.jctc.9b00908

[Stillinger1985] Frank H. Stillinger and Thomas A. Weber
Computer simulation of local order in condensed phases of silicon
Phys. Rev. B **31**, 5262-5271 (1985)
DOI: 10.1103/PhysRevB.31.5262

[Jiang2015] Jinwu Jiang
Parametrization of Stillinger-Weber potential based on valence force field model: application to single-layer MoS2 and black phosphorus
Nanotechnology **26**, 315706 (2015)
DOI: 10.1088/0957-4484/26/31/315706

[Jiang2019] Jinwu Jiang
Misfit strain-induced buckling for transition-metal dichalcogenide lateral heterostructures: a molecular dynamics study
Acta Mech. Solida Sin. **32**, 17-28 (2019)
DOI: 10.1007/s10338-018-0049-z

[Leite2016] Rodolfo Paula Leite, Rodrigo Freitas, Rodolfo Azevedo and Maurice de Koning
The Uhlenbeck-Ford model: Exact virial coefficients and application as a reference system in fluid-phase free-energy calculations
J. Chem. Phys. 145, 194101 (2016)
DOI: 10.1063/1.4967775

[Leite2019] Rodolfo Paula Leite and Maurice de Koning
Nonequilibrium free-energy calculations of fluids using LAMMPS
Computational Materials Science, Volume 159, 316-326 (2019)
DOI: 10.1016/j.commatsci.2018.12.029

[Menon2021] Sarath Menon, Yury Lysogorskiy, Jutta Rogal and Ralf Drautz
Automated free-energy calculation from atomistic simulations

Phys. Rev. Materials 5, 103801, (2021)
DOI: 10.1103/PhysRevMaterials.5.103801

[Steinhardt1983] Steinhardt, P. J., Nelson, D. R., & Ronchetti, M.
Bond-orientational order in liquids and glasses
Physical Review B, 28(2), 784, (1983)
DOI: 10.1103/PhysRevB.28.784

[Mickel2013] Mickel, W., Kapfer, S. C., Schröder-Turk, G. E., & Mecke, K. (2013).
Shortcomings of the bond orientational order parameters for the analysis of disordered particulate matter
The Journal of chemical physics, 138(4), (2013).
DOI: 10.1063/1.4774084

[Bu2025] Hekai Bu, Wenwu Jiang, Penghua Ying, Ting Liang, Zheyong Fan, and Wengen Ouyang
Accurate modeling of LEGO-like vdW heterostructures: integrating machine learned with anisotropic interlayer potentials
arXiv, 2504, 12985 (2025)
DOI: 10.48550/arXiv.2504.12985

INDEX

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

Z