
GPUMD documentation

Version 5.6

The GPUMD developer team

Jul 21, 2026

1	Introduction	3
1.1	Python interfaces	3
1.2	GitHub discussions	3
1.3	Discussion groups	3
2	Installation	5
2.1	Download	5
2.2	Prerequisites	5
2.3	Compilation	5
2.4	Building for AMD GPUs (ROCm/HIP)	6
2.5	Examples	6
2.6	GNEP setup	6
2.7	NetCDF setup	7
2.8	PLUMED setup	8
2.9	DP potential support	8
2.10	NNAP support	14
3	Theoretical background	19
3.1	Forces and stresses	19
3.2	Ensembles	20
3.3	Heat current	23
3.4	Heat transport	24
4	Interatomic potentials	29
4.1	Tersoff potential (1988)	29
4.2	Tersoff potential (1989)	31
4.3	Tersoff mini-potential	32
4.4	Lennard-Jones potential	33
4.5	Embedded atom method	35
4.6	Force constant potential	37
4.7	Neuroevolution potential	39
4.8	Hybrid NEP+ILP potential	41
4.9	Hybrid SW+ILP potential	44
4.10	Hybrid Tersoff+ILP potential	46
4.11	Angular Dependent Potential (ADP)	48
5	gpumd executable	53
5.1	Input files	53
5.2	Input parameters	58
5.3	Output files	133

6	nep executable	153
6.1	Input files	153
6.2	Input parameters	156
6.3	Output files	166
7	Credits	171
7.1	Acknowledgments	171
8	Bibliography	173
9	Publications	175
9.1	2026	175
9.2	2025	186
9.3	2024	197
9.4	2023	202
9.5	2022	205
9.6	2021	207
9.7	2020	208
9.8	2019	208
9.9	2018	209
9.10	2017	209
9.11	2016	210
9.12	2015	210
9.13	2013	210
10	Glossary	211
11	Index	215
	Bibliography	217
	Index	225

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*.

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

INSTALLATION

2.1 Download

The source code is hosted on [github](#).

2.2 Prerequisites

Make

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`.

Alternatively, **GPUMD** can be built for AMD GPUs through ROCm/HIP (see *Building for AMD GPUs* below); this requires a ROCm installation.

CMake (pre-release)

To compile (and run) **GPUMD** one requires an Nvidia GPU card and CUDA toolkit. On Linux systems, one also needs a C++ compiler supporting the C++17 standard. On Windows systems, one also needs the `cl.exe` compiler from Microsoft Visual Studio and CMake 3.24 or newer.

2.3 Compilation

Make

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.

CMake (pre-release)

Configure and compile from the project root:

```
cmake -S . -B build
cmake --build build --config Release --parallel
```

The executables (`gpumd` and `nep`) will be placed in the `build` directory. Omit `--parallel` to fall back to serial compilation.

To target a specific GPU architecture, add `-D` at configure time:

```
-D CMAKE_CUDA_ARCHITECTURES=value # 80, 90, ... (default native)
```

2.4 Building for AMD GPUs (ROCm/HIP)

Make

GPUMD also runs on AMD GPUs through ROCm/HIP. In the `src` directory, build with the HIP makefile instead of the default one:

```
make -f makefile.hip
```

This uses `hipcc` and links `rocThrust/rocPRIM`, `hipBLAS`, `hipSOLVER`, and `hipFFT`, so it requires a ROCm installation. The target GPU architecture defaults to `gfx90a`; build for another AMD GPU by setting `HIP_ARCH`:

```
make -f makefile.hip HIP_ARCH=gfx1100
```

CMake (pre-release)

CMake build for AMD GPUs is not yet supported.

2.5 Examples

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

2.6 GNEP setup

GNEP stands for a method of training NEP models using analytical Gradients (G stands for Gradients). See the [implementation paper](#) for details.

Make

To compile the `gnep` executable, run `make gnep` in the `src` directory.

CMake (pre-release)

To compile the `gnep` executable, configure with CMake as described in the [Compilation](#) section above, then run:

```
cmake --build build --target gnep
```

The usage of the `gnep` executable is similar to that of the `nep` executable. The major difference is that training hyperparameters are written in `gnep.in` instead of `nep.in`. Below we use an explicit example with default parameters (except for the `type` keyword) to illustrate the inputs in `gnep.in`:

```
type          2 Ge Se      # same usage as in nep.in
prediction    0            # same usage as in nep.in
cutoff        8 4          # same usage as in nep.in
n_max         4 4          # same usage as in nep.in
basis_size    8 8          # same usage as in nep.in
l_max         4            # same usage as in nep.in but does not support 4-body and 5-
↪body descriptors
```

(continues on next page)

(continued from previous page)

```

neuron      30          # same usage as in nep.in
lambda_e    1.0         # same usage as in nep.in
lambda_f    2.0         # same usage as in nep.in but defaults to 2
lambda_v    0.1         # same usage as in nep.in
start_lr    1e-3        # new keyword to set the starting learning rate, which should
↳ be a non-negative floating-point number
stop_lr     1e-7        # new keyword to set the stopping learning rate, which should
↳ be a non-negative floating-point number
weight_decay 0.0        # new keyword to set the weight decay parameter, which should
↳ be a non-negative floating-point number
batch       2           # same usage as in nep.in but favors small values
epoch       50          # one epoch equals #structures/#batchsize training steps

```

2.7 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.

2.8 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*.

2.9 DP potential support

2.9.1 Program introduction

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

Supported model formats

- `.pb`: TensorFlow backend, generated by `dp --tf freeze`
- `.pth`: PyTorch TorchScript backend (DPA2/DPA3), generated by `dp --pt freeze`
- `.pt2`: PyTorch AOTInductor backend (DPA4), generated by `dp --pt freeze`

Note: The DeePMD-kit C++ API auto-detects the backend from the file extension. No changes to `run.in` are needed when switching between backends.

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.

2.9.2 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.

2.9.3 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
```

2.9.4 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_DEEPM`. So add it to `CFLAGS`:

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

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
```

GPUMD-DP with PyTorch backend

DeePMD-kit C++ interface installation

The PyTorch backend requires the DeePMD-kit C++ interface libraries (`libdeepmd_cc`, `libdeepmd_c`) and PyTorch (LibTorch). Two installation methods are available:

Method 1: pip install (recommended)

Install DeePMD-kit with PyTorch support:

```
>> $ pip install deepmd-kit[torch]
```

After installation, the C++ interface files are located under the Python package directory:

```
>> $ python -c "import deepmd; print(deepmd.__path__[0] + '/lib')"
# Output example: /path/to/python/site-packages/deepmd/lib
# Contains: include/deepmd/*.h, libdeepmd_cc.so, libdeepmd_c.so
```

LibTorch libraries are bundled with PyTorch:

```
>> $ python -c "import torch; print(torch.__path__[0] + '/lib')"  
# Contains: libtorch.so, libtorch_cpu.so, libc10.so, libtorch_cuda.so, etc.
```

Verify the installation:

```
>> $ dp --version  
>> $ python -c "import torch; print(torch.__version__, torch.cuda.is_available())"
```

Method 2: Build from source

Clone and compile the DeePMD-kit C++ interface with PyTorch backend enabled:

```
>> $ git clone https://github.com/deepmodeling/deepmd-kit.git  
>> $ cd deepmd-kit/source  
>> $ mkdir build && cd build  
>> $ cmake -DENABLE_PYTORCH=TRUE \  
           -DUSE_CUDA_TOOLKIT=TRUE \  
           -DCMAKE_INSTALL_PREFIX=/path/to/install \  
           ../..  
>> $ make -j$(nproc) && make install
```

The installed files will be at:

```
/path/to/install/include/deepmd/  # Header files  
/path/to/install/lib/            # libdeepmd_cc.so, libdeepmd_c.so
```

Note: Method 1 is sufficient for most users. Method 2 is needed only if you require a custom build (e.g., specific CUDA version, debug symbols, or a development branch of DeePMD-kit).

Compile GPUMD

With the C++ interface installed, configure the GPUMD makefile. Enable DP support with `-DUSE_DEEPMD`:

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

Link against the DeePMD-kit C interface and PyTorch/LibTorch libraries:

```
INC += -I$(shell python -c "import deepmd; print(deepmd.__path__[0])")/lib/include/deepmd  
LDFLAGS += -ldeepmd_cc -ldeepmd_c  
LDFLAGS += $(shell python -c "import torch; print(' '.join(['-L'+p for p in torch.__path_  
→_] + ['-L'+p+'/lib' for p in torch.__path_]))")  
LDFLAGS += -ltorch -ltorch_cpu -lc10
```

For GPU support, also link CUDA-related torch libraries:

```
LDFLAGS += -ltorch_cuda -lc10_cuda
```

Then compile:

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


Freeze a PyTorch model

To obtain a frozen model for the PyTorch backend:

```
>> $ dp --pt freeze -o frozen_model.pth # DPA2/DPA3 -> .pth
>> $ dp --pt freeze -o frozen_model.pth # DPA4 -> produces .pt2 automatically
```

For pre-trained universal models (e.g., DPA-2.3.1 from AIS Square), download from <https://aissquare.com> and freeze as above.

The type map can be extracted from the frozen model:

```
>> $ python -c "from deepmd.infer import DeepPot; print(' '.join(DeepPot('frozen_model.
↳pt2').get_type_map()))"
```

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 setting file specifies the number of atom types and their names. For example, the dp_settings.txt for a water system:

```
dp 2 0 H
```

Use in run.in:

```
potential dp_settings.txt frozen_model.pb # TensorFlow backend
potential dp_settings.txt frozen_model.pth # PyTorch TorchScript (DPA2/DPA3)
potential dp_settings.txt frozen_model.pt2 # PyTorch AOTInductor (DPA4)
```

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>

2.10 NNAP support

2.10.1 Program introduction

This is the beginning of **GPUMD** support for NNAP in jse.

jse (Java Simulation Environment, GitHub: [liqa1024/jse](https://github.com/liqa1024/jse)) is a high-performance, extensible simulation framework for atomistic and materials modeling. It provides the latest support for NNAP (Neural-Network Atomic Potentials), including direct execution, *ASE*, *LAMMPS*, and **GPUMD** as described in this document. **GPUMD** invokes dedicated NNAP support in jse via JNI to run simulations with NNAP potentials.

2.10.2 Necessary instructions

- This is a development version.
- The NNAP potential file (`.json` / `.jnn`) and the corresponding GPUMD setting file (`.txt`) must be correctly prepared before running *GPUMD-NNAP*.
- The element order in setting file must be consistent with that in the NNAP potential file.

2.10.3 Installation dependencies

To compile and run *GPUMD-NNAP*, the following requirements must be satisfied:

- The new version ($\geq 4.1.0$) of jse must be installed and able to pass JNI build `jse --jnibuild`.
- The installation requirements of **GPUMD** itself must be met, including a working CUDA compiler and a compatible NVIDIA GPU.

2.10.4 Installation details

If you have any questions, please contact Qing'an Li (liqa1024@vip.qq.com) and Ke Xu (twtdq@qq.com).

This section describes how to compile **GPUMD** with NNAP support on Linux. For an introduction to NNAP/jse and complete installation and usage guides, refer to [liqa1024/jse](https://github.com/liqa1024/jse) and [liqa1024/jse-skill](https://github.com/liqa1024/jse-skill).

Prepare the environment

Check the system environment:

```
nvcc --version
```

Set the installation directory:

```
export install_dir="$HOME/software/GPUMD-NNAP"
mkdir -p ${install_dir}
cd ${install_dir}
```

Install jse

Install jse from the dev channel to obtain the latest development version (for linux):

```
bash <(curl -fsSL https://raw.githubusercontent.com/liqa1024/jse/dev/scripts/get.sh)
```

Optionally, manually run the JNI build to detect any potential environment issues in advance:

```
jse --jnibuild
jse -t 'jse.gpu.CudaCore.InitHelper.init()'
```

Configure and compile GPUMD-NNAP

Make

Detect the required paths using jse and export them as environment variables:

```
export JSE_JAR_PATH=$(jse -t 'println(jse.code.OS.JAR_PATH)')
export JVM_INCLUDE=$(jse -t 'println(jse.clib.JVM.INCLUDE_DIR)')
export JVM_LLIB_DIR=$(jse -t 'println(jse.clib.JVM.LLIB_DIR)')
```

Enable NNAP support and add the jse class path by modifying CFLAGS:

```
CFLAGS = -std=c++14 -O3 -arch=sm_60 -DUSE_NNAP -DJVM_CLASS_PATH=\"-Djava.class.path=
↳$(JSE_JAR_PATH)\"
```

Add the JVM header paths by modifying INC:

```
INC = -I./ \
      -I$(JVM_INCLUDE) \
      -I$(JVM_INCLUDE)/linux
```

Here \$(JVM_INCLUDE)/linux corresponds to Linux; for other systems, replace it with the corresponding platform directory.

Add the JVM library path and runtime path by modifying LIBS:

```
LIBS = -lcublas -lcusolver -lcufft \
      -L$(JVM_LLIB_DIR) -ljvm \
      -Xlinker -rpath -Xlinker $(JVM_LLIB_DIR)
```

Here, JSE_JAR_PATH, JVM_INCLUDE, and JVM_LLIB_DIR should be replaced by the actual paths obtained from the jse commands above, or exported as environment variables before running make.

Compile the executable files:

```
make -j
ls gpumd nep
```

If the compilation is successful, the executables gpumd and nep should be generated.

CMake (pre-release)

CMake uses PKG_NNAP to control NNAP support:

```
-D PKG_NNAP=yes
```

Add environment variable to import jse jar path:

```
export JSE_JAR_PATH=$(jse -t 'println(jse.code.OS.JAR_PATH)')
```

In Windows, you need to additionally specify the JVM library path (PowerShell):

```
$env:JSE_JAR_PATH = "$(jse -t 'println(jse.code.OS.JAR_PATH)')"
```

```
$env:JVM_LIB_PATH = "$(jse -t 'println(jse.clib.JVM.LIB_PATH)')"
```

2.10.5 Run the NNAP test

In the GPUMD input file, use the NNAP potential as follows:

```
potential nnap.txt CuZr-sphs.json
```

An example GPUMD setting file nnap.txt file is:

```
nnap 2 Cu Zr
```

The element order in nnap.txt must be consistent with that in the NNAP potential file CuZr-sphs.json.

For example, if the NNAP potential file uses the element order Zr Cu, then nnap.txt should be written as:

```
nnap 2 Zr Cu
```

Run the test with:

```
gpumd
```

2.10.6 Notice

The element list in the NNAP setting file and the NNAP potential file must be the same. Otherwise, the atom types will be mapped incorrectly during the simulation.

References

- jse: <https://github.com/liqa1024/jse>
- jse-skill: <https://github.com/liqa1024/jse-skill>
- jsex-NNAP: <https://github.com/liqa1024/jsex-NNAP>

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 :

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

Here, $\mathbf{r}_{ij} \equiv \mathbf{r}_j - \mathbf{r}_i$ 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]:

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

where

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

Here, $\partial U_i / \partial \mathbf{r}_{ij}$ 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

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

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

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

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

where

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

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

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

starting from the initial phase point

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

The time interval between two time points $\Delta t = t_1 - t_0 = t_2 - t_1 = \dots$ 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 ensemble is also called the micro-canonical ensemble. We use the velocity-Verlet integration method with the following equations:

$$\begin{aligned} \mathbf{v}_i(t_{m+1}) &\approx \mathbf{v}_i(t_m) + \frac{\mathbf{F}_i(t_m) + \mathbf{F}_i(t_{m+1})}{2m_i} \Delta t \\ \mathbf{r}_i(t_{m+1}) &\approx \mathbf{r}_i(t_m) + \mathbf{v}_i(t_m) \Delta t + \frac{1}{2} \frac{\mathbf{F}_i(t_m)}{m_i} (\Delta t)^2. \end{aligned}$$

Here, $\mathbf{v}_i(t_m)$ is the velocity vector of particle i at time t_m . $\mathbf{r}_i(t_m)$ is the position vector of particle i at time t_m . $\mathbf{F}_i(t_m)$ 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:

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

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:

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

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):

$$\begin{aligned} \frac{d\mathbf{r}_i}{dt} &= \frac{\mathbf{p}_i}{m_i} \\ \frac{d\mathbf{p}_i}{dt} &= \mathbf{F}_i - \frac{\pi_0}{Q_0} \mathbf{p}_i. \end{aligned}$$

Here, $\mathbf{r}_i$ is the position of atom i . $\mathbf{p}_i$ is the momentum of atom i . m_i is the mass of atom i . $\mathbf{F}_i$ is the total force on atom i resulting from the potential model used. $Q_0 = N_f k_B T_0 \tau_T^2$ is the “mass” of the thermostat variable directly coupled to the system and π_0 is the corresponding “momentum”. N_f is the degree of freedom in the thermostatted region. k_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

$$\begin{aligned} \frac{d\mathbf{r}_i}{dt} &= \frac{\mathbf{p}_i}{m_i} \\ \frac{d\mathbf{p}_i}{dt} &= \mathbf{F}_i - \frac{\mathbf{p}_i}{\tau_T} + \mathbf{f}_i, \end{aligned}$$

Here, $\mathbf{r}_i$ is the position of atom i . $\mathbf{p}_i$ is the momentum of atom i . m_i is the mass of atom i . $\mathbf{F}_i$ is the total force on atom i resulted from the potential model used. $\mathbf{f}_i$ 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:

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

where

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

Here, $\mathbf{v}_i$ 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

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

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]:

$$\mathbf{J} = \mathbf{J}^{\text{pot}} + \mathbf{J}^{\text{kin}} = \sum_i \mathbf{J}_i^{\text{pot}} + \sum_i \mathbf{J}_i^{\text{kin}};$$

where

$$\mathbf{J}_i^{\text{kin}} = \mathbf{v}_i E_i;$$

and

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

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

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

and

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

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

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

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]:

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

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 thermostating method is recommended. Therefore, the *ensemble* keyword with the first parameter of `heat_1an` 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]

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

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

$$\mathbf{F}_i^{\text{ext}} = E_i \mathbf{F}_e + \mathbf{F}_e \cdot \mathbf{W}_i$$

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. $\mathbf{r}_{ij} \equiv \mathbf{r}_j - \mathbf{r}_i$, and $\mathbf{r}_i$ is the position of particle i .

The parameter $\mathbf{F}_e$ 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 $\langle \mathbf{J} \rangle_{\text{ne}}$ linearly related to $\mathbf{F}_e$:

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

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]:

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

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

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

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

where

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

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

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

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

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

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:

$$\kappa(\omega) = \frac{2\tilde{K}(\omega)}{VT F_e}$$

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:

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

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

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

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

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

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

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_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 $J_{x,n}(t')$ 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]:

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

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 $\langle J_n^\mu(t) \rangle_{\text{ne}}$ and the driving force parameter is F_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 coefficients that can be used to derive the thermal conductivity.

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

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

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:

$$\langle \mathbf{A}(t) \rangle = \langle \mathbf{A}(0) \rangle + \left(\int_0^t dt' \frac{\langle \mathbf{A}(t') \otimes \mathbf{J}_q(0) \rangle}{k_B T} \right) \cdot \mathbf{F}_e$$

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

$$\mathbf{F}_i^{\alpha, \text{ext}} = c_\alpha \mathbf{F}_e$$

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,

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

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

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

The onsager matrix is arranged as:

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

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.

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

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

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:

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

The bond-order is

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

where

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

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

4.1.2 File format

We have adopted a file format that similar but not identical to that used by [lammmps](#).

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 $i, j, k, \dots$ for atom indices and $I, J, K, \dots$ for atom types.

The site potential can be written as

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

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:

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

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

$$\begin{aligned} f_R(r) &= A_{IJ} e^{-\lambda_{IJ} r_{ij}} \\ f_A(r) &= B_{IJ} e^{-\mu_{IJ} r_{ij}}. \end{aligned}$$

The bond-order function is

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

where

$$\begin{aligned} \zeta_{ij} &= \sum_{k \neq i, j} f_C(r_{ik}) g_{ijk} \\ g_{ijk} &= 1 + \frac{c_I^2}{d_I^2} - \frac{c_I^2}{d_I^2 + (h_I - \cos \theta_{ijk})^2}. \end{aligned}$$

Parameter	Unit
A_{IJ}	eV
B_{IJ}	eV
λ_{IJ}	$\text{\AA}^{-1}$
μ_{IJ}	$\text{\AA}^{-1}$
β_I	dimensionless
n_I	dimensionless
c_I	dimensionless
d_I	dimensionless
h_I	dimensionless
R_{IJ}	$\text{\AA}$
S_{IJ}	$\text{\AA}$
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

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

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:

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

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

$$f_R(r_{ij}) = \frac{D_0}{S-1} \exp\left(\alpha r_0 \sqrt{2S}\right) e^{-\alpha \sqrt{2S} r_{ij}}$$

$$f_A(r_{ij}) = \frac{D_0 S}{S-1} \exp\left(\alpha r_0 \sqrt{2/S}\right) e^{-\alpha \sqrt{2/S} r_{ij}}.$$

The bond-order function is

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

where

$$\zeta_{ij} = \sum_{k \neq i, j} f_C(r_{ik}) g_{ijk}$$

$$g_{ijk} = \beta (h - \cos \theta_{ijk})^2.$$

4.3.2 Parameters

Parameter	Unit
D_0	eV
α	$\text{\AA}^{-1}$
r_0	$\text{\AA}$
S	dimensionless
n	dimensionless
β	dimensionless
h	dimensionless
R	$\text{\AA}$
S	$\text{\AA}$

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 supports 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:

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

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

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

The function for the density is

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

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.

4.5.4 Maximum neighbour count (eam/alloy)

For the eam/alloy form, the *potential* keyword accepts an optional second argument that sets the per-atom upper bound on neighbours used to allocate the neighbour list on the GPU:

```
potential <potential_file>           # default: max_neighbor = 400
potential <potential_file> 150       # smaller value to save GPU memory
```

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

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

with

$$\begin{aligned} U_2 &= \frac{1}{2!} \sum_{ij} \sum_{ab} \Phi_{ij}^{ab} u_i^a u_j^b \\ U_3 &= \frac{1}{3!} \sum_{ijk} \sum_{abc} \Phi_{ijk}^{abc} u_i^a u_j^b u_k^c \\ U_4 &= \frac{1}{4!} \sum_{ijkl} \sum_{abcd} \Phi_{ijkl}^{abcd} u_i^a u_j^b u_k^c u_l^d \\ U_5 &= \frac{1}{5!} \sum_{ijklm} \sum_{abcde} \Phi_{ijklm}^{abcde} u_i^a u_j^b u_k^c u_l^d u_m^e \\ U_6 &= \frac{1}{6!} \sum_{ijklmn} \sum_{abcdef} \Phi_{ijklmn}^{abcdef} u_i^a u_j^b u_k^c u_l^d u_m^e u_n^f. \end{aligned}$$

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:

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

- `fcf` 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 NEP4, as NEP1, NEP2, and NEP3 are deprecated.

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_{des} components,

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

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

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

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

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

where the summation runs over all the neighbors of atom i within a certain cutoff distance. There are thus $n_{\text{max}}^{\text{R}} + 1$ 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 ($0 \leq n \leq n_{\text{max}}^{\text{A}}$, $1 \leq l \leq l_{\text{max}}^{\text{3b}}$)

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

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,

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

with

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

and

$$f_c(r_{ij}) = \begin{cases} \frac{1}{2} \left[1 + \cos \left(\pi \frac{r_{ij}}{r_c^{\text{R}}} \right) \right], & r_{ij} \leq r_c^{\text{R}}; \\ 0, & r_{ij} > r_c^{\text{R}}. \end{cases}$$

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

4.7.3 Model dimensions

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

The total number of trainable parameters is the sum of the number of trainable descriptor parameters and the number of *NN* parameters N_{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.

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

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. $U^{\text{NEP}}(n, \mathbf{z})$ and $W_{\mu\nu}^{\text{NEP}}(n, \mathbf{z})$ are the per-atom energy and virial tensor predicted by the *NEP* model with parameters $\mathbf{z}$ for the n^{th} structure, and $\mathbf{F}_i^{\text{NEP}}(\mathbf{z})$ is the predicted force for the i^{th} atom. $U^{\text{tar}}(n)$, $W_{\mu\nu}^{\text{tar}}(n)$, and $\mathbf{F}_i^{\text{tar}}$ 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 [Bu2026] 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.

4.8.1 Potential form

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

$$U_i^{\text{ILP}} = \text{Tap}(r_{ij}) [U^{\text{att}}(r_{ij}) + U^{\text{rep}}(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)]$$

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

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

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

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

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

$$\begin{aligned}\rho_{ij}^2 &= r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_i)^2 \\ \rho_{ji}^2 &= r_{ij}^2 - (\mathbf{r}_{ij} \cdot \mathbf{n}_j)^2\end{aligned}$$

Other variables are all fitting parameters.

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

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

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 $16\text{\AA}$.

`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 you 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 you 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_NB alpha_NB delta_NB epsilon_NB C_NB d_NB sR_NB reff_NB C6_NB S_NB rcut1_NB rcut2_NB
beta_NN alpha_NN delta_NN epsilon_NN C_NN d_NN sR_NN reff_NN C6_NN S_NN rcut1_NN rcut2_NN
beta_NMo alpha_NMo delta_NMo epsilon_NMo C_NMo d_NMo sR_NMo reff_NMo C6_NMo S_NMo rcut1_
↪NMo rcut2_NMo
beta_NS alpha_NS delta_NS epsilon_NS C_NS d_NS sR_NS reff_NS C6_NS S_NS rcut1_NS rcut2_NS
...
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 [Jiang2025] in **GPUMD** combines the Stillinger-Weber potential (*SW*) [Stillinger1985] for intralayer 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:

$$U_i^{\text{ILP}} = \text{Tap}(r_{ij}) [U^{\text{att}}(r_{ij}) + U^{\text{rep}}(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)]$$

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

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

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

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

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

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

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

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

Other variables are all fitting parameters.

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

$$U_i^{\text{SW}} = \sum_i \sum_{j>i} \phi_2(r_{ij}) + \sum_i \sum_{j \neq i} \sum_{k>j} \phi_3(r_{ij}, r_{ik}, \theta_{ijk})$$

$$\phi_2(r_{ij}) = A_{ij} \epsilon_{ij} \left[B_{ij} \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{p_{ij}} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{q_{ij}} \right] \exp \left(\frac{\sigma_{ij}}{r_{ij} - a_{ij} \sigma_{ij}} \right)$$

$$\phi_3(r_{ij}, r_{ik}, \theta_{ijk}) = \lambda_{ijk} \epsilon_{ijk} [f_C(\delta) \delta]^2 \exp \left(\frac{\gamma_{ij} \sigma_{ij}}{r_{ij} - a_{ij} \sigma_{ij}} \right) \exp \left(\frac{\gamma_{ik} \sigma_{ik}}{r_{ik} - a_{ik} \sigma_{ik}} \right)$$

$$\delta = \cos \theta_{ijk} - \cos \theta_{0ijk}$$

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 intralayer 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:

$$U_i^{\text{ILP}} = \text{Tap}(r_{ij}) [U^{\text{att}}(r_{ij}) + U^{\text{rep}}(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)]$$

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

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

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

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

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

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

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

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

Other variables are all fitting parameters.

The site potential of Tersoff can be written as:

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

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

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

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 $\rho = 0, \Delta\rho, 2\Delta\rho, \dots, (N_\rho - 1)\Delta\rho$
 - Units: eV
- Next Nr values: Electron density function $\rho(r)$
 - Tabulated values at $r = 0, \Delta r, 2\Delta r, \dots, (N_r - 1)\Delta r$
 - 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·Angstrom²
 - 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

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 propagated from one run command to the next (*Prop.*).

Simulation setup

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

Actions

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

Output

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

$$\mathbf{a} = a_x \mathbf{e}_x + a_y \mathbf{e}_y + a_z \mathbf{e}_z$$

$$\mathbf{b} = b_x \mathbf{e}_x + b_y \mathbf{e}_y + b_z \mathbf{e}_z$$

$$\mathbf{c} = c_x \mathbf{e}_x + c_y \mathbf{e}_y + c_z \mathbf{e}_z$$

- `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 Ångström. 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 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:

```
# high symmetry points
k_points_x(0) k_points_y(0) k_points_z(0) name(0)
k_points_x(1) k_points_y(1) k_points_z(1) name(1)
...
```

Example

For example, the command:

```
0 0 0 G
0.5 0 0 M
0.333 0.333 0 K
0 0 0 G
```

or

```
0 0 0 G
0.5 0 0.5 X
0.625 0.25 0.625 U

0.375 0.375 0.75 K
0 0 0 G
0.5 0.5 0.5 L
0.5 0.25 0.75 W
0.5 0 0.5 X
```

Here,

- The first three columns represent the coordinates of high symmetry points. The last column corresponds to names.
- A blank line must be used to indicate a breakpoint or the start of a second path.

5.1.4 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.

Similar 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:

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

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

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

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

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

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

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

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

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 supports two forms.

4-parameter form

```
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.

In this form, the same deformation rate is applied to all directions that are flagged as deformed.

6-parameter form

```
deform <A_per_step_x> <A_per_step_y> <A_per_step_z> <deform_x> <deform_y> <deform_z>
```

Here, `A_per_step_x`, `A_per_step_y`, `A_per_step_z` specify the increment (or decrement) of the box length in the x , y , and z directions per simulation step, respectively. The unit is Ångstrom/step. For example, if the box length in the x -direction is initially 100 Å and `A_per_step_x` is set to 10^{-5} Å/step, then after 10^6 steps, the box length will change by 10%. With a time step of 1 fs, this corresponds to a strain rate of 10^8 s^{-1} . Use `deform_x`, `deform_y`, `deform_z` to enable or disable deformation in each direction. A value of 1 enables deformation, while 0 disables it. For a direction where deformation is disabled, the box length remains constant throughout the simulation.

This form allows independent deformation rates in the x , y , and z directions.

Example

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

Using the 4-parameter form:

```
# 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
```

Using the 6-parameter form:

```
# 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 0 0 1 0 0
run 1000000
```

Both examples achieve the same deformation (x-direction stretched at 0.00001 Å/step).

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.
- The 6-parameter form is more flexible and is recommended for new input files.

5.2.9 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.

5.2.10 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 `nvt_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 $\langle p_{\text{hydro}} \rangle = (p_{\text{xx}} + p_{\text{yy}} + p_{\text{zz}})/3$. 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:

$$\begin{aligned} C_{xx}=C_{xxxx}=C_{11}, C_{yy}=C_{yyyy}=C_{22}, C_{zz}=C_{zzzz}=C_{33}, \\ C_{yz}=C_{zyyz}=C_{44}, C_{zx}=C_{zxzx}=C_{55}, C_{xy}=C_{xyxy}=C_{66} \end{aligned}$$

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 canonical (NVT), 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 `nvt_mttk`, `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` updates 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 analogous fashion to run simulations in the isenthalpic (NPH) ensemble:

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


The `nvt_mttk` keyword can be used to run simulations in the canonical (NVT) ensemble:

```
ensemble nvt_mttk temp <T_1> <T_2> tperiod <tau_temp>
```

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 simulation 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.

NVT Ensemble

```
ensemble nvt_mttk temp 300 300
```

This command sets the target temperature to 300 K using the Nosé-Hoover chain thermostat. No barostat is used, so the simulation cell is fixed.

```
ensemble nvt_mttk temp 300 1000 tperiod 100
```

This command ramps the temperature from 300 K to 1000 K and explicitly sets the period of the thermostat to 100 timesteps.

5.2.13 ensemble (QTB)

The variants of the `ensemble` keyword described on this page implement the Quantum Thermal Bath (QTB) method [Dammak2009]. The QTB thermostat is a Langevin-type thermostat that uses colored noise with a quantum Bose-Einstein energy spectrum instead of classical white noise. This allows approximate inclusion of nuclear quantum effects (zero-point energy and quantum heat capacity) in otherwise classical MD simulations.

The `npt_qtb` variant combines the QTB thermostat with the Parrinello-Rahman (MTTK) barostat [Martyna1994], equivalent to LAMMPS `fix nph + fix qtb`.

Syntax

nvt_qtb

Run an NVT simulation with the QTB thermostat:

```
ensemble nvt_qtb <T_1> <T_2> <T_coup> [f_max <value>] [N_f <value>]
```

- `<T_1>` and `<T_2>`: Initial and final target temperature (K). The target temperature varies linearly during the run.
- `<T_coup>`: Thermostat coupling parameter (in units of timestep). Controls the friction coefficient: $\gamma = 1/(T_{\text{coup}} \times dt)$.
- `f_max`: (Optional, default 200) Maximum frequency of the QTB filter in ps^{-1} . Should be larger than the highest phonon frequency in the system.
- `N_f`: (Optional, default 100) Number of frequency points in the filter. The filter uses $2N_f$ points total.

npt_qtb

Run an NPT simulation with the QTB thermostat and Parrinello-Rahman (MTTK) barostat:

```
ensemble npt_qtb <direction> <p_1> <p_2> temp <T_1> <T_2> tperiod <tau_T> pperiod <tau_p>  
↪ [f_max <value>] [N_f <value>]
```

Pressure control parameters:

- `<direction>`: One or more of `iso`, `aniso`, `tri`, `x`, `y`, `z`. Same syntax as `npt_mttk`.
- `<p_1>` and `<p_2>`: Initial and final target pressure (GPa).

Temperature and coupling parameters:

- **temp**: Initial and final target temperature (K).
- **tperiod**: QTB thermostat coupling period (in units of timestep). Controls friction: $\gamma = 1/(\text{tperiod} \times dt)$.
- **ppperiod**: Barostat coupling period (in units of timestep, must be ≥ 200).

QTB-specific optional parameters (same as **nvt_qtb**):

- **f_max**: Maximum frequency (ps^{-1} , default 200).
- **N_f**: Number of frequency points (default 100).

Examples

NVT-QTB

```
ensemble nvt_qtb 300 300 100
```

Run at 300 K with QTB thermostat. The coupling parameter is 100 timesteps.

```
ensemble nvt_qtb 300 300 100 f_max 150 N_f 200
```

Same as above but with custom filter parameters.

NPT-QTB

```
ensemble npt_qtb iso 0 0 temp 300 300 tperiod 100 pperiod 1000
```

Run at 300 K and 0 GPa with isotropic pressure control.

```
ensemble npt_qtb aniso 0 0 temp 300 300 tperiod 100 pperiod 1000 f_max 200 N_f 100
```

Anisotropic pressure control with explicit QTB parameters.

```
ensemble npt_qtb x 5 5 y 0 0 z 0 0 temp 300 300 tperiod 100 pperiod 1000
```

Apply 5 GPa along x and 0 GPa along y and z.

Notes

- The QTB method generates colored noise whose power spectrum matches the quantum energy distribution $E(\omega) = \hbar\omega[\frac{1}{2} + n_{BE}(\omega, T)]$, where n_{BE} is the Bose-Einstein distribution.
- The kinetic temperature reported in **thermo.out** will be higher than the target temperature due to zero-point energy contributions. This is expected behavior.
- For liquid water at 300 K, the kinetic temperature is typically around 1000-1100 K.
- The **npt_qtb** ensemble uses the MTTK (Martyna-Tuckerman-Tobias-Klein) integrator for pressure control, which is the same as [npt_mtk](#) but with the Nosé-Hoover chain thermostat replaced by the QTB thermostat.
- The **f_max** parameter should be set larger than the highest phonon frequency in the system. For water, 200 ps^{-1} is sufficient.

References

5.2.14 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*.

`ttm`

If the first parameter is `ttm`, the run uses a two-temperature-model (TTM) integrator. The selected atom group is coupled to an electron temperature grid.

The full command is:

```
ensemble ttm <grouping_method> <group_id> <Ce> <rho_e> <kappa_e> <gamma_p> <gamma_s> <v_
↪0> <nx> <ny> <nz> <T_e_init> [{optional_args}]
```

The meanings of the parameters and optional arguments are described in *TTM integrators*.

heat_ttm

If the first parameter is `heat_ttm`, the run uses local source/sink Langevin thermostats together with the TTM electron grid.

The full command is:

```
ensemble heat_ttm <T> <T_coup> <delta_T> <label_source> <label_sink> <grouping_method>
↪ <group_id> <Ce> <rho_e> <kappa_e> <gamma_p> <gamma_s> <v_0> <nx> <ny> <nz> <T_e_init> [
↪ {optional_args}]
```

The first five parameters, `<T>`, `<T_coup>`, `<delta_T>`, `<label_source>`, and `<label_sink>`, have the same meanings as in `heat_lan`. The remaining parameters and optional arguments are described in [TTM integrators](#).

5.2.15 ensemble (TTM)

This page describes the two-temperature-model (TTM) integrators `ttm` and `heat_ttm`.

Syntax

ttm

The full command is:

```
ensemble ttm <grouping_method> <group_id> <Ce> <rho_e> <kappa_e> <gamma_p> <gamma_s> <v_
↪ 0> <nx> <ny> <nz> <T_e_init> [{optional_args}]
```

heat_ttm

The full command is:

```
ensemble heat_ttm <T> <T_coup> <delta_T> <label_source> <label_sink> <grouping_method>
↪ <group_id> <Ce> <rho_e> <kappa_e> <gamma_p> <gamma_s> <v_0> <nx> <ny> <nz> <T_e_init> [
↪ {optional_args}]
```

For `heat_ttm`, the first five parameters, `<T>`, `<T_coup>`, `<delta_T>`, `<label_source>`, and `<label_sink>`, have the same meanings as in `heat_lan`.

Required parameters

For both `ttm` and `heat_ttm`:

- `<grouping_method>` and `<group_id>` specify the atom group coupled to the electron grid.
- `<Ce>` is the electron specific heat per electron in eV/K.
- `<rho_e>` is the electron number density in $\text{\AA}^{-3}$.
- `<kappa_e>` is the electron thermal conductivity in eV/(ps K $\text{\AA}$).
- `<gamma_p>` and `<gamma_s>` are friction coefficients in amu/ps.
- `<v_0>` is the threshold velocity in $\text{\AA}/\text{ps}$.

- `<nx>`, `<ny>`, and `<nz>` are the numbers of electron grid cells in the three directions.
- `<T_e_init>` is the initial electron temperature in K.

The product `<Ce> times` is the volumetric electron heat capacity in $\text{eV}/(\text{K} \text{ \AA}^3)$.

Optional arguments

`ttm_out_interval`

Syntax:

```
ttm_out_interval <interval>
```

Set the output interval for *ttm_electron_temperature.out*. The default value is 1.

`ttm_infile`

Syntax:

```
ttm_infile <filename>
```

Read the initial electron temperature from a file. The file must contain one line for each cell in the form:

```
ix iy iz T_e
```

The grid indices are 1-based.

`ttm_properties_file`

Syntax:

```
ttm_properties_file <filename>
```

Read per-cell electron properties from a file. The file must contain one line for each cell in the form:

```
ix iy iz C_vol kappa_e gamma_p eta
```

Here, `C_vol` is the volumetric electron heat capacity in $\text{eV}/(\text{K} \text{ \AA}^3)$, `kappa_e` is in $\text{eV}/(\text{ps K} \text{ \AA})$, `gamma_p` is in amu/ps , and `eta` is the dimensionless source absorption efficiency. This file must define all grid cells and overrides the uniform `<Ce> times`, `<kappa_e>`, and `<gamma_p>` values.

`ttm_source`

Syntax:

```
ttm_source <source>
```

Add a volumetric heat source to the electron grid. The source strength is in $\text{eV}/(\text{ps} \text{ \AA}^3)$. When `ttm_properties_file` is used, the source in each cell is multiplied by `eta`.

`ttm_active_x`, `ttm_active_y`, `ttm_active_z`

Syntax:

```
ttm_active_x <range>
ttm_active_y <range>
ttm_active_z <range>
```

Set the active range of the electron grid in each direction. The range can be all, a single 1-based cell index, or an inclusive interval such as 3:10 or 3-10. Cells outside the active region have zero electron temperature and do not exchange energy with neighboring cells or atoms.

Notes

- The simulation box for the electron grid is the full MD box.
- The electron grid is always uniform in real space.
- Electron-temperature snapshots are written to *ttm_electron_temperature.out*.
- In `heat_ttm`, the source and sink labels always refer to grouping method 0.

Examples

Uniform pure TTM:

```
ensemble ttm 0 0 1.0 1.0 0.005 0.01 0.0 100.0 1 1 12 300
```

Pure TTM with an input electron-temperature profile and periodic snapshots:

```
ensemble ttm 0 0 1.0 1.0 0.005 0.01 0.0 100.0 1 1 12 300 ttm_infile te_init.dat ttm_out_
↪interval 50
```

Pure TTM with per-cell properties:

```
ensemble ttm 0 0 1.0 1.0 0.005 0.01 0.0 100.0 1 1 40 300 ttm_properties_file properties.
↪dat ttm_active_z 6:35 ttm_out_interval 100
```

Heat source/sink plus TTM:

```
ensemble heat_ttm 300 100 30 0 1 1 0 1.0 1.0 0.005 0.02 0.0 100.0 1 1 40 300 ttm_out_
↪interval 100
```

5.2.16 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 thermostat is applied to the internal modes.

5.2.17 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 simulation 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.

5.2.18 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 simulation 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 0 5
```

This command switch lambda for 4000 timesteps, and equilibrate for 1000 timesteps. The spring constant is 6 eV/Å² for Si and 5 eV/Å² 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.

5.2.19 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)")
```

5.2.20 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 simulation. 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, dlambd, 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")
```

(continues on next page)

(continued from previous page)

```

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

```

5.2.21 ensemble (TI)

This keyword is used to set up a equilibrium thermodynamic integration integrator. It is for testing purpose and its only difference 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 0 5
```

This command uses lambda value 0.3 (30% spring force and 70% original force field). The spring constant is 6 eV/Å² for Si and 5 eV/Å² for O.

5.2.22 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/Å². This keyword is optional with the default value of 10.

5.2.23 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 $\text{amu}^2/4$.
- `<mu_value>`: The artificial viscosity. It improves convergence. Its unit is $\sqrt{\text{amu} \times \text{eV}}/2$.
- `<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.

5.2.24 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 similar 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` updates 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.

5.2.25 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 $\text{eV}/\text{\AA}$.

Example 1

Add a constant force of $0.1 \text{ eV}/\text{\AA}$ 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.

5.2.26 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.

For qNEP models, the force equals to the dot product of the electric field and the Born effective charge (*BEC*). This is only meaningful when the qNEP model was trained with target *BEC*.

For other models, 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 ways:

```
add_efield <group_method> <group_id> <Ex> <Ey> <Ez> # usage 1
add_efield <group_method> <group_id> <add_efield_file> # usage 2
add_efield <group_method> <group_id> <Ex> <Ey> <Ez> <mode> # usage 3
add_efield <group_method> <group_id> <add_efield_file> <mode> # usage 4
```

- Electric field is applied to atoms in group `group_id` of group method `group_method`.
- In usage 1, the constant electric field with components `Ex`, `Ey`, and `Ez` is applied to each selected atom.
- In usage 2, a series of electric fields specified in the file `add_efield_file` will be periodically applied to each selected atom.
- In usages 1 and 2, if the potential model is qNEP, the added electric force equals to the dot product of the electric field and the *BEC*; otherwise it equals to the product of the electric field and the charge of the atom as specified in `model.xyz` via `charge:R:1`.
- In usages 3 and 4, `mode` can be `charge` or `bec`. * When `mode` is `charge`, the electric force will be calculated via
 - the qNEP predicted charges for qNEP potential models
 - the user-specified charges for other potential models
 - When `mode` is `bec`, the potential model must be qNEP, and the *BEC* will be used to calculate the electric force.
- Electric field is in units of $\text{V}/\text{\AA}$.

Example 1

Add a constant electric field of $0.1 \text{ V}/\text{\AA}$ 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.

5.2.27 add_spring

This keyword adds spring interactions to selected atom groups at each step of a run.

It supports three modes: `ghost_com`, `ghost_atom`, and `com_com`, which add spring forces between a ghost atom and the center of mass (COM) of a group, between ghost atoms and their corresponding atoms in a group, and between the COMs of two groups, respectively. The ghost atom(s) can be static or move at a constant velocity. Each mode supports two spring types: `couple` (a single radial spring) and `decouple` (three independent springs along the x, y, and z directions).

Syntax

The complete syntax for the six combinations is:

```
add_spring ghost_com <group_method> <group_id> <ghost_vx> <ghost_vy> <ghost_vz> couple
↪<k_couple> <R0> <offset_x> <offset_y> <offset_z>
add_spring ghost_com <group_method> <group_id> <ghost_vx> <ghost_vy> <ghost_vz> decouple
↪<k_decouple_x> <k_decouple_y> <k_decouple_z> <offset_x> <offset_y> <offset_z>
add_spring ghost_atom <group_method> <group_id> <ghost_vx> <ghost_vy> <ghost_vz> couple
↪<k_couple> <R0> <offset_x> <offset_y> <offset_z>
add_spring ghost_atom <group_method> <group_id> <ghost_vx> <ghost_vy> <ghost_vz>
↪decouple <k_decouple_x> <k_decouple_y> <k_decouple_z> <offset_x> <offset_y> <offset_z>
add_spring com_com <group_method> <group_id_1> <group_id_2> couple <k_couple> <R0>
add_spring com_com <group_method> <group_id_1> <group_id_2> decouple <k_decouple_x> <k_
↪decouple_y> <k_decouple_z>
```

The coupled spring potential is:

$$U_{\text{couple}} = \frac{1}{2} k_{\text{couple}} (R - R_0)^2$$

where R is the distance between the two points connected by the spring.

The decoupled spring potential is:

$$U_{\text{decouple}} = \frac{1}{2} k_{\text{decouple},x} (x_2 - x_1)^2 + \frac{1}{2} k_{\text{decouple},y} (y_2 - y_1)^2 + \frac{1}{2} k_{\text{decouple},z} (z_2 - z_1)^2$$

where (x_1, y_1, z_1) and (x_2, y_2, z_2) are the Cartesian coordinates of the two points connected by the spring.

- In `ghost_com` mode, force is added to atoms in group `group_id` of group method `group_method` with mass-weighted distribution. The ghost is initially placed at the COM + (`offset_x`, `offset_y`, `offset_z`) and is displaced by (`ghost_vx`, `ghost_vy`, `ghost_vz`) at each step.
- In `ghost_atom` mode, each atom in group `group_id` is attached to its own ghost anchor. The anchor is initially placed at the atom position + (`offset_x`, `offset_y`, `offset_z`) and is displaced by (`ghost_vx`, `ghost_vy`, `ghost_vz`) at each step. The input spring constant(s) (`k_couple` or `k_decouple_x`, `k_decouple_y`, `k_decouple_z`) represent the total stiffness of the entire group and are divided internally by the number of atoms in the group. Thus, each atom experiences a spring with an effective stiffness of k / N_{atoms} .
- In `com_com` mode, a spring interaction is applied between the COM of group `group_id_1` and the COM of group `group_id_2` under the same `group_method`. Equal and opposite forces are applied to the two groups with mass-weighted distribution within each group.
- In `couple` mode, `k_couple` must be positive and `R0` must be non-negative.
- In `decouple` mode, `k_decouple_x`, `k_decouple_y`, and `k_decouple_z` must be non-negative.
- In `com_com` mode, `group_id_1` and `group_id_2` must be different.

- Force is in units of eV/Å, distance is in units of Å, velocity is in units of Å/step, and spring constant is in units of eV/Å².
- Spring forces are written to the file `spring_force_*.out`, where the asterisk is replaced by a zero-based index that increments with each invocation of the command. The meaning of each column in the file is:

```
# step mode Fx Fy Fz Ftotal energy
```

where `mode` is an integer flag: 0 = ghost_com, 1 = ghost_atom, 2 = com_com. In `com_com` mode, the reported `Fx`, `Fy`, and `Fz` correspond to the net spring force applied to `group_id_1`.

Example 1 (ghost_com + couple)

Add a coupled spring with spring constant 10 eV/Å² and equilibrium distance 0 Å between a static ghost atom and the COM of atoms in group 2 defined by group method 0. The ghost atom is initially located at the COM:

```
add_spring ghost_com 0 2 0 0 0 couple 10 0 0 0 0
```

Example 2 (ghost_com + decouple)

Add a decoupled spring with spring constants 10 eV/Å² in the x direction and 0 eV/Å² in the y and z directions between a ghost atom moving at velocity (0.00005, 0, 0) Å/step and the COM of atoms in group 2 defined by group method 0. The ghost atom is initially located at the COM:

```
add_spring ghost_com 0 2 0.00005 0 0 decouple 10 0 0 0 0
```

Example 3 (ghost_atom + couple)

Add a coupled spring between each atom in group 2 defined by group method 0 and its corresponding moving ghost anchor. Each anchor is initially placed at the corresponding atom position and moves at velocity (0.00005, 0, 0) Å/step:

```
add_spring ghost_atom 0 2 0.00005 0 0 couple 10 0 0 0 0
```

Example 4 (com_com + decouple)

Add a decoupled Cartesian spring between the COM of group 1 and the COM of group 2 under the same group method 0:

```
add_spring com_com 0 1 2 decouple 10 0 0
```

Note

This keyword can be used multiple times during a run.

5.2.28 deposit

This keyword is used to simulate a deposition process, where new atoms are periodically added to the system during a run.

Syntax

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

```
deposit <interval> <direction> <height_min> [height_max] atom <type_1> <num_1> <velocity_
↪1> [<type_2> <num_2> <velocity_2> ...] # usage 1
deposit <interval> <direction> <height_min> [height_max] file <add_atom_file> [velocity]_
↪                                     # usage 2
```

- **interval** is the deposition interval (number of steps) and must be a positive integer. The number of steps in the *run keyword* must be divisible by it.
- **direction** is the deposition direction: 0, 1, and 2 correspond to the x, y, and z directions, respectively. New atoms are placed at the given height along this direction and are given an initial velocity along it. The special value -1 is only allowed in the second usage and means that the atoms are added at the exact positions and with the exact velocities given in the file.
- **height_min** and the optional **height_max** define the deposition height (in units of Ångstrom) along the deposition direction. If **height_max** is given, the height of each deposited atom is uniformly sampled between **height_min** and **height_max**; otherwise, all the atoms are deposited at **height_min**. When **direction** is -1, the height value(s) are not used.
- In the first usage, one or more triplets <type> are given. For each triplet, num atoms of type type are added at each deposition, at random lateral positions in the simulation box, with an initial velocity velocity along the deposition direction. The atom types refer to those defined in the potential file, and the velocity is in units of Å/fs.
- In the second usage, the atoms to be added at each deposition are read from the file **add_atom_file**, which contains one atom per row with 7 columns:

```
type x y z vx vy vz
```

The positions are in units of Ångstrom and the velocities are in units of Å/fs. With a non-negative **direction**, the coordinate along the deposition direction is replaced by the sampled height, and the optional **velocity** (in units of Å/fs) after the file name sets the velocity component along the deposition direction. With **direction** set to -1, no **velocity** is allowed after the file name and the file is used as it is.

At each deposition, the new atoms are appended to the model and the *run* is effectively split into sub-runs of **interval** steps, with one deposition between two consecutive sub-runs. After each sub-run, the current structure is written to a file named **deposited_N.xyz** in extended XYZ format, including the velocities (and the group labels if the model has grouping). If the *simulation model file* contains no velocity data, one extra sub-run of the pristine model is performed before the first deposition.

Example 1

Deposit 10 atoms of type 0 every 10000 steps, at a height of 30 Å in the z direction, with an initial velocity of -0.1 Å/fs (towards decreasing z):

```
deposit 10000 2 30 atom 0 10 -0.1
```

Example 2

Deposit 5 atoms of type 0 and 5 atoms of type 1 every 5000 steps, at heights uniformly sampled between 30 Å and 40 Å in the z direction:

```
deposit 5000 2 30 40 atom 0 5 -0.1 1 5 -0.1
```

Example 3

Deposit atoms read from the file `add_atoms.txt` every 10000 steps, resetting the z coordinate to 30 Å and the z velocity to -0.1 Å/fs:

```
deposit 10000 2 30 file add_atoms.txt -0.1
```

Example 4

Deposit atoms read from the file `add_atoms.txt` every 10000 steps, keeping the positions and velocities exactly as given in the file (direction is -1 and no velocity follows the file name):

```
deposit 10000 -1 0 file add_atoms.txt
```

Caveats

- This keyword can appear only once in a `run.in` file, and only one *run keyword* is allowed when it is used.
- The original `run.in` and `model.xyz` are saved as `run.in.original` and `model.xyz.original`, respectively, and the `run.in` and `model.xyz` files will be modified during the simulation.
- If the model has grouping, all the deposited atoms are assigned to new groups (with labels one larger than the largest existing group label for each group method), such that they can be distinguished from the original atoms.

5.2.29 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 $E_{\min}$; * the third is the maximum of the energy range $E_{\max}$.

The stopping power data listed after this line should have N lines, each corresponding to one energy, which increases linearly from $E_{\min}$ to $E_{\max}$ with a spacing of $(E_{\max} - E_{\min})/(N - 1)$. 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
```

5.2.30 fix

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

Syntax

This keyword accepts 1 or 2 parameters. The atoms in the specified group will be fixed (velocities and forces are always set to zero such that the atoms do not move). The full command reads:

```
fix <group_label>  
fix <grouping_method> <group_label>
```

- If only `group_label` is given, grouping method 0 is used by default.
- If `grouping_method` is also specified, the given grouping method defined in the *simulation model file* will be used.

Example

Fix group 0 using the default grouping method 0:

```
fix 0
```

Fix group 2 using grouping method 1:

```
fix 1 2
```

Caveats

- This keyword is not propagating, which means that it only affects the simulation within the run it belongs to.
- When both `fix` and `move` are used, they must use the same grouping method.

5.2.31 kspace

This keyword is used to set the computation method for the reciprocal space contribution to the electrostatic energy.

Syntax

This keyword is used as follows:

```
kspace <method>
```

where `<method>` can be either *ewald* or *pppm*. The default is *pppm*, which implies that the particle-particle particle-mesh (PPPM) method is used.

Example

To use the Ewald method use:

```
kspace ewald
```

5.2.32 move

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

Syntax

The move keyword accepts 4 or 5 parameters:

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

- If only `moving_group_id` and velocities are given, grouping method 0 is used by default.
- If `grouping_method` is also specified, the given grouping method defined in the *simulation model file* will be used.

The last 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_ber 300 300 100
fix 0 # fix atoms in group 0 (default grouping method 0)
move 1 0.001 0 0 # move atoms in group 1, with a speed of 0.001 Ångstrom/fs in the x_
↳ direction (default grouping method 0)
run 1000000
```

Using a specific grouping method:

```
ensemble nvt_ber 300 300 100
fix 1 0 # fix group 0 in grouping method 1
move 1 1 0.001 0 0 # move group 1 in grouping method 1, with a speed of 0.001 Ångstrom/
↳ fs in the x direction
run 1000000
```

Caveats

- One cannot use NPT when using this keyword. Only `nvt_ber`, `nvt_nhc`, `nvt_bdp`, and `heat_lan` ensembles are supported.
- When both `fix` and `move` are used, they must use the same grouping method.

5.2.33 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 =  $\phi_i$`). 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.

5.2.34 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.

5.2.35 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 (iterations) performed is smaller than `maximal_number_of_steps`. `maximal_number_of_steps` is the maximal number of steps (iterations) 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 assign 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 output 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*.

5.2.36 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 ...
```

5.2.37 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.

5.2.38 `compute_chunk`

This keyword computes space- and time-averaged per-chunk quantities based on dynamic spatial binning. Unlike the *compute keyword*, which uses static group assignments defined in the model file, `compute_chunk` assigns atoms to spatial bins (chunks) based on their real-time coordinates at each sampling step. This is particularly useful for systems where atoms diffuse across spatial regions during the simulation, such as low-density, porous, or amorphous materials in NEMD simulations.

The results are written to the `compute_chunk.out` output file.

Syntax

For 1D binning:

```
compute_chunk <sample_interval> <output_interval> bin/1d <dim> <origin> <delta> {
  ↪ <quantity>}
```

For 2D binning:

```
compute_chunk <sample_interval> <output_interval> bin/2d <dim1> <origin1> <delta1> <dim2>
  ↪ <origin2> <delta2> {<quantity>}
```

For 3D binning:

```
compute_chunk <sample_interval> <output_interval> bin/3d <dim1> <origin1> <delta1> <dim2>
  ↪ <origin2> <delta2> <dim3> <origin3> <delta3> {<quantity>}
```

The parameters are defined as follows:

- `sample_interval`: Sampling interval in time steps. The quantities are sampled every this many steps.

- `output_interval`: Number of samples to average before producing one output. This is consistent with the *compute keyword*: one output is produced every `sample_interval × output_interval` time steps.
- `bin/1d`, `bin/2d`, `bin/3d`: The binning style, specifying 1D, 2D, or 3D spatial binning respectively.
- `dim` (or `dim1`, `dim2`, `dim3`): The axis along which to bin. Must be `x`, `y`, or `z`. For `bin/2d` and `bin/3d`, the specified axes must be distinct.
- `origin` (or `origin1`, `origin2`, `origin3`): The bin origin. Currently only `lower` is supported, meaning bins start from the lower boundary of the simulation box (coordinate 0).
- `delta` (or `delta1`, `delta2`, `delta3`): The bin width in Ångströms. The number of bins along each axis is automatically calculated as $\lceil L/\delta \rceil$, where L is the box length along that axis. The last bin may be narrower than δ if L is not an integer multiple of δ ; its volume and center coordinate are computed using the actual width.

Starting after the binning parameters, one can list the quantities to be computed. The allowed names for `quantity` are:

- `temperature`, which yields the temperature of each chunk
- `density/number`, which yields the number density (atoms per volume) of each chunk
- `density/mass`, which yields the mass density of each chunk
- `vx`, which yields the average velocity in the `x` direction
- `vy`, which yields the average velocity in the `y` direction
- `vz`, which yields the average velocity in the `z` direction
- `fx`, which yields the average force in the `x` direction
- `fy`, which yields the average force in the `y` direction
- `fz`, which yields the average force in the `z` direction

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

For triclinic (non-orthogonal) simulation boxes, the box length along each axis is computed as the geometric thickness $L = V/A$, where V is the box volume and A is the cross-sectional area perpendicular to that axis. This is consistent with GPUMD's internal geometry calculations.

Example

Example 1: 1D temperature profile along `z`

```
compute_chunk 10 100 bin/1d z lower 1.0 temperature
```

This means:

- sample the temperature every 10 time steps
- average over 100 samples before writing output (one output every $10 \times 100 = 1000$ steps)
- bin atoms along the `z` axis with 1.0 Å bin width, starting from the lower boundary
- compute the temperature of each bin

Example 2: 1D temperature and density profile

```
compute_chunk 10 100 bin/1d z lower 1.0 temperature density/number
```

Same as above, but also computes the number density of each bin.

Example 3: 2D velocity field

```
compute_chunk 10 100 bin/2d x lower 2.0 z lower 2.0 vx vz
```

This creates a 2D grid of bins in the x-z plane with 2.0 Å bin width in each direction, and computes the average x and z velocity components in each bin.

Example 4: NEMD temperature profile

A typical use case for NEMD simulations:

```
ensemble heat_lan 300 300 50 0 1
compute_chunk 10 100 bin/1d z lower 1.0 temperature density/number
run 1000000
```

This computes the temperature and density profiles along z during a Langevin heat bath NEMD simulation.

Output file

The results are written to `compute_chunk.out`. The file is appended to if it already exists. For each output time, there are N_{chunk} consecutive lines (one per chunk), with the following format per line:

```
<chunk_id> <coord1> [<coord2>] [<coord3>] <avg_count> <value1> [<value2>] ...
```

The columns are:

- `chunk_id`: the zero-based index of the chunk (dimensionless integer).
- `coord1` (and `coord2`, `coord3` for 2D/3D binning): the center coordinates (in units of Å) of the chunk along the corresponding binning axis.
- `avg_count`: the time-averaged number of atoms in the chunk (dimensionless).
- `value1`, `value2`, ...: the time-averaged values of the requested quantities, in the same order as specified in the command. The units for each quantity are:
 - `temperature`: K
 - `density/number`: Å⁻³ (number of atoms per volume)
 - `density/mass`: amu/Å³ (total atomic mass per volume)
 - `vx`, `vy`, `vz`: Å/fs
 - `fx`, `fy`, `fz`: eV/Å

The temperature is computed from the kinetic energy of the atoms in each chunk as $T = 2E_k / (3k_B N)$, where E_k is the total kinetic energy, N is the atom count, and k_B is the Boltzmann constant. The velocity and force values are per-atom averages within each chunk.

For example, the command:

```
compute_chunk 10 100 bin/1d z lower 2.0 temperature density/number
```

produces output lines of the form:

```
0 1.000000 52.0 3.0012345678e+02 2.5000000000e-02
1 3.000000 48.0 2.9876543210e+02 2.4000000000e-02
...
```

Here, each line contains: chunk index, bin center along z (Å), average atom count, temperature (K), and number density (Å⁻³). The output blocks for successive time points are written consecutively with no blank line separator; each block contains exactly N_{chunk} lines.

Caveats

- The `origin` parameter currently only accepts `lower`. Numeric origin values are not supported.
- This keyword addresses the measurement/analysis side of dynamic spatial binning. The NEMD heat baths (`ensemble heat_*` source/sink regions) still use static group assignments. Dynamic heat bath regions would require separate modifications to the integrator.
- Multiple `compute_chunk` commands can be used in the same run.

5.2.39 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 $rc_{\text{min}} < R < rc_{\text{max}}$ 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> <ctype1> <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 `ctypeN`.

- 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

5.2.40 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> <direction>
```

Here, *e1* is the smaller box-scaling factor, *e2* is the larger box-scaling factor, and *direction* specifies the direction of the scaling, which can take values from 0 to 6, corresponding to the x, y, z, xy, yz, zx, and xyz directions, respectively.

Examples

The command:

```
compute_cohesive 0.9 1.2 0
```

means that one wants to compute the cohesive energy curve in the x-direction from the box-scaling factor 0.9 to the box-scaling factor 1.2, with $(e2 - e1) * 1000 + 1$ points. The box-scaling points will be 0.9, 0.901, 0.902, ..., 1.2.

Caveats

This keyword must occur after the *potential keyword*.

5.2.41 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.

5.2.42 compute_dpdt

This keyword can be used to calculate the time derivative of the polarization via the dot product between the Born effective charge (*BEC*) and the velocity. The keyword is only meaningful for a simulation with the qNEP model trained with target *BEC* values. The results will be written to the file *dpdt.out*.

Syntax

This keyword has 1 parameter:

```
compute_dpdt <sampling_interval>
```

Here,

- *sampling_interval* is the sampling interval for calculating the time derivative of the polarization.

Examples

Example 1

```
compute_dpdt 5
```

This means to calculate the time derivative of the polarization every 5 time steps.

5.2.43 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>
```

strain_value is the amount of strain to be applied in the calculations.

Example

For example, the command:

```
compute_elastic 0.01
```

means that one wants to compute the elastic constants with a strain of 0.01.

Caveats

This keyword must occur after the *potential keyword*.

5.2.44 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 *eigen-vector.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.

5.2.45 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 1000000 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 million 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 1000000 10
```

This is similar to the above example but with one expectation: The *HAC/RTC* data will be averaged for every 10 data before outputting, generating 10^4 rows of data in the output file.

5.2.46 compute_hnema

The `compute_hnema` keyword is used to calculate the modal thermal conductivity using the *homogeneous non-equilibrium 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 $\text{\AA}^{-1}$.

`Fe_y` is the y direction component of the external driving force F_e in units of $\text{\AA}^{-1}$.

`Fe_z` is the z direction component of the external driving force F_e in units of $\text{\AA}^{-1}$.

`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 0.0000008 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 $0.8 \times 10^{-5} \text{ \AA}^{-1}$
- 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 0.0000008 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 0.0000008 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 *eigen-vector.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.

5.2.47 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 $\mathbf{F}_e$ in units of $\text{\AA}^{-1}$.

Usually, there should be only one nonzero component of $\mathbf{F}_e$. According to Eq. (8) of [Fan2019]:

- Using a nonzero x component of $\mathbf{F}_e$, one can obtain the xx , yx and zx components of the thermal conductivity tensor.
- Using a nonzero y component of $\mathbf{F}_e$, one can obtain the xy , yy and zy components of the thermal conductivity tensor.
- Using a nonzero z component of $\mathbf{F}_e$, 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 $10^{-5} \text{\AA}^{-1}$.

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.

5.2.48 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 equivalent to dissipative flux, with driving force F_e in the units of $\text{\AA}^{-1}$. For a non zero positive integer such as i , a momentum flux of the i th 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 $\text{eV}/\text{\AA}$.

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 $10^{-5} \text{\AA}^{-1}$.

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 $10^{-5} \text{ eV}/\text{\AA}$. The force is applied along the y direction.

5.2.49 compute_ic

This keyword computes the iron conductivity (*IC*) from the mean-square displacement (*MSD*) function. If this keyword appears in a run, the *IC* function will be calculated as a time derivative of it. The results will be written to *ic.out output file*.

Syntax

For this keyword, the command looks like:

```
compute_ic <sample_interval> <Nc> <type_iex> <charge>
```

with parameters defined as

- `sample_interval`: Sampling interval of the position data
- `Nc`: Maximum number of correlation steps
- `type_iex`: Type index of atom to be calculated which in potential
- `charge`: Charge of the ion to compute the conductivity.

Examples

An example of this function is:

```
compute_ic 5 200 0 1
```

This means that you

- want to calculate the iron conductivity from the mean-square displacement function
- the position data will be recorded every 5 steps
- the maximum number of correlation steps is 200
- you would like to compute only for the atom with type 0 in `nep.txt` and charge 1.

5.2.50 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 :

$$q_{lm}(i) = \frac{1}{N_b} \sum_{j=1}^{N_b} Y_{lm}(\theta(\mathbf{r}_{ij}), \phi(\mathbf{r}_{ij})).$$

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 $\bar{q}_{lm}(i)$, which is the mean value of $q_{lm}(k)$ over all neighbors k of particle i , including atom i itself:

$$\bar{q}_{lm}(i) = \frac{1}{N_b} \sum_{k=0}^{N_b} q_{lm}(k).$$

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.

5.2.51 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 prepared in the input directory:

Input filename	Brief description
kpoints.in	Specify the k -points along the high-symmetry paths

To use this keyword, replicate keywords must write head in the `run.in` file.

Syntax

This keyword is used as follows:

```
compute_phonon <displacement>
```

`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 0.01
```

means that one wants to compute the phonon dispersion using 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](#).

5.2.52 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.

5.2.53 compute_msdc

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
```


5.2.54 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.

5.2.55 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.

5.2.56 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
```

5.2.57 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.

Mathematical details

The *RDF* is proportional to the relative probability of finding atomic pairs at a distance around r in the system, and to a large extent, reflects the local structural information of the system. It is defined as

$$g(r) = \frac{V}{N^2} \frac{dN(r, dr)}{4\pi r^2 dr},$$

where $dN(r, dr)$ represents the number of particle pairs with distances between r and $r + dr$, V is the volume of the system, and N is the total number of particles in the system.

It is also possible to calculate the *RDF* between different element types. In a system with N_a atoms of type a and N_b atoms of type b , $g_{ab}(r)$ and $g_{ba}(r)$ is defined as

$$g_{ab}(r) = g_{ba}(r) = \frac{V}{N_a N_b} \frac{dN_{ab}(r, dr)}{4\pi r^2 dr},$$

where $dN_{ab}(r, dr) = dN_{ba}(r, dr)$ is the number of (a, b) particle pairs with distances between r and $r + dr$.

Syntax

This keyword is used as follows:

```
compute_rdf <cutoff> <num_bins> <interval>
```

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.

Starting from GPUMD-v4.9, there is no need to specify the atom pairs and the code will calculate the partial *RDFs* for all atom pairs.

Example

```
compute_rdf 8.0 400 1000 # Calculate all RDFs every 1000 MD steps with 400 data up to 8 Angstrom
```

5.2.58 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
```

5.2.59 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.

5.2.60 dump_xyz

Write some data into *dump.xyz* in *extended XYZ format*.

Syntax

```
dump_xyz <interval> <has_velocity>
dump_xyz <interval> <has_velocity> <has_force>
dump_xyz <interval> <has_velocity> <has_force> <has_potential>
dump_xyz <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_xyz 1000          # dump positions every 1000 steps
dump_xyz 1000 1         # dump positions and velocities
dump_xyz 1000 1 1       # dump positions, velocities, and forces
dump_xyz 1000 1 1 1     # dump positions, velocities, forces, and potentials
dump_xyz 1000 0 1 1     # dump positions, forces and potentials
dump_xyz 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.

5.2.61 dump_xyz

Write per-atom data into user-specified file(s) in [extended XYZ](#) format.

Syntax

```
dump_xyz <grouping_method> <group_id> <interval> <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, charge, bec, 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.
- For qNEP models, the charge values dumped out are predicted by the qNEP models. For other models, the charge values are those specified in `model.xyz` via `charge:R:1`.
- The Born effective charge (*BEC*) is only meaningful for a qNEP model trained with target *BEC*.

5.2.62 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_xyz* 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_xyz* keyword.
- This keyword should not be used in a run not related to PIMD.

5.2.63 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_xyz* 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_xyz* time steps. Every *interval_thermo* timesteps files in the style of *thermo.out* will be written, and every *interval_xyz* 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_xyz* 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_xyz> <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_xyz* parameter is the output interval (number of steps) of writing xyz files. *has_velocity* can be 1 or 0, which means the velocities will or will not be included in the xyz output. *has_force* can be 1 or 0, which means the forces will or will not be included in the xyz 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 xyz 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.

5.2.64 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.

5.2.65 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.

5.2.66 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* should 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
```

5.2.67 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 optional 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 velocities

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).

5.2.68 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` should 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.

5.2.69 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.

5.2.70 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.

5.2.71 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` should 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
```

5.2.72 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 partial 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
<i>thermo.out</i>	<i>dump_thermo</i>	Global thermodynamic quantities
<i>movie.xyz</i>	<i>dump_position</i>	Trajectory (atomic positions, velocities etc)
<i>restart.xyz</i>	<i>dump_restart</i>	The restart file
<i>dump.xyz</i>	<i>dump_xyz</i>	Atomistic positions, velocities and forces
<i>observer.xyz</i>	<i>dump_observer</i>	Atomistic quantities as evaluated with observing potentials
<i>observer.out</i>	<i>dump_observer</i>	Thermodynamic quantities evaluated with observing potentials
<i>active.xyz</i>	<i>active</i>	Structures selected through active learning
<i>active.out</i>	<i>active</i>	Simulation time and uncertainty during active learning
<i>dipole.out</i>	<i>dump_dipole</i>	Predicted dipole
<i>polarizability.out</i>	<i>dump_polarizability</i>	Predicted polarizability
<i>velocity.out</i>	<i>dump_velocity</i>	Contains the atomic velocities
<i>force.out</i>	<i>dump_force</i>	Contains the atomic forces
<i>compute.out</i>	<i>compute</i>	Time and space (group) averaged quantities
<i>ttm_electron_temperature.out</i>	<i>ensemble</i> with <i>ttm</i> or <i>heat_ttm</i>	Electron temperature snapshots on the TTM grid
<i>hac.out</i>	<i>compute_hac</i>	Thermal conductivity data from the <i>EMD</i> method
<i>kappa.out</i>	<i>compute_hnemd</i>	Thermal conductivity data from the <i>HNEMD</i> method
<i>shc.out</i>	<i>compute_shc</i>	Spectral heat current (<i>SHC</i>) data
<i>heatmode.out</i>	<i>compute_gkma</i>	Modal heat current data from <i>GKMA</i> method
<i>kappamode.out</i>	<i>compute_hnema</i>	Modal thermal conductivity data from <i>HNEMA</i> method
<i>dos.out, mvac.out</i>	<i>compute_dos</i>	Phonon density of states (<i>PDOS</i>) data
<i>dpdt.out</i>	<i>compute_dpdt</i>	time derivative of the polarizability and its time integration
<i>sdsc.out</i>	<i>compute_sdc</i>	Self-diffusion coefficient (<i>SDC</i>) data
<i>msd.out</i>	<i>compute_msd</i>	Mean-square displacement (<i>MSD</i>) data
<i>ic.out</i>	<i>compute_ic</i>	Iron conductivity (<i>IC</i>) data
<i>cohesive.out</i>	<i>compute_cohesive</i>	Cohesive energy curve
<i>D.out</i>	<i>compute_phonon</i>	Dynamical matrices $D(\mathbf{k})$ for the input $\mathbf{k}$ points
<i>omega2.out</i>	<i>compute_phonon</i>	Phonon frequency squared $\omega^2(\mathbf{k})$ for the input $\mathbf{k}$ -points
<i>viscosity.out</i>	<i>compute_viscosity</i>	Viscosity and stress auto-correlation function
<i>onsager.out</i>	<i>compute_hnemdec</i>	Onsager coefficients
<i>rdf.out</i>	<i>compute_rdf</i>	Radial distribution function (<i>RDF</i>)
<i>adf.out</i>	<i>compute_adf</i>	Angular distribution function (<i>ADF</i>)
<i>angular_rdf.out</i>	<i>compute_angular_rdf</i>	Angular-dependent radial distribution function (<i>ARDF</i>)
<i>mcmd.out</i>	<i>mc</i>	Acceptance ratio and species concentrations
<i>lsqt_dos.out</i>	<i>compute_lsqt</i>	Electronic density of states
<i>lsqt_velocity.out</i>	<i>compute_lsqt</i>	Electron group velocity
<i>lsqt_sigma.out</i>	<i>compute_lsqt</i>	Electrical conductivity
<i>orientorder.out</i>	<i>compute_orientorder</i>	Steinhardt bond-orientational order parameters

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, 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/Å) 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.

For runs that do not use explicit source and sink thermostats, such as a pure `ttm` run, the last two columns are present but remain zero.

5.3.3 ttm_electron_temperature.out

This file contains electron temperature snapshots from `ttm` and `heat_ttm` runs. It is written every `ttm_out_interval` steps and the output mode is `overwrite`.

File format

The file starts with a header such as:

```
# electron temperature snapshots for TTM
# nx 1 ny 1 nz 12
# active_x 1 1 active_y 1 1 active_z 3 10
# properties_file yes
# electron_source 0.000000000000e+00
# output_interval 50 step(s)
# columns: ix iy iz T_e[K]
```

Each snapshot starts with:

```
# step 2000
```

followed by one line for each electron grid cell:

```
ix iy iz T_e
```

where:

- `ix`, `iy`, `iz` are the 1-based electron-grid indices
- `T_e` is the electron temperature in K

Notes

- The grid order is x-fastest, then y, then z.
- Cells outside the active TTM region are written with zero electron temperature.

5.3.4 D.out

This file contains the dynamical matrices $D(\mathbf{k})$ at different $\mathbf{k}$ -points.

File format

The file contains $3 * N_{\text{basis}} * N_{\text{kpoints}}$ rows and $6 * N_{\text{basis}}$ columns, where N_{basis} is the number of basis atoms in the unit cell defined and N_{kpoints} is the number of $\mathbf{k}$ -points from *kpoints.in input file* created.

Each consecutive $3 * N_{\text{basis}}$ rows correspond to one $\mathbf{k}$ -point. For each $\mathbf{k}$ -point, the first $3*N_{\text{basis}}$ columns correspond to the real part and the second $3*N_{\text{basis}}$ 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 $\text{eV } \text{\AA}^{-2} \text{ amu}^{-1}$.

5.3.5 ic.out

This file contains the iron conductivity (*IC*) based on mean-square displacement (*MSD*). 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: IC (in units of mS/cm) in the *x* direction
- column 3: IC (in units of mS/cm) in the *y* direction
- column 4: IC (in units of mS/cm) in the *z* direction

Only the element selected which in potential via the arguments of the *compute_msd* keyword is included in this output.

5.3.6 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).

5.3.7 dpdt.out

This file contains the time derivative of the polarization and its time integration. The integration assumes a starting value of zero for the polarization. It is produced when invoking *compute_dpdt* keyword in the *run.in* input file.

File format

The file is organized as follows:

- column 1: time (in units of fs)
- column 2: time derivative of the polarization in the x direction dP_x/dt (in units of e A / fs)
- column 3: time derivative of the polarization in the y direction dP_y/dt (in units of e A / fs)
- column 4: time derivative of the polarization in the z direction dP_z/dt (in units of e A / fs)
- column 5: polarization in the x direction P_x (in units of e A)
- column 6: polarization in the y direction P_y (in units of e A)
- column 7: polarization in the z direction P_z (in units of e A)

5.3.8 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.

5.3.9 hac.out

This file contains the heat current auto-correlation (*HAC*) function and the running thermal conductivity (*RTC*) from 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: $\langle J_x^{\text{in}}(0)J_x^{\text{tot}}(t) \rangle$ (in units of eV³/amu)
- column 3: $\langle J_x^{\text{out}}(0)J_x^{\text{tot}}(t) \rangle$ (in units of eV³/amu)
- column 4: $\langle J_y^{\text{in}}(0)J_y^{\text{tot}}(t) \rangle$ (in units of eV³/amu)
- column 5: $\langle J_y^{\text{out}}(0)J_y^{\text{tot}}(t) \rangle$ (in units of eV³/amu)
- column 6: $\langle J_z^{\text{tot}}(0)J_z^{\text{tot}}(t) \rangle$ (in units of eV³/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.

5.3.10 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:

$$\begin{array}{cccccc}
 J_{1,1}^{x,in} & J_{1,1}^{x,out} & J_{1,1}^{y,in} & J_{1,1}^{y,out} & J_{1,1}^z & \dots \\
 J_{1,2}^{x,in} & J_{1,2}^{x,out} & J_{1,2}^{y,in} & J_{1,2}^{y,out} & J_{1,2}^z & \dots \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \\
 J_{1,n}^{x,in} & J_{1,n}^{x,out} & J_{1,n}^{y,in} & J_{1,n}^{y,out} & J_{1,n}^z & \dots \\
 J_{2,1}^{x,in} & J_{2,1}^{x,out} & J_{2,1}^{y,in} & J_{2,1}^{y,out} & J_{2,1}^z & \dots \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
 \end{array}$$

- Each output is in units of $\text{eV}^{3/2}\text{amu}^{-1/2}$.
- 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.

5.3.11 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 contributions 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:

$$\begin{aligned}
 \kappa_{\mu x}^{\text{tot}}(t) &= \kappa_{\mu x}^{\text{in}}(t) + \kappa_{\mu x}^{\text{out}}(t) \\
 \kappa_{\mu y}^{\text{tot}}(t) &= \kappa_{\mu y}^{\text{in}}(t) + \kappa_{\mu y}^{\text{out}}(t).
 \end{aligned}$$

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.

5.3.12 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.

5.3.13 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 $\text{\AA}^2/\text{s}^2$) in the math:x direction
- column 3: VAC (in units of $\text{\AA}^2/\text{s}^2$) in the math:y direction
- column 4: VAC (in units of $\text{\AA}^2/\text{s}^2$) in the math:z direction

Only the group selected via the *compute_dos* keyword is included in the output.

5.3.14 movie.xyz

This file contains the positions (trajectory) of the atoms sampled at a given frequency. It is generated 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.

5.3.15 omega2.out

This file contains the squared frequencies $\omega^2(\mathbf{k})$ at different $\mathbf{k}$ -points.

File format

There are `N_kpoints` rows and `3 * N_basis + 1` columns, where `N_basis` is the number of atoms in `model.xyz`, and `N_kpoints` is the number of k-points generated from high-symmetry points in the [kpoints.in input file](#).

Each line corresponds to one $\mathbf{k}$ -point. Each line contains `3 * N_basis + 1` numbers, the first column represents the relative distance along the high-symmetry path, while the remaining columns correspond to the `3 * N_basis` values for $\omega^2(\mathbf{k})$ (in units of THz²) in the order of increasing magnitude.

5.3.16 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.

5.3.17 dump.xyz

File containing atomistic positions, velocities and forces. It is generated when invoking the *dump_xyz keyword*.

File format

This file is in the *extended XYZ format*. The output mode for this file is *append*.

5.3.18 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*.

5.3.19 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.

5.3.20 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
```

5.3.21 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
```

5.3.22 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.

5.3.23 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.

5.3.24 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 $\text{\AA}^2/\text{ps}^2$) in the x direction
- column 3: VAC (in units of $\text{\AA}^2/\text{ps}^2$) in the y direction
- column 4: VAC (in units of $\text{\AA}^2/\text{ps}^2$) in the z direction
- column 5: SDC (in units of $\text{\AA}^2/\text{ps}$) in the x direction
- column 6: SDC (in units of $\text{\AA}^2/\text{ps}$) in the y direction
- column 7: SDC (in units of $\text{\AA}^2/\text{ps}$) in the z direction

Only the group selected via the arguments of the `compute_sdc_keyword` is included in this output.

5.3.25 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 $\text{\AA}^2$) in the x direction
- column 3: MSD (in units of $\text{\AA}^2$) in the y direction
- column 4: MSD (in units of $\text{\AA}^2$) in the z direction
- column 5: SDC (in units of $\text{\AA}^2/\text{ps}$) in the x direction
- column 6: SDC (in units of $\text{\AA}^2/\text{ps}$) in the y direction
- column 7: SDC (in units of $\text{\AA}^2/\text{ps}$) 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.

5.3.26 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 $2*N_c-1 + \text{num_omega}$ rows. Here, N_c is the number of correlation steps and num_omega is the number of frequency points.

In the first $2*N_c-1$ rows:

- column 1: correlation time t from negative to positive, in units of ps
- column 2: $K^{\text{in}}(t)$ in units of $\text{\AA eV/ps}$
- column 3: $K^{\text{out}}(t)$ in units of $\text{\AA eV/ps}$

$K^{\text{in}}(t) + K^{\text{out}}(t) = K(t)$ 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: $J_q^{\text{in}}(\omega)$ in units of $\text{\AA eV/ps/THz}$
- column 3: $J_q^{\text{out}}(\omega)$ in units of $\text{\AA eV/ps/THz}$

$J_q^{\text{in}}(\omega) + J_q^{\text{out}}(\omega) = J_q(\omega)$ is exactly the left expression in Eq. (20) of [Fan2019].

Only the potential part of the heat current has been included.

If `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.

5.3.27 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:

$$\mathbf{a} = a_x \mathbf{e}_x + a_y \mathbf{e}_y + a_z \mathbf{e}_z$$

$$\mathbf{b} = b_x \mathbf{e}_x + b_y \mathbf{e}_y + b_z \mathbf{e}_z$$

$$\mathbf{c} = c_x \mathbf{e}_x + c_y \mathbf{e}_y + c_z \mathbf{e}_z$$

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.

5.3.28 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.

5.3.29 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: $\eta_{xx} = \frac{1}{k_B TV} \int_0^t \langle S_{xx}(0)S_{xx}(t') \rangle dt'$ (in units of Pa s)
- column 12: $\eta_{yy} = \frac{1}{k_B TV} \int_0^t \langle S_{yy}(0)S_{yy}(t') \rangle dt'$ (in units of Pa s)
- column 13: $\eta_{zz} = \frac{1}{k_B TV} \int_0^t \langle S_{zz}(0)S_{zz}(t') \rangle dt'$ (in units of Pa s)
- column 14: $\eta_{xy} = \frac{1}{k_B TV} \int_0^t \langle S_{xy}(0)S_{xy}(t') \rangle dt'$ (in units of Pa s)
- column 15: $\eta_{xz} = \frac{1}{k_B TV} \int_0^t \langle S_{xz}(0)S_{xz}(t') \rangle dt'$ (in units of Pa s)
- column 16: $\eta_{yz} = \frac{1}{k_B TV} \int_0^t \langle S_{yz}(0)S_{yz}(t') \rangle dt'$ (in units of Pa s)
- column 17: $\eta_{yx} = \frac{1}{k_B TV} \int_0^t \langle S_{yx}(0)S_{yx}(t') \rangle dt'$ (in units of Pa s)
- column 18: $\eta_{zx} = \frac{1}{k_B TV} \int_0^t \langle S_{zx}(0)S_{zx}(t') \rangle dt'$ (in units of Pa s)
- column 19: $\eta_{zy} = \frac{1}{k_B TV} \int_0^t \langle S_{zy}(0)S_{zy}(t') \rangle dt'$ (in units of Pa s)

Note:

- The shear viscosity can be calculated as $\eta_S = \frac{1}{3} (\eta_{xy} + \eta_{xz} + \eta_{yz})$.
- The longitudinal viscosity can be calculated as $\eta_L = \frac{1}{3} (\eta_{xx} + \eta_{yy} + \eta_{zz})$.
- The bulk viscosity η_B can be calculated from $\eta_B + \frac{4}{3}\eta_S = \eta_L$.
- We have the following symmetric property, $\eta_{\alpha\beta} = \eta_{\beta\alpha}$, which should be confirmed by the output data.

5.3.30 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: $L_{x\mu}^{qq}(t)$ (in units of W/mK)
- column 2: $L_{y\mu}^{qq}(t)$ (in units of W/mK)
- column 3: $L_{z\mu}^{qq}(t)$ (in units of W/mK)
- column 4: $L_{x\mu}^{1q}(t)$ (in units of $10^{-6}kg/smK$)
- column 5: $L_{y\mu}^{1q}(t)$ (in units of $10^{-6}kg/s/m/K$)
- column 6: $L_{z\mu}^{1q}(t)$ (in units of $10^{-6}kg/s/m/K$)
- ...
- column $3M+1$: $L_{x\mu}^{Mq}(t)$ (in units of $10^{-6}kg/smK$)
- column $3M+2$: $L_{y\mu}^{Mq}(t)$ (in units of $10^{-6}kg/smK$)
- column $3M+3$: $L_{z\mu}^{Mq}(t)$ (in units of $10^{-6}kg/smK$)

If the dissipative flux is changed to momentum flux of component α , this file reads:

- column 1: $L_{x\mu}^{q\alpha}(t)$ (in units of $10^{-6}kg/smK$)
- column 2: $L_{y\mu}^{q\alpha}(t)$ (in units of $10^{-6}kg/smK$)
- column 3: $L_{z\mu}^{q\alpha}(t)$ (in units of $10^{-6}kg/smK$)
- column 4: $L_{x\mu}^{1\alpha}(t)$ (in units of $10^{-12}kgs/m^3K$)
- column 5: $L_{y\mu}^{1\alpha}(t)$ (in units of $10^{-12}kgs/m^3K$)
- column 6: $L_{z\mu}^{1\alpha}(t)$ (in units of $10^{-12}kgs/m^3K$)
- ...
- column $3M + 1$: $L_{x\mu}^{M\alpha}(t)$ (in units of $10^{-12}kgs/m^3K$)
- column $3M + 2$: $L_{y\mu}^{M\alpha}(t)$ (in units of $10^{-12}kgs/m^3K$)
- column $3M + 3$: $L_{z\mu}^{M\alpha}(t)$ (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 coefficient Λ_{ij}^{ml} by:

$$\Lambda_{ij}^{ml} = T^2 L_{ij}^{ml}$$

The thermal conductivity can be derived from the onsager matrix Λ , that is:

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

5.3.31 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

5.3.32 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

5.3.33 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

5.3.34 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.

5.3.35 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.

5.3.36 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.

5.3.37 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 $-\pi$ 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
...
```

5.3.38 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: q14 q16 w14 w16
- **Subsequent lines**: per-atom order parameter values for the given step.

NEP EXECUTABLE

6.1 Input files

To run a *NEP* construction using the `nep` executable, one has to prepare three input files that respectively specify 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
<i>version</i>	select the NEP version
<i>type</i>	number of atom types and list of chemical species
<i>type_weight</i>	force weights for different atom types
<i>model_type</i>	select to train potential, dipole, or polarizability
<i>charge_mode</i>	select the charge mode for a potential model
<i>prediction</i>	select between training and prediction (inference)
<i>zbl</i>	outer cutoff for the universal <i>ZBL</i> potential [Ziegler1985]
<i>cutoff</i>	radial (r_c^R) and angular (r_c^A) cutoffs
<i>n_max</i>	size of radial ($n_{\max}^R$) and angular ($n_{\max}^A$) basis
<i>basis_size</i>	number of radial (N_{bas}^R) and angular (N_{bas}^A) basis functions
<i>l_max</i>	expansion order for angular terms
<i>neuron</i>	number of neurons in the hidden layer (N_{neu})
<i>lambda_1</i>	weight of $\mathcal{L}_1$ -norm regularization term
<i>lambda_2</i>	weight of $\mathcal{L}_2$ -norm regularization term
<i>lambda_e</i>	weight of energy loss term
<i>lambda_f</i>	weight of force loss term
<i>lambda_v</i>	weight of virial loss term
<i>atomic_v</i>	fit atomic or global virial
<i>force_delta</i>	bias term that can be used to make smaller forces more accurate
<i>batch</i>	batch size for training
<i>population</i>	population size used in the <i>SNES</i> algorithm [Schaul2011]
<i>generation</i>	number of generations used by the <i>SNES</i> 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        # the only option
cutoff    8 4      # please choose these reasonably
n_max     6 6      # default
basis_size 6 6     # default
l_max     4 1      # 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:

$$\mathbf{a} = a_x \mathbf{e}_x + a_y \mathbf{e}_y + a_z \mathbf{e}_z$$

$$\mathbf{b} = b_x \mathbf{e}_x + b_y \mathbf{e}_y + b_z \mathbf{e}_z$$

$$\mathbf{c} = c_x \mathbf{e}_x + c_y \mathbf{e}_y + c_z \mathbf{e}_z$$

- `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/Å³. 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
 - `bec:R:9` target Born effective charge (*BEC*) tensor (optional)
- 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.
- *BEC* is in units of elementary charge e .

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.
- The *BEC* is optional. You can also have it only for some structures.

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 only be 4, corresponding to NEP4.

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_{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 $r_c^{\text{ZBL-outer}}$, beyond which the *ZBL* potential is zero. 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, $r_c^{\text{ZBL-inner}} = r_c^{\text{ZBL-outer}}/2$, 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 $r_c^{\text{ZBL-outer}}$ is irrelevant.

Permissible values are $1 \text{ \AA} \leq r_c^{\text{ZBL-outer}} \leq 3 \text{ \AA}$

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 0.7.

If this keyword is present, the outer ZBL cutoff between two elements is the minimum between the global outer ZBL cutoff $r_{\text{outer}}^{\text{ZBL}}$ and <factor> times of the sum of the covalent radii of the two elements, and the inner ZBL cutoff is always set to 0.

By default, this keyword is not in effect.

A good usage is to first set up a global ZBL cutoff of 2.5 Å and then enable this feature:

```
zbl 2.5
use_typewise_cutoff_zbl 0.7
```

6.2.8 cutoff

This keyword enables one to specify the radial (r_c^{R}) and angular (r_c^{A}) cutoffs of the *NEP* model. One syntax is:

```
cutoff <radial_cutoff> <angular_cutoff>
```

where <radial_cutoff> and <angular_cutoff> correspond to r_c^{R} and r_c^{A} , respectively. The cutoffs must satisfy the conditions $3 \text{ \AA} \leq r_c^{\text{A}} \leq r_c^{\text{R}} \leq 10 \text{ \AA}$.

The defaults are $r_c^{\text{R}} = 8 \text{ \AA}$ and $r_c^{\text{A}} = 4 \text{ \AA}$. It can be computationally beneficial to use (possibly much) smaller r_c^{R} but the default values should be reasonable in most cases.

Another syntax is:

```
cutoff <radial_cutoff_species_1> <angular_cutoff_species_1> <radial_cutoff_species_2>
↪<angular_cutoff_species_2> ...
```

which can be used to specify a set of radial and angular cutoffs for each species. The cutoff between two species (a and b) is the arithmetic average of the cutoffs for the two species:

$$r_c^{R/A}(a, b) = r_c^{R/A}(b, a) = \frac{r_c^{R/A}(a) + r_c^{R/A}(b)}{2}$$

6.2.9 n_max

This keyword sets the number of radial ($n_{\max}^R$) and angular ($n_{\max}^A$) 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 $n_{\max}^R$ and $n_{\max}^A$, respectively. The two parameters must satisfy $0 \leq n_{\max}^R \leq 12$ and $0 \leq n_{\max}^A \leq 8$.

The default values of $n_{\max}^R = 6$ and $n_{\max}^A = 6$ are usually sufficient.

Note: These parameters should not be confused with N_{bas}^R and N_{bas}^A , which are set via the *basis_size* keyword.

6.2.10 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 $\langle N_{\text{bas_R}} \rangle$ and $\langle N_{\text{bas_A}} \rangle$ set N_{bas}^R and N_{bas}^A , respectively. The parameters must satisfy $0 \leq N_{\text{bas}}^R \leq 8$ and $0 \leq N_{\text{bas}}^A \leq 8$.

The default values of $N_{\text{bas}}^R = 6$ and $N_{\text{bas}}^A = 6$ are usually sufficient.

Note: These parameters should not be confused with $n_{\max}^R$ and $n_{\max}^A$, which are set via the *n_max* keyword.

6.2.11 l_max

It sets the angular descriptors, see Sect. II.C of [Fan2022b]. The syntax is:

```
l_max <l_max_3b> {<has_q_222> <has_q_1111> <has_q_112> <has_q_123> <has_q_233> <has_q_
↪134>}
```

where $l_{\max_3b}$ sets the limit for the three-body descriptors, has_q_222 , has_q_112 , has_q_123 , has_q_233 , and has_q_134 optionally specify which four-body descriptors are used, and has_q_1111 specifies if the five-body descriptor is used.

$l_{\max}^{3b}$ can take values from 2 to 8, and has_q_222 , has_q_1111 , has_q_112 , has_q_123 , has_q_233 , and has_q_134 can be 0 (not to use) or 1 (to use).

The default values are:

```
l_max 4 1
l_max 4 1 0 0 0 0 0 # equivalent way to write it
```

The five-body descriptor switch `has_q_1111` is only provided for backward compatibility. This five-body descriptor is equivalent to the square of two three-body descriptors and thus does not provide new information. Its use is strongly discouraged.

The new four-body descriptors (`q_112`, `q_123`, `q_233`, and `q_134`) have not been comprehensively tested, and should be used with caution. More suggestions on this will be made in a future version.

6.2.12 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_{neu} in the *NEP formalism* [Fan2022b]. Values must satisfy $1 \leq N_{\text{neu}} \leq 120$.

The default value is 30, which is relatively small but can lead to relatively high speed. Larger values rarely lead to a notable improvement.

6.2.13 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.

6.2.14 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.

6.2.15 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.

6.2.16 `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 $\lambda_f = 1.0$.

6.2.17 `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 $\lambda_v = 0.1$.

6.2.18 `lambda_q`

This keyword sets the weight λ_q of the loss term associated with the **total charge** (the charge for a whole structure) in the *loss function*. The syntax is:

```
lambda_q <weight>
```

Here, <weight> represents λ_q , which must satisfy $\lambda_q \geq 0$ and defaults to $\lambda_q = 0.1$.

This keyword is only relevant for the qNEP models.

6.2.19 `lambda_z`

This keyword sets the weight λ_z of the loss term associated with the **Born effective charge** in the *loss function*. The syntax is:

```
lambda_z <weight>
```

Here, <weight> represents λ_z , which must satisfy $\lambda_z \geq 0$ and defaults to $\lambda_z = 0.5$.

This keyword is only relevant for the qNEP models.

6.2.20 `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

6.2.21 `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.

6.2.22 `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/Å and defaults to $\delta = 0$ eV/Å (i.e., no bias).

When $\delta = 0$ eV/Å, the *loss term associated with the forces* is proportional to the *RMSE* of the forces:

$$\sqrt{\frac{1}{3N} \sum_{i=1}^N \left(\mathbf{F}_i^{\text{NEP}} - \mathbf{F}_i^{\text{tar}} \right)^2}$$

When $\delta > 0$ eV/Å, this expression is modified to read:

$$\sqrt{\frac{1}{3N} \sum_{i=1}^N \left(\mathbf{F}_i^{\text{NEP}} - \mathbf{F}_i^{\text{tar}} \right)^2 \frac{1}{1 + \|\mathbf{F}_i^{\text{tar}}\|/\delta}}$$

In this case, a smaller δ implies a larger weight on smaller forces.

6.2.23 `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 $N_{\text{bat}} \geq 1$ and defaults to $N_{\text{bat}} = 1000$.

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 $N_{\text{bat}} = 100$ and $N_{\text{bat}} = 1000$ 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 $N_{\text{bat}} = 1000$) or the full-batch ($N_{\text{bat}} \geq$ 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.

6.2.24 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 $10 \leq N_{\text{pop}} \leq 100$ and defaults to $N_{\text{pop}} = 50$.

6.2.25 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_{gen} , which must satisfy $0 \leq N_{\text{gen}} \leq 10^7$ and defaults to $N_{\text{gen}} = 10^5$. For simple systems, $N_{\text{gen}} = 10^4 \sim 10^5$ is enough. For more complicated systems, values in the range $N_{\text{gen}} = 10^5 \sim 10^6$ can be required to obtain a well converged model.

6.2.26 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.

6.2.27 output_interval

This keyword sets the number of generations between writes to *loss.out* (and the corresponding console output, `nep.txt`, and test-set output files).

The syntax is:

```
output_interval <number_of_generations>
```

Here, <number_of_generations> must be a positive integer and defaults to 100.

6.2.28 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

6.2.29 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
#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
```

(continues on next page)

(continued from previous page)

```
batch      5000
population 50
save_potential 1000 0
generation 5000
```

6.2.30 import_q_scaler

By default, when training or resuming a NEP potential without `fine_tune`, **nep** computes `q_scaler`, the per-descriptor-component normalization used by the neural network, from the training structures of the current run at generation 0. This keyword instead tells **nep** to read `q_scaler` from the `nep.txt` file present in the run directory, skipping that computation. This naturally requires such a file to be present. The syntax is:

```
import_q_scaler <0 or 1>
```

with a default value of 0.

This is intended for resuming training from a restart state (`nep.restart`) that was transplanted from a different model, e.g. a foundation model restricted to a subset of species via a species-selection tool. In that case, the transplanted network weights were tuned under the `q_scaler` of the original model, and recomputing a different one from a smaller or less diverse training set can force the weights to spend part of the optimization readapting to the new descriptor scale, rather than only to the new training data. Unlike `fine_tune`, this keyword does not perform any species extraction: the `nep.txt` file it reads from is expected to already have the exact same architecture and species as the current run (**nep** validates this and raises an error on a mismatch), which is why the file is referred to implicitly rather than by a path argument, matching the way `nep.restart` is referred to. The mean/standard-deviation restart state (`mu/sigma`) is unaffected by this keyword: it is unconditionally read from `nep.restart`, exactly as in any other non-`fine_tune` run.

6.2.31 charge_mode

This keyword allows one to specify the charge mode of a NEP4 potential model to be trained.

The syntax is:

```
charge_mode <mode_value>
```

where `<mode_value>` must be an integer that can assume one of the following values.

Value	Model description
0	original NEP without charge
1	qNEP model with charge, including both real- and reciprocal-space contributions
2	qNEP model with charge, including reciprocal-space contribution only

Here, qNEP means NEP with charge (q) [Fan2026]. Mode 1 is consistent with the approach first proposed by Song et al. [Song2024b]. Mode 2 is consistent with the approach first proposed by Cheng [Cheng2025]. To train a qNEP model, we do not need to provide target charges. One can choose to provide target Born effective charges (*BEC*), but this usually only works for a single matter in a single phase, as we have assumed a constant high-frequency dielectric constant for the whole training dataset.

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
<i>loss.out</i>	loss function, regularization terms, and <i>RMSE</i> data as a function of generation
<i>nep.txt</i>	<i>NEP</i> potential parameters
<i>nep.restart</i>	restart file
<i>energy_train.out</i>	target and predicted energies for training data set
<i>energy_test.out</i>	target and predicted energies for test data set
<i>force_train.out</i>	target and predicted forces for training data set
<i>force_test.out</i>	target and predicted forces for test data set
<i>virial_train.out</i>	target and predicted virials for training data set
<i>virial_test.out</i>	target and predicted virials for test data set
<i>stress_train.out</i>	target and predicted stress values for training data set
<i>stress_test.out</i>	target and predicted stress values for test data set
<i>dipole_train.out</i>	target and predicted dipole values for training data set
<i>dipole_test.out</i>	target and predicted dipole values for test data set
<i>polarizability_train.out</i>	target and predicted polarizability values for training data set
<i>polarizability_test.out</i>	target and predicted polarizability values for test data set
<i>charge_train.out</i>	predicted charge values for training data set
<i>charge_test.out</i>	predicted charge values for test data set
<i>bec_train.out</i>	target and predicted <i>BEC</i> values for training data set
<i>bec_test.out</i>	target and predicted <i>BEC</i> values for test data set
<i>descriptor.out</i>	descriptor values for training data set in prediction mode

6.3.1 loss.out

This file contains the terms that enter the *loss function*, written every *output_interval* generations (100 by default).

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 charge_*.out

The `charge_train.out` and `charge_test.out` files contain the predicted partial charges of the configurations provided in the *train.xyz* and *test.xyz* input files.

There is a single column. Each row gives the predicted charge of one atom in units of the elementary charge e . The order of the atoms is consistent with the training/test dataset for the prediction mode or the full-batch training mode.

6.3.11 bec_*.out

The `bec_train.out` and `bec_test.out` files contain the predicted and target Born effective charges (*BEC*) of the configurations provided in the *train.xyz* and *test.xyz* input files.

There are 18 columns. The first 9 columns are the predicted *BEC* components in units of e . The last 9 columns are the corresponding target values. The 9 columns are arranged in the order of 11, 12, 13, 21, 22, 23, 31, 32, 33. Each row corresponds to one atom. The order of the atoms is consistent with the training/test dataset for the prediction mode or the full-batch training mode. For a structure without target *BEC*, a target value of 0 for each atom will be output.

6.3.12 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).

CREDITS

TBC

7.1 Acknowledgments

- NSFC with project numbers 11404033 and 11974059
- Aalto Science-IT project
- Finland's IT Center for Science (CSC)

BIBLIOGRAPHY

PUBLICATIONS

9.1 2026

1. Yu Bao, Keke Song, Jiahui Liu, Yanzhou Wang, Yifei Ning, Penghua Ying, and Ping Qian. Atomistic understanding of hydrogen bubble-induced embrittlement in tungsten enabled by machine learning molecular dynamics. *npj Computational Materials*, 12(1):108, 2026. doi:10.1038/s41524-026-01986-2.
2. Ethan Berger, Tobias Hainer, Tobias Möslinger, Paul Erhart, and Julia Wiktor. Stability of polarons and oxygen vacancies in BiVO₄ revealed by machine-learning-driven simulations. *PRX Energy*, 5:033004, Jul 2026. doi:10.1103/xwxxv-6s55.
3. Tieyuan Bian, Wenjia Zhu, Qiong Lei, and Jun Yin. Machine learning potentials for inorganic and hybrid lead halide perovskites: From phase stability to defects and interfaces. *ACS Applied Materials & Interfaces*, 18(12):17219–17238, 2026. doi:10.1021/acsami.5c24460.
4. Hekai Bu, Wenwu Jiang, Penghua Ying, Ting Liang, Zheyong Fan, and Wengen Ouyang. Modular hybrid machine learning and physics-based potentials for scalable modeling of van der waals heterostructures. *Journal of the Mechanics and Physics of Solids*, 210:106540, 2026. doi:10.1016/j.jmps.2026.106540.
5. Xitai Cai, Yuxun Wu, Libo Li, Lijun Liao, Penghua Ying, and Yanying Wei. HULU: A unified monte carlo framework for adsorption simulations with machine learning potentials. *Nano Research*, 19(4):94908548, 2026. doi:10.26599/NR.2026.94908548.
6. Bin Cao, Boqia Ju, Gang Liu, Ge Yao, Jiuyu Sun, and Yongping Du. Temperature-stable thermal transport in graphene antidot lattices: Phonon scattering regulation. *Chemical Physics Letters*, 890:142740, 2026. doi:10.1016/j.cplett.2026.142740.
7. Chenyang Cao, Xinfang Jia, Hongfei Li, Shuo Cao, Yanzhou Wang, and Ping Qian. Efficient molecular dynamics of primary radiation damage in vanadium using neuroevolution potential. *Journal of Nuclear Materials*, 630:156726, 2026. doi:10.1016/j.jnucmat.2026.156726.
8. Shuo Cao, Benrui Tang, Zezhu Zeng, Zheyong Fan, Ye Su, and Yu Yan. Intrinsic and tunable lattice thermal conductivity of single- and multi-layer goldene: A machine-learning molecular dynamics study. *Journal of Applied Physics*, 139(2):025103, 01 2026. doi:10.1063/5.0300239.
9. Wenlong Cao, Caiyu Chen, Hai Liu, Lei Yang, Tianlong Tan, Ruiyang Xu, Chengjian Ma, Tianyu Wang, Daniel Hedman, Shigeo Maruyama, Rong Xiang, Jue Wang, and Qinfang Zhang. Interfacial engineering of core-shell Cu@Zn nanoclusters enables methane selective CO₂ electroreduction via accelerated water dissociation. *Chemical Engineering Journal*, 541:177779, 2026. doi:10.1016/j.cej.2026.177779.
10. Junwei Che, Guoliang Ren, Xuezhi Wang, and Shengli Zhang. Revisiting phonon thermal transport in perovskite CsPbBr₃ crystals: Critical role of temperature-dependent quartic anharmonicity. *Applied Physics Letters*, 128(24):241904, 06 2026. doi:10.1063/5.0324950.

11. Junwei Che and Xuezhi Wang. Revisiting phonon thermal transport in penta-graphene via a machine-learning potential-driven large-scale molecular dynamics simulation. *Computational Materials Science*, 267:114606, 2026. doi:10.1016/j.commatsci.2026.114606.
12. Hao Chen, Yang Tao, Cheng Xin Li, and Han Xiao. Atomistic origin of faulting, twinning, and work hardening in CuSn10P1 phosphor bronze revealed by in-situ characterization and machine-learning molecular dynamics. *Journal of Alloys and Compounds*, 1074:189180, 2026. doi:10.1016/j.jallcom.2026.189180.
13. Hao Chen, Yang Tao, Cheng Xin Li, and Han Xiao. Sequential transformation mediated twinning in a low stacking-fault energy Cu-Sn-P alloy: The essential precursor role of the 9R phase. *Journal of Alloys and Compounds*, 1062:187653, 2026. doi:10.1016/j.jallcom.2026.187653.
14. Ming Chen, Yihao Wang, Junjie Yang, Yuchi Cui, Shengyi Zhong, and Zhe Chen. Potassium bubble-dislocation interaction mechanisms and dynamic rupture effects: An atomistic study based on machine learning potentials. *Journal of Nuclear Materials*, 630:156704, 2026. doi:10.1016/j.jnucmat.2026.156704.
15. Qiao Chen, Dongyang Li, Bing Yang, Yunqing Tang, and Lin Li. Neural networks accelerate discovery of ultralow thermal conductivity configurations in disordered graphene/h-BN multilayers. *Applied Thermal Engineering*, 292:130338, 2026. doi:10.1016/j.applthermaleng.2026.130338.
16. Yuanyuan Chen, Shuo Cao, Yu Bao, KeKe Song, Ping Qian, and Yanjing Su. Machine learning potential-assisted design for thermoelectric performance in anharmonic CsPbBr₃ and CsPbI₃. *J. Mater. Chem. C*, 14:8701–8714, 2026. doi:10.1039/D6TC00498A.
17. Yuanyuan Chen, Shuo Cao, Chenyang Cao, Yu Bao, Keke Song, Min Zhong, Ping Qian, and Yanjing Su. Insight into ultra-low lattice thermal conductivity and native defect behavior in lead-free perovskite Cs₃Sb₂I₉. *Journal of Alloys and Compounds*, 1071:188734, 2026. doi:10.1016/j.jallcom.2026.188734.
18. Yuxuan Chen, Qing-an Li, Hanpu Liang, Heng Kang, Guanchen Dong, Kexin Zhang, Lin Wang, Huanrong Liu, Shan Zhang, Jian Li, Bin Xu, Xuelin Yang, Shiwu Gao, Rui Su, and Pengfei Guan. Atomic-bond-modulated thermal transport in Si/AlN heterointerfaces revealed via machine-learning potentials. *Acta Materialia*, 312:122241, 2026. doi:10.1016/j.actamat.2026.122241.
19. Haixia Cheng, Yanzhou Wang, Keke Song, Minhui Song, Yaqing Hou, and Zhongnan Bi. Machine-learned interatomic potential for Fe-Co-X (X = Re, W) alloys: Ductility-brittleness behavior at finite temperatures. *Materials & Design*, 263:115604, 2026. doi:10.1016/j.matdes.2026.115604.
20. Mo Cheng, Xuanyu Jiang, Xiaodong Pi, Deren Yang, and Tianqi Deng. Dislocation-limited thermal transport in silicon carbide dislocations from machine-learning molecular dynamics. *Chinese Physics B*, 2026. doi:10.1088/1674-1056/ae48c5.
21. Alfredo Ranes Coro III, Hao-Jen You, Rovi Angelo Beloya Villaos, Zhi-Quan Huang, Liang-Zi Yao, Gengchiao Liang, Hsin Lin, and Feng-Chuan Chuang. The high-value thermoelectric properties of ruby-silver ores Ag₃XS₃ (X = As, Sb): A combined density functional theory and machine learning approach. *ACS Applied Energy Materials*, 2026. doi:10.1021/acsaeam.6c01192.
22. Wenjun Cui, Cheng Qian, Weixiao Lin, Zefan Xue, Zhencui Ge, Wen Zhao, Gustaaf Van Tendeloo, Jinsong Wu, Feng Ding, Xiahan Sang, and Zhengyi Fu. Geometrically driven reversible solid-liquid phase transition at the atomic scale. *Science*, 393(6808):280–286, 2026. doi:10.1126/science.aed6019.
23. Chi Ding, Yu Han, Jiuyang Shi, Hao Gao, Qiuhan Jia, Ziyang Yang, Junjie Wang, Hui-Tian Wang, Dingyu Xing, and Jian Sun. Monolayer methane hydrate formation in 2D confinement with multiple plastic phases and low superionic pressure. *Science China Physics, Mechanics & Astronomy*, 69(7):276811, 2026. doi:10.1007/s11433-025-2848-2.
24. Hairui Ding, Artem R. Oganov, Haixu Cui, Jiachang Zhang, and Xiao Dong. Phonons and the lattice dynamics in crystals with atomic diffusion: Concept and application to superionic ices. *Phys. Rev. B*, 114:024101, Jul 2026. doi:10.1103/m1s6-6hdf.
25. Zheyong Fan, Benrui Tang, Esmée Berger, Ethan Berger, Erik Fransson, Ke Xu, Zihan Yan, Zhoulin Liu, Zichen Song, Haikuan Dong, Shunda Chen, Lei Li, Ziliang Wang, Yizhou Zhu, Julia Wiktor, and Paul Erhart. qNEP:

- A highly efficient neuroevolution potential with dynamic charges for large-scale atomistic simulations. *Journal of Chemical Theory and Computation*, 22(9):4787–4801, 2026. doi:10.1021/acs.jctc.6c00146.
26. Zheyong Fan, Wenjun Zhang, Zhenhao Zhang, Ke Xu, Xuecheng Shao, and Haikuan Dong. NEP-CG and NEP-AACG: Efficient coarse-grained and multiscale all-atom-coarse-grained neuroevolution potentials. *Computational Materials Today*, 10:100055, 2026. doi:10.1016/j.commt.2026.100055.
 27. Yilin Fang, Weiyi Li, Guiyun Hang, Jintao Wang, Xiyao Yun, Wanxiao Guo, Tao Wang, and Wenli Yu. New insights into thermal transport in ϵ -CL-20 revealed by machine-learned potentials and mode-projection analysis. *The Journal of Physical Chemistry A*, 130(22):4120–4135, 2026. doi:10.1021/acs.jpca.6c00862.
 28. Xingze Geng, Wentao Zhang, Lin-Wang Wang, and Xiangying Meng. Origin of the machine learning forces field errors across metal elements. *npj Computational Materials*, 12:102, 2026. doi:10.1038/s41524-026-01977-3.
 29. Yaohui Gu, Binbo Li, Linyang Jiang, Yuhui Hu, Wenqiang Liu, Lijun Xu, Pengfei Zhai, Haizhou Xue, Jie Liu, and Jinglai Duan. A neuroevolution potential for gallium oxide: Accurate and efficient modeling of polymorphism and swift heavy-ion irradiation. *Applied Physics Reviews*, 13(3):031404, 2026. doi:10.1063/5.0323486.
 30. Sanqi Guo, Dinggen Li, and Bo Xu. Molecular dynamics study on effect of crystal defects in ncm811 cathode structure based on machine learning potential. *Journal of Power Sources*, 665:239008, 2026. doi:10.1016/j.jpowsour.2025.239008.
 31. Shijie Guo, Qijun Wang, Zijun Qi, Zhanpeng Sun, Jingyuan Ni, Bingkun Chen, Rui Li, Wei Shen, and Gai Wu. Shear-strain-dependent thermal conductivity of diamond explored via a machine learning potential. *Applied Physics Letters*, 128(23):231905, 06 2026. doi:10.1063/5.0336299.
 32. Haimei Han, Weijie Lin, Man Zhou, Chongtao Wei, and Ping Zhang. Molecular dynamics simulation of Mo-Cu diffusion bonding mechanism based on NEP+ZBL potential function. *Computational Materials Science*, 262:114340, 2026. doi:10.1016/j.commatsci.2025.114340.
 33. Yue Han, Shun Li, Xuping Wang, Lei Wei, Haozhi Zhang, Xiaochen Liu, Shenao Wang, Yuanyuan Zhang, Xianshun Lv, and Bing Liu. Broad-spectrum phonon suppression by ferroelectric domain walls for tunable thermal conductivity in BiFeO₃. *Physica Scripta*, 101(21):216004, may 2026. doi:10.1088/1402-4896/ae6d49.
 34. Yuzhou Hao, Yujie Liu, Xuejie Li, Turab Lookman, Xiangdong Ding, Jun Sun, and Zhibin Gao. LLM-driven discovery for carbon allotropes with bond-network entropy. *Applied Physics Letters*, 128(10):102202, Mar 2026. doi:10.1063/5.0314279.
 35. Yuzhou Hao, Turab Lookman, Xiangdong Ding, Jun Sun, and Zhibin Gao. Role of octahedral tilting induced acoustic softening on limiting thermal transport in SrSnO₃. *Phys. Rev. B*, 113:075205, Feb 2026. doi:10.1103/76nt-hw3h.
 36. Q.F. He, S.H. Ma, D.H. Chung, H. Wang, Z.Q. Chen, Z.Y. Ding, Q. Yu, J.H. Luan, Q. Wang, F. Zhang, H.B. Lou, Q.S. Zeng, J.F. Gu, S.J. Zhao, and Y. Yang. Polytypic phase transformation of topologically close-packed phase enabled toughening in multi-principal-element eutectic alloy. *Acta Materialia*, 306:121737, 2026. doi:10.1016/j.actamat.2025.121737.
 37. Pengfei Hou, Yumiao Tian, Zifeng Liu, Junwen Duan, Hanyu Liu, Xing Meng, Russell J. Hemley, and Yanming Ma. Interface-dependent phase transitions and ultrafast hydrogen superionic diffusion of H₂O ice. *Phys. Rev. B*, 113:184103, May 2026. doi:10.1103/h61k-f3mj.
 38. Qifeng Hou and Chaoyang Zhang. Artificial intelligence for energetic materials: design on the levels of molecule, crystal and composite. *Chemical Engineering Journal*, 534:175131, 2026. doi:10.1016/j.cej.2026.175131.
 39. Qingmei Hu, Daniel Hedman, Ya Feng, Wanyu Dai, Daisuke Asa, Aina Fitó-Parera, Yixi Yao, Yongjia Zheng, Kaoru Hisama, Gunjan Auti, Hirofumi Daiguji, Shohei Chiashi, Dmitry Levshov, Wim Wenseleers, Keigo Otsuka, Yan Li, Christophe Bichara, Sofie Cambré, Rong Xiang, and Shigeo Maruyama. Structures of iron and cobalt bimetallic clusters for optimized chemical vapor deposition growth of single-walled carbon nanotubes. *Carbon*, 257:121679, 2026. doi:10.1016/j.carbon.2026.121679.

40. Zhi-Qiang Hu, Rui Liu, Jian-Li Shao, and Peng-Wan Chen. Anisotropic shock-induced hotspot and reaction kinetics in CL-20 crystal based on neuroevolution potential. *Combustion and Flame*, 289:114990, 2026. doi:10.1016/j.combustflame.2026.114990.
41. Zhi-Qiang Hu, Rui Liu, Jian-Li Shao, and Peng-Wan Chen. Influence of guest molecules on shock-induced intrinsic reactivity and reaction pathways of CL-20 based on neuroevolution potential. *The Journal of Chemical Physics*, 165(1):014502, 2026. doi:10.1063/5.0319640.
42. Dian Huang, Jun Lyu, Zhe Wu, Guihua Tang, and Lin Yang. Thermal conductivity modulation under strain gradients in III-V semiconductors. *Applied Surface Science*, 735:166672, 2026. doi:10.1016/j.apsusc.2026.166672.
43. Hongfu Huang, Junhao Peng, Kaiqi Li, Jian Zhou, and Zhimei Sun. Efficient gpu-accelerated training of a neuroevolution potential with analytical gradients. *Computer Physics Communications*, 320:109994, 2026. doi:10.1016/j.cpc.2025.109994.
44. Tianheng Huang, Chi Ding, Yu Han, Junjie Wang, Shuning Pan, Qing Lu, Chris J. Pickard, Hui-Tian Wang, Dingyu Xing, and Jian Sun. Stable iron hydrosilicates and their influence on planetary evolution. *Phys. Rev. B*, 113:184105, May 2026. doi:10.1103/nhw5-blzt.
45. Shengnan Ji, Qing Ai, Jialong Chen, Minyu Wang, Meng Liu, and Yong Shuai. Interfacial mechanical properties and optimization of GaN/diamond heterostructures via machine learning potential. *The Journal of Physical Chemistry C*, 130(18):6663–6673, 2026. doi:10.1021/acs.jpcc.6c00699.
46. Tian Jiang, Jie Xiong, Tianhao Su, Shuai Chen, and Tong-Yi Zhang. Multi-fidelity, active-learning assisted design of TaNbMoVW refractory high-entropy alloys with superior mechanical properties. *Journal of Physics: Materials*, 9(2):025003, 2026. doi:10.1088/2515-7639/ae44d1.
47. Wenwu Jiang, Hekai Bu, Kun Zhang, Bozhao Wu, and Wengen Ouyang. Chirality-dependent strain modulation of thermal conductivity in double-walled molybdenum disulfide nanotubes. *International Communications in Heat and Mass Transfer*, 170:109935, 2026. doi:10.1016/j.icheatmasstransfer.2025.109935.
48. Di Jin, Shicong Ding, Hailong Qiu, Yang Li, Xinyang Li, and Guochun Yang. Structural insights and predictive screening of ion transport in Li-rich alloys via neuroevolution potentials. *npj Computational Materials*, 12(1):132, 2026. doi:10.1038/s41524-026-02012-1.
49. Yu-Xuan Kang, Shi-Yun Xiong, and Hong-Liang Yi. Improve thermoelectric properties of graphene nanoribbons based on resonant structure engineering. *Chinese Physics B*, 35(3):037301, 2026. doi:10.1088/1674-1056/ae1950.
50. Prakriti Kayastha, Erik Fransson, Paul Erhart, and Lucy Whalley. Diverse polymorphism in Ruddlesden-Popper chalcogenides. *Phys. Rev. Lett.*, 136:086101, 2026. doi:10.1103/f4kv-pk93.
51. Sergey I. Kudryashov, Ivan M. Podlesnykh, Valery A. Dravin, Michael S. Kovalev, Pavel A. Chizhov, Vladislava V. Bulgakova, Alexander A. Ushakov, Yuri G. Goncharov, George K. Krasin, Evgeny V. Kuzmin, Anastasiya G. Bondarenko, Pavel A. Maslov, and Nikita D. Orekhov. Time-resolved non-contact THz diagnostics of impurity state in different sulfur-implanted/laser-annealed silicon samples. *The Journal of Chemical Physics*, 164(13):134203, 2026. doi:10.1063/5.0304765.
52. Shoulong Lai, Xigui Yang, Jiuyang Shi, Shijie Liu, Ying Guo, Longbin Yan, Jinhao Zang, Zhuangfei Zhang, Qiuhan Jia, Jian Sun, Shaobo Cheng, and Chongxin Shan. Bulk hexagonal diamond. *Nature*, 651(8106):621–625, 2026. doi:10.1038/s41586-026-10212-4.
53. Ossi Laurila, Tiia Jacklin, Ouail Zakary, and Perttu Lantto. Machine learning-accelerated path integral molecular dynamics and ^{13}C NMR simulations unlock new insights into quantum effects in C_{60} fullerene. *The Journal of Physical Chemistry A*, 130(10):2169–2181, 2026. doi:10.1021/acs.jpca.6c00238.
54. Kaiqi Li, Xuanguang Zhang, Hongming Chen, Bingke Xiang, Jian Zhou, Bin Liu, and Zhimei Sun. Atomistic insights into the full-cycle switching of Y-doped Sb_2Te_3 phase change memory. *Materials & Design*, 265:115941, 2026. doi:10.1016/j.matdes.2026.115941.

55. Ke Li, ZhongTing Zhang, ZhengYue Lin, YeYao Zhang, ChenXuan Liu, HengAn Wu, YinBo Zhu, and Hao Ma. Density-driven to phonon-dominated: Crossover in heat capacity of amorphous carbons. *ACS Nano*, 2026. doi:10.1021/acs.nano.6c00656.
56. Mingqian Li, Lifeng Wang, and Zhuoqun Zheng. Virial-matching in machine learning coarse-grained potential for multilayer hBN. *The Journal of Physical Chemistry C*, 2026. doi:10.1021/acs.jpcc.6c01847.
57. Qing Li, Haikuan Dong, Penghua Ying, and Zheyong Fan. Anisotropic and isotropic elasticity and thermal transport in monolayer C24 networks from machine-learning molecular dynamics. *International Journal of Heat and Mass Transfer*, 260:128505, 2026. doi:10.1016/j.ijheatmasstransfer.2026.128505.
58. Xiaolong Li, Weina Ren, Jincheng Yue, Yinong Liu, Jiayi Cao, Shiqian Hu, and Chunhua Zeng. Twist-induced suppression of flexural phonons and thermal conductivity in suspended and supported multilayer graphene. *Chinese Physics B*, 2026. doi:10.1088/1674-1056/ae5f01.
59. Yang Li, Mingyu Lei, Di Jin, Xinyang Li, Shicong Ding, Junlin Zhu, Bin Wen, and Guochun Yang. Discovery of Mg_3Zn_2 intermetallic phase in the Mg-Zn system enabled by neuroevolution potentials. *Inorganic Chemistry*, 2026. doi:10.1021/acs.inorgchem.6c00787.
60. Yiyi Li, Weitao Wang, Weikuan Li, Jianlian Huang, Yanping Qiu, Zhirong Shi, Wei Zhang, Yajuan Cheng, and Shiyun Xiong. Synergistic effects of twist angle and substrate coupling on thermal transport in twisted bilayer graphene. *Phys. Rev. B*, 113:125404, 2026. doi:10.1103/q617-thd2.
61. Zefeng Li, Kaiqi Li, Jian Zhou, and Zhimei Sun. Atomic origin of sluggish diffusion in NbMoTaW multi-principal element alloys. *Materials & Design*, 263:115675, 2026. doi:10.1016/j.matdes.2026.115675.
62. Zhixin Li, Xiao-Tong Li, Zhaoqi Chen, Yushan Geng, Hao Gong, Sijia Hu, Wenli Song, Wanshun Xia, Chuazheng Li, Linfa Peng, Yue Fan, Pengfei Guan, Bo Zhang, Weihua Wang, and Yong Yang. Transforming grain-boundary brittle precipitates to ductility pathways in complex concentrated alloy. *Advanced Science*, pages e18465, 2026. doi:10.1002/advs.202518465.
63. Chenjia Liang, Jun Yao, Xiaoxia Hou, Liang Li, NingZe Gao, Haoyu Lu, Ruiyao Zhao, Xiangke Guo, Nianhua Xue, Luming Peng, Xuefeng Guo, Yan Zhu, and Weiping Ding. Dynamic vibration-coupled energy transfer for boosting ORR catalysts in fuel cells. *Journal of the American Chemical Society*, 148(5):5115–5124, 2026. doi:10.1021/jacs.5c15936.
64. Wenkai Liao, Wei Cao, Ziyu Wang, Ling Miao, Jing Shi, and Rui Xiong. Theoretical investigation of grain-boundary engineering for thermal transport control in janus 2D transition metal dichalcogenides: Implications for nanoscale thermal management. *ACS Applied Nano Materials*, 2026. doi:10.1021/acsanm.6c00566.
65. Bin Liu, Zhiguo Tian, Alexander A. Barinov, and Moran Wang. Enhanced heat conduction in diamond/copper composites via interconnected structures: Machine learning molecular dynamics simulation. *International Journal of Thermal Sciences*, 221:110501, 2026. doi:10.1016/j.ijthermalsci.2025.110501.
66. Bin Liu, Zhiguo Tian, Alexander A. Barinov, and Moran Wang. Interfacial and coherent thermal transport of phonons in $\text{Bi}_2\text{Te}_3/\text{Sb}_2\text{Te}_3$ superlattices. *International Journal of Heat and Mass Transfer*, 258:128283, 2026. doi:10.1016/j.ijheatmasstransfer.2025.128283.
67. Bin Liu, Zhiguo Tian, Alexander A. Barinov, and Moran Wang. Wavelike thermal phonons revealed by localization in graphene phononic crystals. *Applied Physics Letters*, 128(11):112206, Mar 2026. doi:10.1063/5.0316529.
68. Fuzhu Liu, Rui Song, Ruizhi Qiu, Xiangdong Ding, and Jun Sun. Transferable machine learning potential for modeling uranium nitrides across stoichiometries. *Phys. Rev. Mater.*, 10:073801, Jul 2026. doi:10.1103/1rvq-1ct7.
69. Jie Liu and Lin Zhang. Insight into packing structures as well as electrical states for Ti nanoparticles on heating through integrating machine learning potential with a MD/QM hybrid strategy. *Advanced Theory and Simulations*, 9(1):e01312, 2026. doi:10.1002/adts.202501312.

70. Wenhao Liu, Hao Deng, Jueyi Qi, Xie Zhang, Yujing Liu, Zibo Zhao, Jun Zhu, Hengjun Luo, Hui Yin, Wei Xiang, Ning Zhang, Yulin Qin, Lvjun Zhou, Longqing Chen, and Yingjie Ma. Heterogeneous CRSS evolution and multistage twinning in near β titanium alloys during tensile deformation at cryogenic temperature. *Materials Science and Engineering: A*, 961:150198, 2026. doi:10.1016/j.msea.2026.150198.
71. Yiwen Liu, Hulei Yu, Lei Zhuang, and Yanhui Chu. Uncovering the thermal expansion in high-entropy ceramics by machine learning. *Journal of Materiomics*, pages 101205, 2026. doi:10.1016/j.jmat.2026.101205.
72. Yuqi Liu, Haikuan Dong, Ying Ren, Wen-Lai Huang, and Wei Chen. Molecular dynamics insights into CaF_2 single crystal growth using neuroevolution potentials. *Modelling and Simulation in Materials Science and Engineering*, 2026. doi:10.1088/1361-651X/ae5677.
73. Junfeng Lu, Huibo Chen, Yanlei Wang, and Hongyan He. Machine learning-driven neuroevolution potentials for imidazole ionic liquids: capturing ionic correlations and structure-property relationships. *Chemistry of Materials*, 2026. doi:10.1021/acs.chemmater.5c02624.
74. Jian Luo, Yinglong Hu, Xinran Zhang, and Hao Ma. Unraveling the microscopic mechanism of thermal conductivity in 2D COF-C_{60} assemblies via machine-learning potentials. *Applied Physics Letters*, 128(24):242202, 06 2026. doi:10.1063/5.0335720.
75. Wen-Hao Luo, Lei Zhang, Ke Chen, and Kun Cao. Water evaporation with machine learning interatomic potentials trained on MB-pol. *AIP Advances*, 16(4):045202, 2026. doi:10.1063/5.0311226.
76. Wenzhu Luo, Ershuai Yin, Lei Wang, Enjian Sun, and Qiang Li. Interfacial thermal transport of $\text{Al}_x\text{Ga}_{(1-x)}\text{N}$ -GaN heterostructures under strain via machine learning potential. *International Journal of Heat and Mass Transfer*, 260:128415, 2026. doi:10.1016/j.ijheatmasstransfer.2026.128415.
77. Guocai Lv, Qingyuan Kong, Hao Zhang, and Ping Qian. Primary radiation damage and defect clustering in molybdenum: A high-fidelity benchmark study based on the NEP-ZBL machine learning potential. *Modelling and Simulation in Materials Science and Engineering*, 2026. doi:10.1088/1361-651X/ae8563.
78. Jacob B. Mack and Christopher A. Schuh. Spatial distributions of grain boundary segregation - A combined atom probe and atomistic simulation study. *Acta Materialia*, 315:122398, 2026. doi:10.1016/j.actamat.2026.122398.
79. Soham Mandal, Ashutosh Srivastava, Tanmoy Das, Abhishek Kumar Singh, and Prabal K. Maiti. Probing lattice anharmonicity and thermal transport in ultralow- κ materials using machine learning interatomic potentials. *Small*, 22(8):e13476, 2026. doi:10.1002/smll.202513476.
80. Ruize Qi, Pingyang Zhang, Haowen Ma, Yihan Qin, and Xujiang Wang. Comparative simulations of CO_2 adsorption on diverse carbon materials via machine learning-based molecular dynamics. *Applied Surface Science*, 720:165199, 2026. doi:10.1016/j.apsusc.2025.165199.
81. Bin Qian, Yu Wang, Keyuan Xu, Jiahao Zu, Jiaqi Liu, Wei Liang, Fangli Yu, Yan Li, and Yu Bai. Reverse design of non-equimolar rare-earth monosilicates: Data-driven and molecular dynamics insights into CMAS corrosion mechanism. *Corrosion Science*, 268:113908, 2026. doi:10.1016/j.corsci.2026.113908.
82. Ziyang Qian, Guangwu Zhang, Zhanyu Lai, Ayaka Hanai, Yixin Xu, Guang Wang, Yang Lu, Jiaqi Gu, Yanguang Zhou, Takahiko Yanagitani, Junjun Jia, and Qiye Zheng. Tailoring thermal transport in $(\text{Sc}, \text{Yb})\text{AlN}$ thin films to the glassy limit. *Acta Materialia*, 304:121767, 2026. doi:10.1016/j.actamat.2025.121767.
83. Yihan Qin, Pingyang Zhang, Haoze Sun, Ziqi Zuo, Qiyuan Song, and Xujiang Wang. AI-driven insight of interfacial evolution and SEI formation in Br-doped sulfide solid electrolyte/Li metal anode. *Computational Materials Science*, 272:114868, 2026. doi:10.1016/j.commat.2026.114868.
84. Neha Rajput, Nidheesh Virakante, Hardik L. Kagdada, and Ankit Jain. Thermal transport properties of diamond and hard carbon allotrope using machine-learning driven atomistic simulations. *Computational Materials Science*, 268:114648, 2026. doi:10.1016/j.commat.2026.114648.
85. Guoliang Ren, Junwei Che, Hanchao Zhang, Huangyue Cai, Wenbo Li, Wei Hao, Qiaodan Hu, Xiaofeng Zhao, and Fan Yang. Tailoring the fracture toughness of fluorite-type rare-earth tantalates: from

- atomic scale fracture mechanism to bond-mediated toughening design. *Acta Materialia*, 304:121820, 2026. doi:10.1016/j.actamat.2025.121820.
86. Guoliang Ru, Kaiyuan Xue, Xiaoming Zong, Shusheng Xu, Sergey Chizhik, Xuqing Liu, Weihong Qi, and Weimin Liu. Tunable interfacial friction in two-dimensional ferroelectric In_2Se_3 heterostructures. *Tribology International*, 219:111818, 2026. doi:10.1016/j.triboint.2026.111818.
 87. Yongbo Shi, Yu Bao, Shuo Cao, Ye Su, Keke Song, Lu Xiao, and Ping Qian. Investigation of the transport behavior of BP and BAs using the boltzmann transport equation and machine learning methods. *Journal of Applied Physics*, 139(5):055701, 2026. doi:10.1063/5.0314057.
 88. Yongbo Shi, Yu Bao, Lei Zhang, Ye Su, and Ping Qian. Insights into the mechanical and thermal transport properties of 2D BxCyNz from machine-learned interatomic potentials. *Diamond and Related Materials*, 161:113096, 2026. doi:10.1016/j.diamond.2025.113096.
 89. Fei Shuang, Penghua Ying, Kai Liu, Zixiong Wei, Fengxian Liu, Zheyong Fan, Minqiang Jiang, and Poulumi Dey. Benchmarking chemically scalable machine-learning interatomic potentials for large-scale simulations of multicomponent alloys. *Phys. Rev. Mater.*, Jun 2026. doi:10.1103/1qc9-ypyb.
 90. S.F. Solodovnikov, M. Zeraati, A.B. Kuznetsov, E.S. Zolotova, A.R. Nasyrbaev, I.P. Gulyaev, I.K. Igumenov, R.A. Shutilov, E.A. Maksimovsky, D.P. Pishchur, V.N. Yudin, V.V. Lukashov, I.V. Korolkov, A.P. Maltsev, and A.R. Oganov. Thermal and mechanical properties of double perovskite-type Ba_2YNbO_6 ceramic. *Ceramics International*, 2026. doi:10.1016/j.ceramint.2026.02.315.
 91. Keke Song, Jiahui Liu, Yuanxu Zhu, Shunda Chen, Zheyong Fan, Yanjing Su, and Ping Qian. Solute segregation in polycrystalline aluminum from hybrid monte carlo and molecular dynamics simulations with a unified neuroevolution potential. *Materials Genome Engineering Advances*, pages e70049, 2026. doi:10.1002/mgea.70049.
 92. Yunfei Song, Zijun Qi, Zhanpeng Sun, Rui Li, Qijun Wang, Kang Liang, Wei Shen, Gai Wu, and Sheng Liu. Near-junction thermal management for GaN/diamond HEMTs in extreme environments based on multiscale thermal simulation. *International Communications in Heat and Mass Transfer*, 175:111225, 2026. doi:10.1016/j.icheatmasstransfer.2026.111225.
 93. R.S. Stroud, C. Reynolds, T. Melichar, J. Haley, M. Carter, M. Moody, C. Hardie, D. Bowden, D. Nguyen-Manh, and M.R. Wenman. Defects and impurity properties of VN precipitates in ARAFM steels: Modelling using a universal machine learning potential and experimental validation. *Journal of Nuclear Materials*, 624:156475, 2026. doi:10.1016/j.jnucmat.2026.156475.
 94. Achyut Subedi, Patrick E. Hopkins, and Ashutosh Giri. Topology-driven node-linker coupling enables exceptional thermal and mechanical performance in covalent organic frameworks. *ACS Nano*, 2026. doi:10.1021/acsnano.6c04527.
 95. Enjian Sun, Wenzhu Luo, Lei Wang, and Ershuai Yin. Effect of growth temperature on the quality and interfacial thermal conductance of GaN/AlN heterostructure: A molecular dynamics study. *Journal of Applied Physics*, 139(13):135102, 2026. doi:10.1063/5.0320914.
 96. Yifan Sun, Guoliang Chen, Guoliang Ren, Qingyuan Zhao, Yaming Wang, Fan Yang, Yongchun Zou, Shuqi Wang, Junming Zhao, Shikui Dong, Yong Shuai, Yu Zhou, and Jun Qiu. Machine learning-guided atomic-scale design of binary rare-earth niobates with ultra-low thermal conductivity. *Journal of Materials Science & Technology*, 2026. doi:10.1016/j.jmst.2026.06.015.
 97. Zhanpeng Sun, Zijun Qi, Yunfei Song, Lijie Li, Sheng Liu, Wei Shen, and Gai Wu. Multiscale investigation of thermal transport in β - Ga_2O_3 -based heterointerfaces enabled by machine learning potential: cross-scale parameter. *npj Computational Materials*, 2026. doi:10.1038/s41524-026-02007-y.
 98. Lucas Svensson, Babak Sadigh, Christine Wu, and Paul Erhart. Efficient method for calculation of low-temperature phase boundaries. *The Journal of Chemical Physics*, 165(2):024113, 2026. doi:10.1063/5.0333284.
 99. Daiji Tang, YuTao Liu, Han Song, Cheng Deng, Mengyuan Liu, TingHong Gao, Yongchao Liang, Qingquan Xiao, and Yunjun Ruan. Neuroevolution potential-driven accurate and efficient discovery of Graphene/GaN

- heterojunctions: From ballistic-diffusive transition to thermal conductivity enhancement. *Physica E: Low-dimensional Systems and Nanostructures*, 175:116363, 2026. doi:10.1016/j.physe.2025.116363.
100. Feng Tao, Xiaoliang Zhang, Dawei Tang, Shigeo Maruyama, and Ya Feng. Bose-Einstein phonon statistics explain drastic thermal conductivity reduction in SWCNT bundles. *Nano Letters*, 2026. doi:10.1021/acs.nanolett.5c06521.
 101. Qikun Tian, Ailing Chen, Haofeng Qin, Ruyi Li, Yi Zhang, Yuting Jiang, Xiong Zheng, Zhenzhen Qin, and Guangzhao Qin. Atomic-level engineering anisotropic thermal transport for directional heat dissipation in silicon electronics. *Materials Today Physics*, 62:102049, 2026. doi:10.1016/j.mtphys.2026.102049.
 102. Bikash Timalisina, Huy Gia Nguyen, and Keivan Esfarjani. Effect of stoichiometry on thermodynamic and thermal transport properties of entropy-stabilized oxide MgCoNiCuZnO_5 . *Journal of Applied Physics*, 139(2):025105, 01 2026. doi:10.1063/5.0303178.
 103. Hua Tong, Hui-Ling Kuang, Cheng-Wei Wu, Yu-Jia Zeng, Xue-Kun Chen, and Wu-Xing Zhou. Al-doping effects on β - Ga_2O_3 thermal transport: Neural network potential-based NEMD. *Physics Letters A*, pages 131713, 2026. doi:10.1016/j.physleta.2026.131713.
 104. Feng Wang, Tao Hu, Xiaodan Yu, Xiaoxin Zhang, and Zhongming Ren. Atomic-scale substitutional defect engineering for significant enhancement of thermal transport at gan/diamond interfaces. *Carbon*, 248:121221, 2026. doi:10.1016/j.carbon.2025.121221.
 105. Junjie Wang, Haoting Zhang, Shuning Pan, Qiuhan Jia, Chi Ding, Michel M. Lukanov, Faridun Jalolov, Alexander G. Kvashnin, Zheyong Fan, and Jian Sun. NEPMaker: Active learning of neuroevolution machine learning potential for large cells. *Chinese Physics B*, 2026. doi:10.1088/1674-1056/ae85f8.
 106. Weitao Wang, Xin Wu, Yunhui Wu, Sebastian Volz, Takeshi Takagi, and Masahiro Nomura. Interfacial thermal transport analysis using machine learning potential at AlN/Cu van der waals interface. *Materials Today Physics*, 62:102057, 2026. doi:10.1016/j.mtphys.2026.102057.
 107. Yidan Wang, Jian He, Yang Wu, Xueqi Yin, Zhen Li, Shan Li, and Hongbo Guo. CMAS-induced corrosion behavior of high-entropy rare-earth disilicate at 1500 °C. *npj Materials Degradation*, 2026. doi:10.1038/s41529-026-00784-x.
 108. Yuanrui Wang, Sijin Zhou, and Ziliang Wang. Bridging accuracy and efficiency in solid-liquid interface simulations: An accelerated fine-tuning strategy for MgO hydration. *Computational Materials Science*, 269:114726, 2026. doi:10.1016/j.commatsci.2026.114726.
 109. Zhao Wang and Zhenfu Tian. Active learning of anharmonic phonon transport across carbon allotropes. *The Journal of Physical Chemistry Letters*, 2026. doi:10.1021/acs.jpclett.5c03892.
 110. Zhaoming Wang, Guanghui Shi, Guanghui Wang, Man Wang, Feng Ding, and Xiao Wang. Unveiling sodium storage mechanisms in hard carbon via machine learning-driven simulations integrated with accurate site occupation identification. *Chem. Sci.*, 17:2547–2558, 2026. doi:10.1039/D5SC07068F.
 111. Zhenduo Wang, Junwen Duan, Xuecheng Shao, and Bo Gao. Shear-coupled migration governed by misorientation angles and atomistic structures in [1 1 0] symmetric tilt grain boundaries of face-centered-cubic Pt. *Acta Materialia*, 315:122408, 2026. doi:10.1016/j.actamat.2026.122408.
 112. Zhepeng Wang, Bing Wang, Penghua Ying, and Jin Zhang. Machine learning molecular dynamics simulations on piezopotentials of hexagonal boron nitride/graphene lateral heterostructures: Implications for flexible piezoelectric devices. *ACS Applied Nano Materials*, 2026. doi:10.1021/acsanm.5c05431.
 113. Ziyang Wang, Bingqi Linghui, Donghao Li, Jie Zhu, and Dawei Tang. Systematic investigation of interfacial heat transport at Metal/GaN interfaces: Experimental and simulation validation. *International Journal of Heat and Mass Transfer*, 261:128564, 2026. doi:10.1016/j.ijheatmasstransfer.2026.128564.
 114. Sujing Wei, Xueliang Wang, and Tianyu Shu. Decoupled influence of chemical compositions and nanotopographies on the hydrophilicity of nanotextured Ti-6Al-4V implants: Insights from molecular dynamics simulations. *Journal of Colloid and Interface Science*, 704:139464, 2026. doi:10.1016/j.jcis.2025.139464.

115. Ya Wei, Mingjiang Jin, Xinyu Bie, and Xiaodong Wang. Effect of filler wettability of Cu and CuGa₂ on thermal conductivity in liquid metal matrix composites. *International Journal of Thermal Sciences*, 228:110978, 2026. doi:10.1016/j.ijthermalsci.2026.110978.
116. Zixiong Wei, Fei Shuang, and Poulumi Dey. From O adsorption to Fe oxide growth: Benchmarking reactive force fields and universal machine learning interatomic potentials against DFT for BCC Fe surface oxidation. *Surface and Coatings Technology*, 521:133092, 2026. doi:10.1016/j.surfcoat.2025.133092.
117. Ning Wu, Zhe Wu, Rui Yang, Dian Huang, Guihua Tang, and Zhigang Liu. Atomic-scale bonding configurations and phonon-bridge-mediated interfacial thermal transport in GaN/SiC. *Phys. Rev. B*, 113:195308, May 2026. doi:10.1103/zkfr-mbww.
118. Yunchao Wu, Lichuan Zhang, Yuanping Chen, Yuee Xie, and Xiaohong Yan. Evolution of boron nitride structures from hexagonal to cubic and wurtzite phases: A machine-learning potential approach. *Applied Physics Letters*, 128(15):151908, Apr 2026. doi:10.1063/5.0326291.
119. Tian Xia, Hui Liu, Chun-Hua Long, and Hai-Bo Yi. Hydroxylation mechanism of hydroxyapatite nucleation with aspartate assistance under confinement revealed by machine learning molecular dynamics. *The Journal of Physical Chemistry B*, 2026. doi:10.1021/acs.jpcc.6c00279.
120. Kaibin Xiong, Yuan Li, Ziyang Lin, Gaoyang Luo, and Jianyang Wu. Phonon bottleneck in methane clathrate hydrate from machine learning molecular dynamics. *Chemical Engineering Science*, 321:122955, 2026. doi:10.1016/j.ces.2025.122955.
121. Kaibin Xiong, Yuan Li, Ziyang Lin, Gaoyang Luo, and Jianyang Wu. Phonon-mediated heat transport in CO₂ hydrate. *The Journal of Physical Chemistry A*, 2026. doi:10.1021/acs.jpca.5c06498.
122. Ao Xu, Kaiqi Li, Jian Zhou, and Zhimei Sun. Temperature dependence of thermal conductivity in amorphous Sb₂Te₃. *Scripta Materialia*, pages 117365, 2026. doi:10.1016/j.scriptamat.2026.117365.
123. Ruihan Xu, Boxiang Gao, Danlei Zhao, Jingzhuo Zhou, Qi Zhu, Yupeng Ma, Binzhao Li, Yi Zhang, Juzheng Chen, Qian Zhang, Fanling Meng, Johnny C. Ho, Maolin Yu, and Yang Lu. Activating phase-transition toughening in van der Waals semiconductor GaTe. *Nano Letters*, 2026. doi:10.1021/acs.nanolett.6c01935.
124. Xiao Xu, Shijie Wang, Haifeng Qin, Zhiqiang Zhao, Zheyong Fan, Zhuhua Zhang, and Hang Yin. A high-efficiency neuroevolution potential for tobermorite and calcium silicate hydrate systems with ab initio accuracy. *Cement and Concrete Research*, 200:108091, 2026. doi:10.1016/j.cemconres.2025.108091.
125. Yixin Xu, Xing Xiang, Zhigang Li, Yanglong Lu, and Yanguang Zhou. Fast ionic transport governed by collective vibrational dynamics. *Phys. Rev. Lett.*, 136:126302, 2026. doi:10.1103/hq41-8bt5.
126. Zihan Yan, Zheyong Fan, and Yizhou Zhu. Improving robustness and training efficiency of machine-learned potentials by incorporating short-range empirical potentials. *Journal of Chemical Information and Modeling*, 66(3):1406–1413, 2026. doi:10.1021/acs.jcim.5c02335.
127. Zihan Yan, Denan Li, Xin Wu, Zhoulun Liu, Chen Hua, Boyi Situ, Hao Yang, Shengjie Tang, Benrui Tang, Ziyang Wang, Shangzhao Yi, Huan Wang, Dian Huang, Ke Li, Qilin Guo, Zherui Chen, Ke Xu, Yanzhou Wang, Ziliang Wang, Gang Tang, Shi Liu, Zheyong Fan, and Yizhou Zhu. Gpumdkit: A user-friendly toolkit for GPUMD and NEP. *Materials Genome Engineering Advances*, pages e70074, 2026. doi:10.1002/mgea.70074.
128. Fangwei Yang, Haoran Sun, Xiaoxin Yang, Xu Li, and Gang Yang. A neuroevolution potential for predicting the lattice thermal conductivity of structurally disordered γ -Ga₂O₃. *Computational Materials Science*, 266:114555, 2026. doi:10.1016/j.commatsci.2026.114555.
129. Jianmin Yang, Lin Xie, Mingyuan Hu, Yingpeng Qi, Jun Li, Baohai Jia, Jianghe Feng, Zijiang Chen, Michel Bosman, Dao Xiang, and Jiaqing He. Tracking ultrafast ion diffusion dynamics in AgCrSe₂ superionic conductor. *Phys. Rev. X*, 16:021024, May 2026. doi:10.1103/s6rh-7219.
130. Linpo Yang, Jia Wei, and Yinglin Song. Insight into the temperature-dependent lattice thermal properties of CeO₂ stabilized ZrO₂ by machine learning force field. *Phys. Chem. Chem. Phys.*, 2026. doi:10.1039/D5CP03311J.

131. Ning Yang, Jian Zhou, Hongfu Huang, and Zhimei Sun. Machine learning-assisted high-entropy alloy discovery: a perspective. *Journal of Materials Informatics*, 2026. doi:10.20517/jmi.2025.79.
132. Ningxi Yang, Jincheng Yue, Xinkai Sun, Yinong Liu, Meng An, and Shiqian Hu. Anomalous thermal conductivity enhancement in twisted graphene/h-BN bilayers revealed by neuroevolution potential driven atomistic simulations. *Phys. Rev. B*, 113:035426, Jan 2026. doi:10.1103/m5wt-1hx4.
133. Xinyi Yang, Bensheng Huang, Xiaowei Wang, Yongyou Zhu, and Jianneng Zheng. Study on the microstructure and wear resistance of CoCrFeMnNiMoY_x high entropy alloys based on molecular dynamics simulation. *Tribology International*, 213:111084, 2026. doi:10.1016/j.triboint.2025.111084.
134. Ziyang Yang, Yijie Zhu, Jiuyang Shi, Shuning Pan, Shaobo Yu, Yujian Pan, Zhixin Liang, Junjie Wang, and Jian Sun. Phase diagram and global structure search of bismuth using machine learning potential. *The Journal of Chemical Physics*, 164(11):114503, 2026. doi:10.1063/5.0308764.
135. Fei Yin, Shixian Liu, Yiming Dong, A. A. Barinov, Ke Xu, and V. I. Khvesyuk. Accelerated phonon transport calculations for nanostructures: Combining neuroevolution potentials and compressed sensing. *Journal of Applied Physics*, 139(13):135103, 2026. doi:10.1063/5.0324012.
136. Yuan Yin, Zhetong Liu, Ziyue Xu, Qingxu Su, Szuyu Huang, Jiaxin Liu, Zhenzhong Wang, and Peng Gao. Direct bonding of 6-inch GaAs/SiC wafers for enhanced thermal management in high-power devices. *ACS Applied Materials & Interfaces*, 2026. doi:10.1021/acsami.5c21396.
137. Zhiyong You, Shuaishuai Jin, Peide Han, Xiaofeng Niu, and Hang Li. Understanding the pressure-induced structural evolution and thermophysical response of Mg-Zn intermetallics via a neuroevolution potential. *Journal of Materials Research and Technology*, 43:1002–1012, 2026. doi:10.1016/j.jmrt.2026.06.105.
138. Maolin Yu, Yangming Gui, Zhiqiang Zhao, Jidong Li, Xu Guo, and Zhuhua Zhang. Mechanism for borophene phase transition on substrate. *ACS Nano*, 2026. doi:10.1021/acsnano.5c17811.
139. Maolin Yu, Bowen Wang, Zhiqiang Zhao, Jingtong Zhang, Xu Guo, and Zhuhua Zhang. Moiré-induced toughening in twisted van der Waals bilayers. *Journal of the Mechanics and Physics of Solids*, 215:106735, 2026. doi:10.1016/j.jmps.2026.106735.
140. Ru Yu, Feng Xiao, Qing-Yu Xie, Wen Lei, Huazhong Guo, Bao-Tian Wang, and Xing Ming. Additional phonon scattering channels provided by one-dimensional heavy double-chain rattlers in quasi-one-dimensional perovskite δ -CsSnI₃. *Phys. Rev. B*, 113:184311, May 2026. doi:10.1103/873p-hdjm.
141. Shihong Yu, Peng Bi, Jingsong Liu, Enzhu Li, and Yong Yi. Machine learning molecular dynamics investigation on cation diffusion mechanisms in Ni_{0.2}Cu_{0.3}Zn_{0.5}Fe₂O₄ ferrite and its homogeneous interfaces. *Solid State Ionics*, 439:117164, 2026. doi:10.1016/j.ssi.2026.117164.
142. Wei-Zhe Yuan, Yangyu Guo, and Hong-Liang Yi. Atomistic effect on infrared dielectric response of GaN/AlN superlattices. *Applied Physics Reviews*, 13(2):021413, May 2026. doi:10.1063/5.0316648.
143. Wei-Zhe Yuan, Yangyu Guo, and Hong-Liang Yi. Dynamic charges effect on infrared dielectric response of polar materials. *Journal of Applied Physics*, 139(22):223104, 06 2026. doi:10.1063/5.0337379.
144. Jincheng Yue, Rongkun Chen, Xinkai Sun, Yinong Liu, Xiaolong Li, and Shiqian Hu. Phonon overdamping defines the ultimate limit of lattice thermal conductivity. *Phys. Rev. B*, 113:L161401, 2026. doi:10.1103/jhfd-vzjy.
145. Zifu Zang, Yushun Zhao, Chao Sui, Shaodong Sun, Kaiyi Zheng, Renjie Ding, Yongqiang Tao, Xiaodong He, and Chao Wang. Effects of phase transitions and oxygen vacancies on thermal transport in yttrium-stabilized zirconia studied using neural network potentials. *Phys. Rev. B*, 113:174117, May 2026. doi:10.1103/ntwm-4gf7.
146. Xin-Yue Zhai, Cheng-Wei Wu, Hua Tong, Yu-Jia Zeng, Yan Guo, and Wu-Xing Zhou. Role of interfacial bonding and phonon spectral overlap in thermal transport across GaN/c-BN interfaces. *Chinese Physics B*, 2026. doi:10.1088/1674-1056/ae7e71.

147. Bohan Zhang, Biyuan Liu, Penghua Ying, Zherui Chen, Yanzhou Wang, Yonglin Zhang, Haikuan Dong, Jinglei Yang, and Zheyong Fan. Thermal conductivities of monolayer graphene oxide from machine learning molecular dynamics simulations. *The Journal of Chemical Physics*, 164(15):154703, Apr 2026. doi:10.1063/5.0319735.
148. Chao Zhang, Bo Xu, Jing Wan, Jian Liang, Yongshen Wu, and Cuixia Wang. Low-frequency in-plane phonons dominate thermal transport in montmorillonite: machine-learning molecular dynamics insights. *Langmuir*, 2026. doi:10.1021/acs.langmuir.5c05103.
149. Duohao Zhang, Xinlu Cheng, and Hong Zhang. Unraveling thermal transport mechanisms in monolayer CrSi₂N₄ using machine-learned potentials. *Phys. Chem. Chem. Phys.*, 2026. doi:10.1039/D6CP01105E.
150. Guangwu Zhang, Xue Cheng, Chao Yang, Cheng Shao, and Xinyu Wang. Thermal transport in Ti₃C₂ MX-ene enhanced by hydrogen functionalization-induced symmetry reconstruction. *Journal of Applied Physics*, 139(6):065103, Feb 2026. doi:10.1063/5.0310379.
151. Guangwu Zhang, Xing Xiang, Ziyan Qian, Yixin Xu, Shengying Yue, Hyejin Jang, Lin Yang, Yanguang Zhou, Xinyu Wang, and Qiye Zheng. Strain-gradient-driven decoupling of thermal suppression from anisotropy in β -Ga₂O₃. *Acta Materialia*, 307:121973, 2026. doi:10.1016/j.actamat.2026.121973.
152. Guangzheng Zhang, Shilin Dong, Lin Guo, Chenghao Li, Xinyu Wang, and Gongming Xin. Interfacial thermal transport investigation of β -Ga₂O₃/4H-SiC power devices via active-learning-driven neuroevolution potential. *International Journal of Heat and Mass Transfer*, 263:128710, 2026. doi:10.1016/j.ijheatmasstransfer.2026.128710.
153. Honggang Zhang, Xiaoyi Chen, Yangshun Lan, Changchun Ding, Dingbo Zhang, Yuxiang Ni, and Hua Bao. Vibrational and diffusional thermal transport in amorphous and deeply supercooled liquid alumina. *Phys. Rev. B*, 113:224318, Jun 2026. doi:10.1103/kfp9-rdth.
154. Jiajun Zhang, Jiajun Wang, Yu Yan, Fei Wang, Jian Wang, Jianjiang Zhao, Yunmin Chen, Yang Ju, and Hua Wei. Boosting strength and ductility of Ti-6Al-4V alloy via millisecond-level electric treatment induced heterogeneous nanostructures. *Acta Materialia*, 307:121976, 2026. doi:10.1016/j.actamat.2026.121976.
155. Jian Zhang, Yujin Zhang, Zhuo Zhao, Meng An, and Gang Zhang. Physical origin of high lattice thermal conductivity in β -Si₃N₄: A machine learning potential study. *Review of Materials Research*, 2(2):100164, 2026. doi:10.1016/j.revmat.2026.100164.
156. Jian Zhang, Zhuo Zhao, Dingbo Zhang, Haifei Zhan, and Gang Zhang. Robust mechanical stability and strain-insensitive phonon transport in helical quasi-one-dimensional GaSeI nanochains. *Phys. Rev. Mater.*, 10:024604, Feb 2026. doi:10.1103/b852-6l6m.
157. Kai Zhang, Jinyuan Xu, Ziyue Zhou, Yangyu Guo, and Hong-Liang Yi. How does collective hydrodynamic phonon transport impact interfacial thermal conductance. *Materials Today Physics*, 65:102127, 2026. doi:10.1016/j.mtphys.2026.102127.
158. Leiming Zhang, Yueheng Du, Haotian Sun, and Mingwen Zhao. Machine learning study on electronic structure and plasmonic response of graphene/hBN moiré heterostructures. *Phys. Rev. B*, 113:205414, May 2026. doi:10.1103/1ngq-zht8.
159. Pingyang Zhang, Zhoulin Liu, Xujiang Wang, Renjie Mi, Yingping Pang, Fengzijun Pan, Jingwei Li, Ziliang Wang, Wenlong Wang, and Kai Hong Luo. Unraveling the early-stage hydration mechanism of tricalcium silicate via machine learning potentials. *Chemical Engineering Journal*, 538:176856, 2026. doi:10.1016/j.cej.2026.176856.
160. Wei Zhang, Tinghong Gao, Yutao Liu, Guofa Shen, Bei Wang, Zhongzhong Zhu, Jin Huang, Shipeng Zhang, and Shuang Li. Thermal conductivity modulation in carbon nanotubes via silicon nanowire encapsulation investigated using neuroevolution potential. *Advanced Theory and Simulations*, pages e01834, 2026. doi:10.1002/adts.202501834.
161. Wentao Zhang, Xingxing Wu, Chen Wang, Siyu Hu, Yueyang Liu, and Lin-Wang Wang. Constructing machine learning interatomic potentials with minimum amount of ab initio data. *npj Computational Materials*, 2026. doi:10.1038/s41524-026-02023-y.

162. Xiaoying Zhang, Ning Liu, Qian Guo, Guoyong Shi, and Yue Wang. Beyond boltzmann transport: Green-Kubo prediction of lattice thermal conductivity with machine-learned potentials. *The Journal of Chemical Physics*, 164(7):074101, Feb 2026. doi:10.1063/5.0297462.
163. Xinxu Zhang, Xinran Zhou, Guo Li, Jiahao Wei, Qi Zhao, and Yonghui Li. Size-controlled electronic structure tuning in Ru-CrO_x heteronanoclusters. *The Journal of Physical Chemistry A*, 2026. doi:10.1021/acs.jpca.5c08571.
164. Xuanguang Zhang, Kaiqi Li, Jian Zhou, and Zhimei Sun. Atomistic modeling of thermal-processing rejuvenation in GeSe₂ glass with machine learning potentials. *Computational Materials Science*, 268:114674, 2026. doi:10.1016/j.commatsci.2026.114674.
165. Yuwen Zhang, Tao Ouyang, Zheyong Fan, and Wu Li. Microscopic mechanisms of low lattice thermal conductivity in Al_xGa_{1-x}N alloys revealed by neuroevolution machine-learning potential. *PRX Energy*, 5:023013, Jun 2026. doi:10.1103/df1n-9bll.
166. Yuwen Zhang, Zhipeng Tang, Tao Ouyang, and Wu Li. Achieving optimal GaN/SiC interfacial thermal conductance via ultrathin alloy interlayers for high-power device cooling. *npj Computational Materials*, 2026. doi:10.1038/s41524-026-02134-6.
167. Zhiqiang Zhao, Siyao Shuang, Kepeng Ouyang, Maolin Yu, Junping Du, Liangli Chu, Xiaokai Chen, Shigenobu Ogata, Wanlin Guo, Zhuhua Zhang, and Yong-Wei Zhang. Dual role of Nb in defect-mediated strength and ductility of γ -TiAl alloys. *Acta Materialia*, 313:122290, 2026. doi:10.1016/j.actamat.2026.122290.
168. JunYing Zhong, Lei Zhang, and Tao Bo. Active learning-enhanced neuroevolution potential for predictive modeling of uo2 thermophysical properties. *Phys. Chem. Chem. Phys.*, 2026. doi:10.1039/D5CP03760C.
169. Xianteng Zhou, Chaokun Guo, Zhen Yang, Yuanji Xu, Hongquan Song, De-Ye Lin, and Fuyang Tian. Vacancy-induced mechanism on deformation and thermal conductivity in medium-entropy carbides with typical grain boundaries. *Materials Today Physics*, 61:102025, 2026. doi:10.1016/j.mtphys.2026.102025.
170. Xinran Zhou, Jing Shi, Jiahui Li, Bowei Sha, Guo Li, Qi Zhao, Jiahao Wei, Xinxu Zhang, Yonghui Li, and Xiaodong Zhang. Opening extra transport channels by atomically precise doping in gold nanoclusters with electronic structure modulation. *The Journal of Physical Chemistry A*, 2026. doi:10.1021/acs.jpca.5c08708.
171. Zhewen Zhu and Yizhou Zhu. Regulating surface faceting as a kinetic switch for core-shell nanoparticle crystallization pathways. *ACS Nano*, 2026. doi:10.1021/acsnano.6c00369.

9.2 2025

1. B. Ruşen Argun and Antonia Statt. Machine-learning potentials for efficient simulations of anisotropic colloids. *The Journal of Chemical Physics*, 163(23):234120, 2025. doi:10.1063/5.0303706.
2. 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*, 11(11):2400944, 2025. doi:10.1002/aelm.202400944.
3. Esmée Berger, Erik Fransson, Fredrik Eriksson, Eric Lindgren, Göran Wahnström, Thomas Holm Rod, and Paul Erhart. Dynasor 2: From simulation to experiment through correlation functions. *Computer Physics Communications*, 316:109759, 2025. doi:10.1016/j.cpc.2025.109759.
4. 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.
5. William Bro-Jørgensen, Joseph M. Hamill, Davide Donadio, and Gemma C. Solomon. Bridging the gap: using machine learning force fields to simulate gold break junctions at pulling speeds closer to experiments. *ACS Nano*, 19(46):39735–39746, 2025. doi:10.1021/acsnano.5c11887.

6. 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*, 58:101854, 2025. doi:10.1016/j.mtphys.2025.101854.
7. 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.
8. 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*, 16:8806–8814, 2025. doi:10.1021/acs.jpcclett.5c01616.
9. 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.
10. Bingkun Chen, Like Cao, Qijun Wang, Chunmin Cheng, Zijun Qi, Zhanpeng Sun, Lijie Li, Gai Wu, and Wei Shen. Predicting thermal conductivity of InP via molecular dynamics simulations with machine learning potential. *Journal of Physics D: Applied Physics*, 58(48):485105, Nov 2025. doi:10.1088/1361-6463/ae1c4a.
11. Chengbing Chen, Yutong Li, Rui Zhao, Zhoulin Liu, Zheyong Fan, Gang Tang, and Zhiyong Wang. Neptrain and NepTrainKit: Automated active learning and visualization toolkit for neuroevolution potentials. *Computer Physics Communications*, 317:109859, 2025. doi:10.1016/j.cpc.2025.109859.
12. Fei Chen, Han Wang, Yanan Jiang, Lihua Zhan, and Youliang Yang. Development of a neuroevolution machine learning potential of al-cu-li alloys. *Metals*, 15(1):48, 2025. doi:10.3390/met15010048.
13. Hai-he Chen, Wen-qian Chen, Wen-zhao Ren, Hong-zhou Song, and Yong Lu. Temperature-dependent structural, thermodynamic, and phonon properties of lithium fluoride from a neuro-evolution machine-learning potential. *Journal of Advanced Thermal Science Research*, 12:95–104, Dec 2025. doi:10.15377/2409-5826.2025.12.7.
14. Ming Chen, Linfu Zhang, Kang Liu, Qiang Zhu, Jie Xu, Bin Guo, Debin Shan, Hushan Li, Guohua Fan, Tao Yang, Yinghao Zhou, and Peng Zhang. Atomic-scale insights into the exceptional pinning effect of the k bubbles on w grain boundary. *Acta Materialia*, 305:121845, 2025. doi:10.1016/j.actamat.2025.121845.
15. 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.
16. Guanjian Cheng and Wan-Jian Yin. De novo inverse design superhard C-N compounds via global machine learning interatomic potentials and multiobjective optimization algorithm. *The Journal of Physical Chemistry Letters*, 16(18):4392–4400, 2025. doi:10.1021/acs.jpcclett.5c00181.
17. Mo Cheng, Xuanyu Jiang, Haoming Zhang, Xiaodong Pi, Deren Yang, and Tianqi Deng. Revealing phonon signature of dislocations in silicon carbide using machine-learning interatomic potential. *Applied Physics Letters*, 127(24):242102, 12 2025. doi:10.1063/5.0298852.
18. Ruihuan Cheng, Chen Wang, Niuchang Ouyang, Christophe Candolfi, Emmanuel Guilmeau, Xingchen Shen, and Yue Chen. Strong crystalline thermal insulation induced by extended antibonding states. *Nature Communications*, 16(1):7941, 2025. doi:10.1038/s41467-025-63300-w.
19. 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.
20. 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.

21. 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.
22. Peng-Hu Du, Qian Wang, Qiang Sun, and Puru Jena. Thermal rectification of sierpiński tetrahedron fractals assembled from supertetrahedral T₂-type tin selenide clusters. *Phys. Rev. B*, 111:L121406, Mar 2025. doi:10.1103/PhysRevB.111.L121406.
23. Sangita Dutta, Erik Fransson, Tobias Hainer, Benjamin M. Gallant, Dominik J. Kubicki, Paul Erhart, and Julia Wiktor. Revealing the low-temperature phase of FAPbI₃ using a machine-learned potential. *Journal of the American Chemical Society*, 147:37019–37029, 2025. doi:10.1021/jacs.5c05265.
24. 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*, 11(1):215, 2025. doi:10.1038/s41524-025-01705-3.
25. H. F. Feng, B. Liu, Xin-Gao Gong, and Zhi-Xin Guo. Exceptional four-phonon scattering effects on low-temperature thermal conductivity in two-dimensional materials. *Phys. Rev. B*, 112:134306, Oct 2025. doi:10.1103/7w33-k7rz.
26. Kaiwei Feng, Guanjian Cheng, and Wan-Jian Yin. Searching superhard C and BxCyNz via full-space inverse design approach. *Phys. Rev. Mater.*, 9:123603, 2025. doi:10.1103/8qyx-sqtn.
27. 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.
28. 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*, 216:113751, 2025. doi:10.1016/j.tws.2025.113751.
29. Zhongtang Gao, Yilin Yuan, Teng Cao, Jiahao Yang, Zhiming Gao, and Yuan Yu. Melt structure of NiCrW superalloy near liquidus studied by experiment and molecular dynamics simulation. *Materials Letters*, 389:138405, 2025. doi:10.1016/j.matlet.2025.138405.
30. Zeyu Guo, Guanting Li, Mingyue Lv, and Jing Wan. Phonon thermal transport in diamond-graphene superlattices. *Diamond and Related Materials*, 159:112880, 2025. doi:10.1016/j.diamond.2025.112880.
31. Tobias Hainer, Erik Fransson, Sangita Dutta, Julia Wiktor, and Paul Erhart. A morphotropic phase boundary in MA_{1-x}FA_xPbI₃: linking structure, dynamics, and electronic properties. *Nature Communications*, 16:8775, 2025. doi:10.1038/s41467-025-64526-4.
32. 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, Jul 2025. doi:10.1063/5.0275050.
33. Song Hu and Xiaokun Gu. Approaching to low thermal conductivity limit in layered materials through full-spectrum phonon band engineering. *Materials Today Physics*, 52:101669, 2025. doi:10.1016/j.mtphys.2025.101669.
34. 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.
35. Jiapeng Huang and Chao Wu. An accurate two-body interaction model for describing the structure-energy relationship of binary alloys. *Journal of Chemical Theory and Computation*, 21(21):11293–11306, 2025. doi:10.1021/acs.jctc.5c00951.

36. Xiaoyu Huang, Sebastian Volz, Masahiro Nomura, and Yuxiang Ni. Machine learning-guided broadband phonon blockade via Anderson localization in engineered Si/Ge nanowires. *Phys. Rev. B*, 111:174203, May 2025. doi:10.1103/PhysRevB.111.174203.
37. 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.
38. Wenwu Jiang, Ting Liang, Jianbin Xu, and Wengen Ouyang. Strain-engineered anisotropic thermal transport in layered MoS₂ structures. *ACS Appl. Mater. Interfaces*, 17(23):34833–34844, 2025. doi:10.1021/acsami.5c06264.
39. 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*, 34:096104, 2025. doi:10.1088/1674-1056/ae039a.
40. Prakriti Kayastha, Erik Fransson, Paul Erhart, and Lucy Whalley. Octahedral tilt-driven phase transitions in bazrs₃ chalcogenide perovskite. *The Journal of Physical Chemistry Letters*, 16(8):2064–2071, 2025. doi:10.1021/acs.jpclett.4c03517.
41. Tuğbey Kocabaş, Murat Keçeli, Tanju Gürel, Milorad V. Milošević, and Cem Sevik. Thermal conductivity limits of MoS₂ and MoSe₂: revisiting high-order anharmonic lattice dynamics with machine learning potentials. *Applied Physics Reviews*, 12(4):041424, Dec 2025. doi:10.1063/5.0300627.
42. Rasmus Lavén, Erik Fransson, Paul Erhart, Fanni Juranyi, Garrett E. Granroth, and Maths Karlsson. Unraveling the nature of vibrational dynamics in cspb₃ by inelastic neutron scattering and molecular dynamics simulations. *J. Phys. Chem. Lett.*, 16:4812–4818, 2025. doi:10.1021/acs.jpclett.5c00778.
43. Jianxin Li, Hongwei Zhang, and Tienchong Chang. Mechanical properties of twin graphene nanotube. *Science China Technological Sciences*, 68(6):1620205, 2025. doi:10.1007/s11431-024-2909-y.
44. Jingyu Li, Zheng Ma, Hao Wang, Lanwei Li, Jianbo Zhu, Huaican Chen, Yuanpeng Zhang, Zhuoyang Ti, Jiajun Zhong, Yuanguang Xia, Peng-Fei Liu, Yongsheng Zhang, and Wen Yin. Boosting thermoelectric properties of high-entropy chalcogenides through local structural distortion and tailored chemical bonding. *Journal of the American Chemical Society*, 147:41629–41638, 2025. doi:10.1021/jacs.5c12728.
45. 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.
46. Mingqian Li, Lifeng Wang, and Zhuoqun Zheng. Coarse-grained machine learning potential for mesoscale multilayered graphene. *npj Computational Materials*, 2025. doi:10.1038/s41524-025-01849-2.
47. 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.
48. 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.
49. Weikang Li, Xiaoyu Huang, and Yuxiang Ni. Synergistic effects of phonon anderson localization and resonance in Si/Ge superlattice nanowires: toward lower thermal conductivity. *The Journal of Physical Chemistry C*, 129(42):19093–19099, 2025. doi:10.1021/acs.jpcc.5c04546.
50. 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*, 34:096302, 2025. doi:10.1088/1674-1056/ade5a0.

51. 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*, 289:120942, 2025. doi:10.1016/j.actamat.2025.120942.
52. Youtian Li, Yangyu Guo, Haiping Lin, Shiyun Xiong, and Hongliang Yi. Molecular dynamics study of the impact of coordination number on the thermal conductivity of amorphous Ga_2O_3 . *Phys. Rev. B*, 112:184111, Nov 2025. doi:10.1103/d8lh-jf5y.
53. 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.
54. 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.
55. 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*, 19:32422–32431, 2025. doi:10.1021/acsnano.5c08363.
56. 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.
57. 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, Aug 2025. doi:10.1063/5.0278798.
58. Ting Liang, Ke Xu, Penghua Ying, Wenwu Jiang, Meng Han, Xin Wu, Wengen Ouyang, Yimin Yao, Xiaoliang Zeng, Zhenqiang Ye, Zheyong Fan, and Jianbin Xu. Probing the ideal limit of interfacial thermal conductance in two-dimensional van der Waals heterostructures. *npj Computational Materials*, 12(1):11, Dec 2025. doi:10.1038/s41524-025-01885-y.
59. Yujie Liao, Pengfei Suo, Changhao Wang, Jincang Zhang, and Yunsong Li. Advanced machine learning interatomic potential for accelerated atomistic simulations of lithiation dynamics in large-scale Si@C core-shell anodes. *ACS Applied Materials & Interfaces*, 18(1):2996–3006, 2025. doi:10.1021/acsaami.5c18142.
60. 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.
61. Eric Lindgren, Adam J. Jackson, Erik Fransson, Esmée Berger, Goran Škoro, Svemir Rudić, Rastislav Turanyi, Sanghamitra Mukhopadhyay, and Paul Erhart. Predicting neutron experiments from first principles: a workflow powered by machine learning. *J. Mater. Chem. A*, 13:25509–25520, 2025. doi:10.1039/D5TA03325J.
62. Fachen Liu, Ruilin Mao, Zhiqiang Liu, Jinlong Du, and Peng Gao. Probing phonon transport dynamics across an interface by electron microscopy. *Nature*, 642:941–946, 2025. doi:10.1038/s41586-025-09108-6.
63. Jiahui Liu, Jesper Byggmästar, Zheyong Fan, Bing Bai, Ping Qian, and Yanjing Su. Utilizing a machine-learned potential to explore enhanced radiation tolerance in the MoNbTaVW high-entropy alloy. *Journal of Nuclear Materials*, 616:156004, 2025. doi:10.1016/j.jnucmat.2025.156004.
64. Jie Liu and Lin Zhang. A physics-informed machine learning perspective to present the structures and properties of titanium matrixes and nanoclusters through atomic modeling. *Nanoscale*, 17:11482–11501, 2025. doi:10.1039/D4NR05156D.
65. 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.
66. Runkai Liu, Shu Lin, Jing Wan, Le Li, Guoqiang Zhang, Huasong Qin, and Yilun Liu. Recent advances in machine learning guided mechanical properties prediction and design of two-dimensional materials. *Thin-Walled Structures*, 213:113261, 2025. doi:10.1016/j.tws.2025.113261.

67. 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.
68. 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*, 12:2501157, 2025. doi:10.1002/advs.202501157.
69. 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.
70. 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.
71. Yutao Liu, Tinghong Gao, Qingquan Xiao, Yunjun Ruan, Qian Chen, Bei Wang, and Jin Huang. Temperature-density dependent hidden order of amorphous carbon. *Communications Physics*, 8(1):420, 2025. doi:10.1038/s42005-025-02320-w.
72. 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.
73. 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*, 21:2409092, 2025. doi:10.1002/sml.202409092.
74. 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.
75. Qing Lu, Zhongwei Zhang, Yijie Zhu, Chi Ding, Xiaomeng Wang, Junjie Wang, Yu Han, and Jian Sun. Diverse carbon units in high-pressure c-k system predicted from first-principles and machine-learning methods. *The Journal of Chemical Physics*, 163(24):244706, 12 2025. doi:10.1063/5.0310376.
76. 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.
77. 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*, 16:5429–5434, 2025. doi:10.1021/acs.jpclett.5c00740.
78. Kaiqiang Ma, Lan Zhang, and Huizhong Ma. Machine learning potentials decode the thermophysical properties and two-stage phase transition in LaNbO₄. *Ceramics International*, 51:46211–46226, 2025. doi:10.1016/j.ceramint.2025.07.328.
79. 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.
80. 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*, 129:4183–4191, Feb 2025. doi:10.1021/acs.jpcc.4c07490.
81. 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.

82. Fengzijun Pan, Guangyao Li, Jiaqiu Xu, Zuhao Wang, Zepeng Fan, Ai Zhang, and Dawei Wang. Cenep: a general-purpose machine learning interatomic potential for cementitious materials. *Chemical Engineering Journal*, 526:170639, 2025. doi:10.1016/j.cej.2025.170639.
83. 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*, 129:14829–14840, 2025. doi:10.1021/acs.jpcc.5c03759.
84. 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.
85. 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.
86. 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*, 258:114535, 2025. doi:10.1016/j.matdes.2025.114535.
87. 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*, 202:105237, 2025. doi:10.1016/j.mechmat.2024.105237.
88. Petter Rosander, Erik Fransson, Nicklas Österbacka, Paul Erhart, and Göran Wahnström. Untangling the raman spectra of cubic and tetragonal BaZrO₃. *Phys. Rev. B*, 111:064107, Feb 2025. doi:10.1103/PhysRevB.111.064107.
89. 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.
90. Kairolla S. Sekerbayev, Omid Farzadian, Azat Abdullaev, Yanwei Wang, and Zhandos N. Utegulov. Enhanced interfacial thermal conductance across si/defected sic interface. *Applied Physics Letters*, 127(26):262203, 12 2025. doi:10.1063/5.0304881.
91. 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.
92. 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*, 13:21812–21825, 2025. doi:10.1039/D5TC02960K.
93. Hang Shi, Xuancheng Li, Ting Liang, Zhibin Wen, LinLin Ren, Meng Han, Yimin Yao, Xiaoliang Zeng, Jianbin Xu, and Rong Sun. Inhomogeneous interfacial layer: a hidden cause of high thermal contact resistance in polymer-based thermal interface materials with spherical fillers. *ACS Applied Materials & Interfaces*, 2025. doi:10.1021/acsami.5c20168.
94. Yongbo Shi, Yu Bao, Shuo Cao, Ye Su, and Ping Qian. Design of transition metal dichalcogenides using general-purpose machine-learned interatomic potentials. *Chemistry of Materials*, 2025. doi:10.1021/acs.chemmater.5c02492.
95. Yongbo Shi, Yuanxu Zhu, Zihao Song, Ye Su, and Ping Qian. Formation of native defects in boron arsenide and their impact on thermal conductivity. *Phys. Rev. B*, 112:214110, Dec 2025. doi:10.1103/1zf8-glmg.
96. 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*, 129:13445–13456, 2025. doi:10.1021/acs.jpcc.5c02871.

97. Jie Sun, Yiheng Shen, Tengfei Cao, and Li-Min Liu. Sliding and superlubric moiré twisting ferroelectric transition in HfO_2 . *npj Computational Materials*, 11(1):367, 2025. doi:10.1038/s41524-025-01846-5.
98. 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.
99. Zhanpeng Sun, Hutao Shi, Yilong Zhu, Rui Li, Xiang Sun, Qijun Wang, Zijun Qi, Lijie Li, Sheng Liu, Wei Shen, and Gai Wu. A comprehensive exploration of thermal transport at cu/diamond interfaces via machine learning potentials. *npj Computational Materials*, 11:359, Nov 2025. doi:10.1016/j.diamond.2024.111303.
100. 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.
101. 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*, 172:116259, 2025. doi:10.1016/j.physe.2025.116259.
102. 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*, 27:20334–20343, 2025. doi:10.1039/D5CP02310F.
103. Zhunyun Tang, Xiaoxia Wang, Jin Li, Chaoyu He, Mingxing Chen, Chao Tang, and Tao Ouyang. Ultralow lattice thermal conductivity induced by unique hierarchical rotational dynamics in lightweight cyanide-bridged framework materials. *Advanced Functional Materials*, pages e26344, 2025. doi:10.1002/adfm.202526344.
104. 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.
105. 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.
106. 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*, 389(6767):1342–1346, 2025. doi:10.1126/science.adu0719.
107. Bing Wang, Kaiqi Li, Weiming Zhang, Yuqi Sun, Jian Zhou, and Zhimei Sun. Thermal transport of $\text{GeTe/Sb}_2\text{Te}_3$ 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.
108. 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.
109. 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.
110. 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.
111. 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.
112. 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.

113. 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.
114. 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.
115. 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.
116. 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*, 8:100039, 2025. doi:10.1016/j.commt.2025.100039.
117. 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.
118. Zhihao Wang, Wentao Li, Siying Wang, and Xiaonan Wang. The future of catalysis: Applying graph neural networks for intelligent catalyst design. *WIREs Computational Molecular Science*, 15(2):e70010, 2025. doi:10.1002/wcms.70010.
119. 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*, 301:121529, 2025. doi:10.1016/j.actamat.2025.121529.
120. Yue Wen, Jeongho Cho, Linfeng Yu, Yi-Ming Zhao, Jingxuan Wang, Lei Shen, Guangzhao Qin, Seok Min Yoon, Jongwoo Lim, and Sunmi Shin. Thermal conductivity of individual single-crystalline F₁₆CuPc nanoribbons. *Nanoscale*, 17:20982–20988, 2025. doi:10.1039/D5NR02286J.
121. 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.
122. 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.
123. 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, Wenhui 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*, 12:e03388, 2025. doi:10.1002/advs.202503388.
124. Xing Xiang and Yanguang Zhou. Comparable thermal transport between Li₁₅Si₄ and lisi. *Phys. Rev. Mater.*, 9:085401, Aug 2025. doi:10.1103/ndlz-hxwq.
125. 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 AgSnSbTe₃ crystals. *Phys. Rev. B*, 111:184304, May 2025. doi:10.1103/PhysRevB.111.184304.
126. 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.
127. 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.

128. 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*, 3:e70028, 2025. doi:10.1002/mgea.70028.
129. 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*, 16:8580–8587, 2025. doi:10.1021/acs.jpclett.5c01802.
130. 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.
131. Feng-ning Xue, Zhi-qiang Hu, Yong-Chao Wu, Wei-Dong Wu, Jian-Li Shao, and Pei Wang. Atomic-scale simulations of the interfacial mechanical properties of al/alcu intermetallic compounds. *Phys. Chem. Chem. Phys.*, 27:20901–20914, 2025. doi:10.1039/D5CP02415C.
132. Xuyang Xue, Jian Yuan, Junyi Yang, and Huashan Li. Unraveling the rare ferroelectric-antiferroelectric transition in the two-dimensional hybrid perovskites (PMA)₂PbBr₄. *Phys. Rev. B*, 112:024106, Jul 2025. doi:10.1103/4211-fmkt.
133. 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*, 112:115422, Sep 2025. doi:10.1103/wj6y-bsjq.
134. 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.
135. 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.
136. 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.
137. 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.
138. 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.
139. 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.
140. Wei-Zhe Yuan, Yangyu Guo, and Hong-Liang Yi. Consistency between the Green-Kubo formula and the lorentz model for predicting the infrared dielectric function of polar materials. *Phys. Rev. B*, 111:184310, 2025. doi:10.1103/PhysRevB.111.184310.

141. 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*, 42:036301, 2025. doi:10.1088/0256-307X/42/3/036301.
142. 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.
143. 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 $\text{Cs}_3\text{Bi}_2\text{I}_6\text{Cl}_3$. *Proceedings of the National Academy of Sciences*, 122(41):e2415664122, 2025. doi:10.1073/pnas.2415664122.
144. 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*, 7:2695–2701, 2025. doi:10.1021/acsmaterialslett.5c00263.
145. 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.
146. 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*, 17(42):58080–58093, 2025. doi:10.1021/acsaami.5c11237.
147. Chenxin Zhang, Yanyan Chen, Qian Wang, and Puru Jena. Phonon localization and a boson-peak-like anomaly in twisted penta-pdse₂ bilayer. *Nano Lett.*, 25:8689–8695, 2025. doi:10.1021/acs.nanolett.5c01621.
148. Chenxin Zhang, Qian Wang, and Puru Jena. Twisting-induced phonon localization and ultralow thermal conductivity in penta-pdte₂ bilayer revealed by a universal machine-learning potential. *Small*, pages e09794, 2025. doi:10.1002/sml.202509794.
149. 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.
150. 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.
151. 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.*, 25:094603, Sep 2025. doi:10.1103/z9gl-yb41.
152. Jian Zhang, Zhuo Zhao, Yuan Zhang, Yongjun Huo, Aijun Hou, Haochun Zhang, and Gang Zhang. Machine learning potentials in studying phononic and thermal properties of germanium telluride. *AIP Advances*, 15(10):100702, 2025. doi:10.1063/5.0294385.
153. 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.
154. Lei Zhang, Fei Tian, Ke Chen, Zhongbo Yan, and Kun Cao. Uniaxial stress tuning of interfacial thermal conductance in cubic BAs/4H-SiC heterostructures. *Phys. Rev. Mater.*, 9:094604, 2025. doi:10.1103/12z6-4kj4.
155. 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.
156. 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.

157. Wei Zhang, Shiyun Xiong, Yangyu Guo, Hong-Liang Yi, Masahiro Nomura, and Sebastian Volz. Thermal conductivity enhancement by phonon inelastic scattering in $\text{Si}_{1-x}\text{Ge}_x$ alloyed nanowires. *Phys. Rev. B*, 112:195423, Nov 2025. doi:10.1103/jr1j-djfx.
158. Xiaoliang Zhang, Yanjun Xie, Feng Tao, Chenxi Sun, and Dawei Tang. Thermal transport in MoSi_2N_4 monolayer: A molecular dynamics study based on machine learning. *International Journal of Heat and Mass Transfer*, 250:127290, 2025. doi:10.1016/j.ijheatmasstransfer.2025.127290.
159. 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.
160. 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.
161. Wenjie Zhou, Changming Ke, and Shi Liu. Two-dimensional ferroelectric crystal with temperature-invariant ultralow thermal conductivity. *Phys. Rev. B*, 112:134307, 2025. doi:10.1103/3nqr-9b32.
162. 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.
163. 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.
164. 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.

9.3 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*, 8(9):4364–4372, 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 ca_3crn_3h . *Chemistry of Materials*, 37:489–496, 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, Çağıl 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 $\text{Mg}_3(\text{Sb}, \text{Bi})_2$: an atomistic view of concentration dependent segregation effects. *Materials Today Physics*, 43:101386, 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*, 12:3897–3906, 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*, 35:2418802, 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 Mekkath. Thermal characteristics of CsPbX_3 ($x = \text{cl}/\text{br}/\text{i}$) 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*, 11:1369034, 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-PdSe₂-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 -Ga₂O₃/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 -Ga₂O₃/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 ScF₃. *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*, 5:102042, 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. Timalisina, H. G. Nguyen, and K. Esfarjani. Neuroevolution machine learning potential to study high temperature deformation of entropy-stabilized oxide MgNiCoCuZnO₅. *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 C₆N₇ 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 -Ga₂O₃ and β -Ga₂O₃ 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 CaSiO₃ 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 Li₇La₃Zr₂O₁₂. *Chemistry of Materials*, 36:11551–11557, 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/acs.nano.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 BiI₃ 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.

9.4 2023

1. EA Bea, A Mancardo Viotti, MF Carusela, AG Monastera, 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 Osthuys, 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 ($\text{X}=\text{Cl}, \text{Br}, \text{I}$). *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 InGeX₃ (X = S, Se and Te) monolayers using density functional theory and machine learning. *Physical Chemistry Chemical Physics*, 25:13864–13876, 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 -Ga₂O₃. *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 SrTiO₃ 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/C₃N 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.

9.5 2022

1. Joakim Brorsson, Arsalan Hashemi, Zheyong Fan, Erik Fransson, Fredrik Eriksson, Tapio Ala-Nissila, Arkady V. Krashenninnikov, 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, Çağır Köroğlu, and Eric Pop. Substrate-dependence of monolayer MoS₂ 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, Masahiro 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.surf.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 sp²-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.

9.6 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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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.

9.7 2020

1. E. A. Bea, M. F. Carusela, A. Soba, A. G. Monastera, 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 MoS₂. *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.

9.8 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*, 151(23):234105, 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 MoS₂ from molecular dynamics using different empirical potentials. *Physical Review B*, 99(5):054303, 2019. doi:10.1103/PhysRevB.99.054303.

9.9 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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1088/1361-651X/aae180).

9.10 2017

1. Khaterreh 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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1039/C7NR05231F).

9.11 2016

1. Petri Hirvonen, Mikko M. Ervasti, Zheyong Fan, Morteza Jalalvand, Matthew Seymour, S. Mehdi Vaez Al-laei, 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](https://doi.org/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](https://doi.org/10.1016/j.carbon.2016.03.007).

9.12 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](https://doi.org/10.1103/PhysRevB.92.094301).

9.13 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](https://doi.org/10.1016/j.cpc.2013.01.008).

GLOSSARY

ACE	atomic cluster expansion [Drautz2019]
ADF	angular distribution function
ADP	angular-dependent potential [Mishin2005]
ARDF	angular-dependent radial distribution function
BDP	<i>Bussi-Donadio-Parrinello thermostat</i> [Bussi2007b]
BEC	Born effective charge
DOS	density of states
EAM	<i>embedded atom method</i>
EMD	equilibrium molecular dynamics
FCP	<i>force constant potential</i>
GK	<i>Green-Kubo</i>
GPU	graphics processing unit
GKMA	<i>Green-Kubo modal analysis</i> [Lv2016]
HAC	<i>heat current auto-correlation</i>
HNEMA	<i>homogeneous non-equilibrium modal analysis</i> [Gabourie2021]
HNEMD	<i>The homogeneous non-equilibrium molecular dynamics</i>

HNEMDEC

homogeneous non-equilibrium molecular dynamics Evans-Cummings algorithm

IC

iron conductivity

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

SNESseparable 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

VCSGCvariance-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).
- [Dammak2009] H. Dammak, Y. Chalopin, M. Laroche, M. Hayoun, and J.-J. Greffet, *Quantum Thermal Bath for Molecular Dynamics Simulation*, *Phys. Rev. Lett.* **103**, 190601 (2009).
- [Bernetti2020] Mattia Bernetti and Giovanni Bussi
Pressure control using stochastic cell rescaling
J. Chem. Phys. **153**, 114107 (2020)
DOI: [10.1063/5.0020514](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1063/1.2408420)

- [Ceriotti2010] Michele Ceriotti, Michele Parrinello, Thomas E. Markland, and David E. Manolopoulos
Efficient stochastic thermostating of path integral molecular dynamics
J. Chem. Phys. **133**, 124104 (2010)
DOI: [10.1063/1.3489925](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1063/5.0106617)
- [Fan2026] Zheyong Fan, Benrui Tang, Esmée Berger, Ethan Berger, Erik Fransson, Ke Xu, Zihan Yan, Zhoulin Liu, Zichen Song, Haikuan Dong, Shunda Chen, Lei Li, Ziliang Wang, Yizhou Zhu, Julia Wiktor, and Paul Erhart
qNEP: A Highly Efficient Neuroevolution Potential with Dynamic Charges for Large-Scale Atomistic Simulations
Journal of Chemical Theory and Computation **0**, 0 (2026)
DOI: [10.1021/acs.jctc.6c00146](https://doi.org/10.1021/acs.jctc.6c00146)
- [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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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) <[https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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) <<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) <<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) <<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) <<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 Hugoniot method for following the dynamical evolution of shocked matter
Phys. Rev. B. **70**, 014103 (2004)
DOI: [10.1103/PhysRevB.70.014103](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1038/s41467-024-54554-x)
- [Song2024b] Zichen Song, Jian Han, Graeme Henkelman, Lei Li
Charge-Optimized Electrostatic Interaction Atom-Centered Neural Network Algorithm
 Journal of Chemical Theory and Computation **20**, 2088-2097 (2024)
 DOI: [10.1021/acs.jctc.3c01254](https://doi.org/10.1021/acs.jctc.3c01254)
- [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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1016/j.commatsci.2022.111275)
- [Cheng2025] Bingqing Cheng
Latent Ewald summation for machine learning of long-range interactions
npj Computational Materials **11**, 80 (2025)
DOI: [10.1038/s41524-025-01577-7](https://doi.org/10.1038/s41524-025-01577-7)
- [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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1063/1.4774084)
- [Jiang2025] Wenwu Jiang, Ting Liang, Hekai Bu, Jianbin Xu, and Wengen Ouyang
Moiré-driven interfacial thermal transport in twisted transition metal dichalcogenides
 ACS Nano **19**, 16287 (2025)
 DOI: [10.1021/acsnano.4c12148](https://doi.org/10.1021/acsnano.4c12148)
- [Bu2026] Hekai Bu, Wenwu Jiang, Penghua Ying, Ting Liang, Zheyong Fan, and Wengen Ouyang
Modular hybrid machine learning and physics-based potentials for scalable modeling of Van der Waals heterostructures
 J. Mech. Phys. Solids **210**, 106540 (2026)
 DOI: [10.1016/j.jmps.2026.106540](https://doi.org/10.1016/j.jmps.2026.106540)

A

ACE, [211](#)

active (*keyword in run.in*), [121](#)

active.out (*output file*), [143](#)

active.xyz (*output file*), [143](#)

add_efield (*keyword in run.in*), [84](#)

add_force (*keyword in run.in*), [83](#)

add_spring (*keyword in run.in*), [85](#)

ADF, [211](#)

adf.out (*output file*), [149](#)

ADP, [211](#)

Angular Dependent Potential, [48](#)

angular_rdf.out (*output file*), [150](#)

ARDF, [211](#)

atomic_v (*keyword in nep.in*), [161](#)

B

basis_size (*keyword in nep.in*), [159](#)

batch (*keyword in nep.in*), [162](#)

BDP, [211](#)

BEC, [211](#)

bec_test.out (*output file*), [169](#)

bec_train.out (*output file*), [169](#)

Berendsen barostat, [22](#)

Berendsen thermostat, [21](#)

Bibliography, [171](#)

Bussi-Donadio-Parrinello thermostat, [21](#)

C

Canonical ensemble, [20](#)

change_box (*keyword in run.in*), [62](#)

charge_mode (*keyword in nep.in*), [165](#)

charge_test.out (*output file*), [168](#)

charge_train.out (*output file*), [168](#)

cohesive.out (*output file*), [133](#)

compute (*keyword in run.in*), [98](#)

compute.out (*output file*), [134](#)

compute_adf (*keyword in run.in*), [102](#)

compute_angular_rdf (*keyword in run.in*), [120](#)

compute_chunk (*keyword in run.in*), [99](#)

compute_cohesive (*keyword in run.in*), [103](#)

compute_dos (*keyword in run.in*), [103](#)

compute_dpdt (*keyword in run.in*), [105](#)

compute_elastic (*keyword in run.in*), [105](#)

compute_extrapolation (*keyword in run.in*), [60](#)

compute_gkma (*keyword in run.in*), [106](#)

compute_hac (*keyword in run.in*), [107](#)

compute_hnema (*keyword in run.in*), [108](#)

compute_hnemd (*keyword in run.in*), [109](#)

compute_hnemdec (*keyword in run.in*), [110](#)

compute_ic (*keyword in run.in*), [111](#)

compute_lsqt (*keyword in run.in*), [119](#)

compute_msd (*keyword in run.in*), [115](#)

compute_orientorder (*keyword in run.in*), [112](#)

compute_phonon (*keyword in run.in*), [113](#)

compute_rdf (*keyword in run.in*), [119](#)

compute_sdc (*keyword in run.in*), [114](#)

compute_shc (*keyword in run.in*), [116](#)

compute_viscosity (*keyword in run.in*), [118](#)

correct_velocity (*keyword in run.in*), [59](#)

Credits, [169](#)

cutoff (*keyword in nep.in*), [158](#)

D

D.out (*output file*), [135](#)

Deep Potential, [8](#)

deform (*keyword in run.in*), [63](#)

deposit (*keyword in run.in*), [87](#)

descriptor.out (*output file*), [169](#)

dftd3 (*keyword in run.in*), [61](#)

dipole.out (*output file*), [142](#)

dipole_test.out (*output file*), [168](#)

dipole_train.out (*output file*), [168](#)

DOS, [211](#)

dos.out (*output file*), [136](#)

dpdt.out (*output file*), [136](#)

dump.xyz (*output file*), [141](#)

dump_beads (*keyword in run.in*), [123](#)

dump_dipole (*keyword in run.in*), [125](#)

dump_exyz (*keyword in run.in*), [122](#)

dump_force (*keyword in run.in*), [127](#)

dump_netcdf (*keyword in run.in*), [127](#)

dump_observer (*keyword in run.in*), [124](#)

dump_polarizability (*keyword in run.in*), [126](#)

`dump_position` (*keyword in run.in*), 129
`dump_restart` (*keyword in run.in*), 130
`dump_shock_nemd` (*keyword in run.in*), 132
`dump_thermo` (*keyword in run.in*), 131
`dump_velocity` (*keyword in run.in*), 131
`dump_xyz` (*keyword in run.in*), 122

E

EAM, 211
`eigenvector.in`, 57
`electron_stop` (*keyword in run.in*), 89
Embedded atom method, 34
EMD, 211
EMD method, 24
`energy_test.out` (*output file*), 167
`energy_train.out` (*output file*), 167
`ensemble` (*keyword in run.in*), 65
Ensembles, 20

F

FCP, 211
`fine_tune` (*keyword in nep.in*), 164
`fix` (*keyword in run.in*), 90
Force constant potential, 37
`force.out` (*output file*), 137
`force_delta` (*keyword in nep.in*), 162
`force_test.out` (*output file*), 167
`force_train.out` (*output file*), 167
Forces, 19

G

generation (*keyword in nep.in*), 163
GK, 211
GKMA, 27, 211
Glossary, 210
GPU, 211
gpumd executable, 52
gpumd input files, 53
 Atomic configuration, 55
 Eigenvectors, 57
 Simulation model, 55
 Simulation protocol, 53
gpumd input parameters, 58
gpumd output files, 132
Green-Kubo method, 24
Green-Kubo modal analysis, 27

H

HAC, 211
`hac.out` (*output file*), 137
Heat current, 23
heat current autocorrelation, 24
Heat transport, 24

`heatmode.out` (*output file*), 138
HNEMA, 27, 211
HNEMD, 211
HNEMD method, 25
HNEMDEC, 27, 212
Homogeneous non-equilibrium modal analysis, 27
Homogeneous non-equilibrium molecular dynamics, 25
Homogeneous non-equilibrium molecular dynamics Evans-Cummings algorithm, 27

I

IC, 212
`ic.out` (*output file*), 136
ILP, 212
`import_q_scaler` (*keyword in nep.in*), 165
Installation, 3
Interaction models, 28
Interatomic potentials, 28
Isothermal-isobaric ensemble, 22

K

`kappa.out` (*output file*), 139
`kappamode.out` (*output file*), 139
`kpoints.in`, 57
`kspace` (*keyword in run.in*), 91

L

`l_max` (*keyword in nep.in*), 159
`lambda_1` (*keyword in nep.in*), 160
`lambda_2` (*keyword in nep.in*), 160
`lambda_e` (*keyword in nep.in*), 160
`lambda_f` (*keyword in nep.in*), 160
`lambda_q` (*keyword in nep.in*), 161
`lambda_shear` (*keyword in nep.in*), 161
`lambda_v` (*keyword in nep.in*), 161
`lambda_z` (*keyword in nep.in*), 161
Langevin thermostat, 21
Lennard-Jones potential, 33
LJ, 212
`loss.txt` (*output file*), 166
LSQT, 212
`lsqt_dos.out` (*output file*), 149
`lsqt_sigma.out` (*output file*), 150
`lsqt_velocity.out` (*output file*), 150

M

MC, 212
`mc` (*keyword in run.in*), 92
`mcmd.out` (*output file*), 149
MD, 212
Microcanonical ensemble, 20

minimize (*keyword in run.in*), 96
 Modal analysis methods, 26
 model.xyz, 55
 model_type (*keyword in nep.in*), 157
 move (*keyword in run.in*), 91
 movie.xyz (*output file*), 140
 MSD, 212
 msd.out (*output file*), 144
 MSST, 212
 msst (*keyword in run.in*), 82
 MSST integrator, 82
 MTTK, 212
 MTTK integrator, 68
 mvac.out (*output file*), 140

N

n_max (*keyword in nep.in*), 159
 NEMD, 212
 NEMD method, 24
 NEP, 212
 nep executable, 151
 NEP ILP, 41
 nep input files, 153
 nep input parameters, 156
 NEP loss function, 40
 nep output files, 165
 nep.in (*input file*), 153
 nep.restart (*output file*), 167
 nep.txt (*output file*), 167
 NetCDF setup, 6
 Neuroevolution potential, 39
 neuron (*keyword in nep.in*), 160
 NHC, 212
 NN, 212
 NNAP Potential, 14
 Non-equilibrium molecular dynamics, 24
 Nose-Hoover chain thermostat, 21
 nph_mttk (*keyword in run.in*), 68
 nphug (*keyword in run.in*), 83
 NPHug integrator, 83
 NPT ensemble, 22
 npt_mttk (*keyword in run.in*), 68
 npt_qtb (*keyword in run.in*), 70
 NVE ensemble, 20
 NVT ensemble, 20
 nvt_mttk (*keyword in run.in*), 68
 nvt_qtb (*keyword in run.in*), 70

O

observer.out (*output file*), 142
 observer.xyz (*output file*), 142
 omega2.out (*output file*), 141
 onsager.out (*output file*), 147
 orientorder.out (*output file*), 151

output_descriptor (*keyword in nep.in*), 163
 output_interval (*keyword in nep.in*), 163

P

PDOS, 212
 PIMD, 212
 plumed (*keyword in run.in*), 94
 PLUMED setup, 7
 polarizability.out (*output file*), 142
 polarizability_test.out (*output file*), 168
 polarizability_train.out (*output file*), 168
 population (*keyword in nep.in*), 162
 potential (*keyword in run.in*), 60
 prediction (*keyword in nep.in*), 156
 Publications, 173

Q

Quantum Thermal Bath, 70

R

RDF, 212
 rdf.out (*output file*), 148
 replicate (*keyword in run.in*), 58
 restart.xyz (*output file*), 141
 RMSE, 212
 RPMD, 212
 RTC, 212
 run (*keyword in run.in*), 97
 run.in, 53

S

save_potential (*keyword in nep.in*), 163
 SCR, 212
 SDC, 213
 sdc.out (*output file*), 143
 SGC, 213
 SHC, 213
 shc.out (*output file*), 144
 SNES, 213
 Spectral heat current, 25
 Stochastic cell rescaling barostat, 23
 Stress tensor, 19
 stress_test.out (*output file*), 168
 stress_train.out (*output file*), 168
 SVR, 213
 SW, 213
 SW ILP, 44

T

Tersoff ILP, 46
 Tersoff mini-potential, 32
 Tersoff potential (1988), 29
 Tersoff potential (1989), 30

test.xyz (*input file*), 154
Theoretical background, 17
thermo.out (*output file*), 145
time_step (*keyword in run.in*), 64
train.xyz (*input file*), 154
TRPMD, **213**
ttm_electron_temperature.out (*output file*), 134
type (*keyword in nep.in*), 157
type_weight (*keyword in nep.in*), 157

U

use_typewise_cutoff_zbl (*keyword in nep.in*), 158

V

VAC, **213**
VCSGC, **213**
velocity (*keyword in run.in*), 58
velocity.out (*output file*), 146
version (*keyword in nep.in*), 156
virial_test.out (*output file*), 168
virial_train.out (*output file*), 168
viscosity.out (*output file*), 146

Z

ZBL, **213**
zbl (*keyword in nep.in*), 157