ALAMODE Documentation

Release 1.1.0

Terumasa Tadano

Jul 19, 2020

Contents

CHAPTER 1

Users Guide

1.1 About

1.1.1 What is ALAMODE?

ALAMODE is an open source software designed for analyzing lattice anharmonicity and lattice thermal conductivity of solids. By using an external DFT package such as VASP and Quantum ESPRESSO, you can extract harmonic and anharmonic force constants straightforwardly with ALAMODE. Using the calculated anharmonic force constants, you can also estimate lattice thermal conductivity, phonon linewidth, and other anharmonic phonon properties from first principles.

1.1.2 Features

General

- Extraction of harmonic and anharmonic force constants based on the supercell approach
- Applicable to any crystal structures and low-dimensional systems
- Accurate treatment of translational and rotational invariance
- Interface to VASP, Quantum-ESPRESSO, xTAPP, and LAMMPS codes
- Mainly written in C++, parallelized with MPI+OpenMP

Harmonic properties

- Phonon dispersion
- Phonon DOS, atom-projected phonon DOS
- Two-phonon DOS
- Vibrational thermodynamic functions (heat capacity, entropy, free energy)

- Mean-square displacement
- Animation and visualization of phonon modes (requires VMD or XCrysDen)
- 3-phonon scattering phase space
- Phonon-isotope scattering rate
- Participation ratio for analyzing the localization of phonon modes

Anharmonic properties

- Grüneisen parameter via cubic force constants
- Lattice thermal conductivity by BTE-RTA
- Cumulative thermal conductivity
- Phonon linewidth due to 3-phonon interactions
- Phonon frequency shift due to 3- and 4-phonon interactions
- Temperature-dependent effective potential method
- Self-consistent phonon (SCPH) calculation
- Anharmonic vibrational free-energy

1.1.3 Links

- Download page : http://sourceforge.net/projects/alamode
- Documentation : http://alamode.readthedocs.io (this page)
- Git repository : http://github.com/ttadano/alamode

1.1.4 License

1.1.5 How to Cite ALAMODE

Please cite the following article when you use ALAMODE:

T. Tadano, Y. Gohda, and S. Tsuneyuki, J. Phys.: Condens. Matter **26**, 225402 (2014) [Link].

If you use the self-consistent phonon (SCPH) method, please cite the following paper as well:

T. Tadano and S. Tsuneyuki, Phys. Rev. B **92**, 054301 (2015). [Link]

If you use ALAMODE to compute anharmonic vibrational free-energies in your research paper, please cite the following paper as well:

Y. Oba, T. Tadano, R. Akashi, and S. Tsuneyuki, Phys. Rev. Materials **3**, 033601 (2019). [Link]

1.1.6 Acknowledgment

This project is/was partially supported by the following projects:

- Grant-in-Aid for Young Scientists (B) (16K17724)
- Grant-in-Aid for Scientific Research on Innovative Areas 'Materials Design through Computics: Complex Correlation and Non-Equilibrium Dynamics'. (http://computics-material.jp)

1.1.7 Author & Contact

Research Center for Magnetic and Spintronic Materials (CMSM),
National Institute for Material Science (NIMS),
Japan

If you have any questions, suggestions, and problems regarding ALAMODE, please feel free to contact the author.

1.2 Download

You can download the latest and previous versions of ALAMODE at http://sourceforge.net/projects/alamode .

You can also download the package from the git repository as

```
$ git clone http://github.com/ttadano/alamode.git
```

If you choose the GitHub version, please use the 'master' branch.

1.3 Installation

1.3.1 Requirement

Mandatory requirements

- C++ compiler (Intel compiler is recommended.)
- LAPACK library
- MPI library (OpenMPI, MPICH2, IntelMPI, etc.)
- Boost C++ library
- FFTW library
- Eigen3 library
- spglib

In addition to the above requirements, users have to get and install a first-principles package (such as VASP, QUANTUM-ESPRESSO, OpenMX, or xTAPP) or another force field package (such as LAMMPS) by themselves in order to compute harmonic and anharmonic force constants.

Optional requirements

- Python (> 2.6), Numpy, and Matplotlib
- XcrySDen or VMD

We provide some small scripts written in Python for visualizing phonon dispersion relations, phonon DOSs, etc. To use these scripts, one need to install the above Python packages. Additionally, XcrySDen is necessary to visualize the normal mode directions and animate the normal mode. VMD may be more useful to make an animation, but it may be replaced by any other visualization software which supports the XYZ format.

1.3.2 How to install

Here, we do not explain how to install a C++ compiler, LAPACK, MPI, and FFTW libraries because they are usually available on supercomputing systems.

Boost C++ and Eigen3 libraries (header files only)

Some header files of Boost C++ and Eigen3 libraries are necessary to build ALAMODE binaries. Here, we install header files of these libraries in `$(HOME)/include`. You can skip this part if these libraries are already installed on your system.

To install the Boost C++ library, please download a source file from the webpage and unpack the file. Then, copy the 'boost' subdirectory to `$(HOME)/include`. This can be done as follows:

```
$ cd
$ mkdir etc; cd etc
(Download a source file and mv it to ~/etc)
$ tar xvf boost_x_yy_z.tar.bz2
$ cd ../
$ mkdir include; cd include
$ ln -s ../etc/boost_x_yy_z/boost .
```

In this example, we place the boost files in `$(HOME)/etc` and create a symbolic link to the `$(HOME)/boost_x_yy_z/boost` in `$(HOME)/include`. Instead of installing from source, you can install the Boost library with Homebrew on Mac OSX.

In the same way, please install the Eigen3 include files as follows:

```
$ cd
$ mkdir etc; cd etc
(Download a source file and mv it to ~/etc)
$ tar xvf eigen-eigen-*.tar.bz2 (* is an array of letters and digits)
$ cd ../
$ cd include
$ ln -s ../etc/eigen-eigen-*/Eigen .
```

If you have followed the instruction, you will see the following results:

```
$ pwd
/home/tadano/include
$ ls -l
total 0
lrwxrwxrwx 1 tadano sim00 25 May 17  2017 boost -> ../etc/boost_1_64_0/boost
lrwxrwxrwx 1 tadano sim00 38 May 17  2017 Eigen -> ../etc/eigen-eigen-67e894c6cd8f/
↪Eigen/
```

spglib

ALAMODE uses spglib to handle symmetries of crystal structures. Please install it by following the instruction on the spglib webpage. Here, we assume spglib is installed in `$(SPGLIB_ROOT)`.

Download ALAMODE source

From the download page:

```
$ (visit https://sourceforge.net/projects/alamode/files/latest/download?source=files␣
↪to download the latest version source)
$ tar xvzf alamode-x.y.z.tar.gz
$ cd alamode-x.y.z
```

From GitHub repository:

```
$ git clone https://github.com/ttadano/alamode.git
$ cd alamode
$ git checkout master
```

The meaning of each subdirectory is as follows:

- alm/ : Source files of alm (force constant calculator)
- anphon/ : Source files of anphon (anharmonic phonon calculator)
- external/ : Third-party include files
- include/ : Commonly-used include files
- tools/ : Small auxiliary programs and scripts
- docs/ : Source files for making documents
- example/ : Example files

Build ALAMODE by Makefile

ALAMODE contains two major codes, **alm** and **anphon**, and other small utility scripts. In directories `alm/`, `anphon/`, and `tools`, we provide sample Makefiles for Linux (Intel compiler) and Mac OSX (gcc, clang). Please copy either of them, edit the options appropriately, and issue `make` command as follows:

```
$ cd alm/
$ cp Makefile.linux Makefile
(Edit Makefile here)
$ make -j

$ cd ../anphon/
$ cp Makefile.linux Makefile
(Edit Makefile here)
$ make -j

$ cd ../tools/
(Edit Makefile here)
$ make -j
```

An example of the Makefiles is shown below:

Listing 1: **ALM Makefile.linux**

```
CXX = icpc
CXXFLAGS = -O2 -xHOST -qopenmp -std=c++11
INCLUDE = -I../include -I$(HOME)/include -I$(SPGLIB_ROOT)/include

CXXL = ${CXX}
LDFLAGS = -mkl -L$(SPGLIB_ROOT)/lib -lsymspg

LAPACK =
LIBS = ${LAPACK}
```

The default options are expected to work with modern Intel compilers.

Note: To build the binaries with the example Makefiles, you need to set `SPGLIB_ROOT` beforehand from the terminal as:

```
$ export SPGLIB_ROOT=/path/to/spglib/installdir
```

Edit LD_LIBRARY_PATH in bashrc

Finally, add the following line in `$(HOME)/.bashrc` (or `.zshrc` etc.):

```
(bash, zsh)
export LD_LIBRARY_PATH=/path/to/spglib/installdir/lib:$LD_LIBRARY_PATH

(csh, tcsh)
setenv LD_LIBRARY_PATH /path/to/spglib/installdir/lib:$LD_LIBRARY_PATH
```

This is necessary when you link spglib dynamically.

1.4 Running ALAMODE

1.4.1 Program alm

Program *alm* estimates harmonic and anharmonic interatomic force constants (IFCs) based on the *supercell approach*.

1. Perform usual SCF calculations for a *primitive cell*

   Before performing phonon calculations, one needs to perform usual self-consistent field calculations and check the convergence with respect to the cutoff energy and the k point density. After that, please optimize the internal coordinate so that the atomic forces are negligibly small. Optimization of cell parameters may also be necessary, but please note that phonon properties are relatively sensitive to the cell parameters in polar materials such as perovskites.

2. Decide the size of supercell

   Next, please decide the size of a supercell. Here, one may use a conventional cell. When the primitive cell is fairly large ($a \sim 10$ Å), one may proceed using the primitive cell.

3. Prepare an input file for *alm*

Please make an input file for *alm*, say alm.in, which should contain `&general`, `&interaction`, `&cutoff`, and `&position` entries. For details of available input variables, please refer to *here*. Once the input file is properly prepared with `MODE = suggest`, necessary displacement patterns can be generated by executing *alm* as follows:

```
$ alm alm.in > alm.log
```

This produces the following files containing the pattern of atomic displacements.

- `PREFIX`.HARMONIC_pattern
- `PREFIX`.ANHARM?_pattern (If `NORDER` > 1)

In pattern files, all necessary displacement patterns are given in **Cartesian coordinates**.

Note: Pattern files just indicate the direction of displacements. The magnitude of displacements should be specified by each user. ($\Delta u \sim 0.01$ Å is usual for calculating harmonic force constants.)

---

4. Perform SCF calculations to generate displacement-force data set

   Then, please prepare necessary input files for a DFT engine (or a classical force field engine) and calculate atomic forces for each displaced configuration. Once the atomic forces are calculated for all configurations, please collect the atomic displacements and atomic forces to separate files, say *disp_all.dat* and *force_all.dat*, in Rydberg atomic units. The detail of the file format is described on *this page*.

   Note: We provide some auxiliary Python scripts to expedite the above procedure for VASP, Quantum ESPRESSO, and xTAPP users. The script files can be found in the tools/ directory. We are willing to support other software if necessary.

   ---

5. Estimate IFCs by linear regression

   In order to perform a fitting, please change the variable `MODE` of the input file *alm.in* to `MODE = optimize`. In addition please add the `&optimize` entry with appropriate `DFSET`. Then, IFCs can be estimated by executing

   ```
   $ alm alm.in > alm.log
   ```

   which makes the following two files in the working directory.

   - `PREFIX`.fcs : The list of force constants
   - `PREFIX`.xml : XML file containing necessary information for subsequent phonon calculations

1.4.2 Program anphon

1. Prepare an input file for *anphon*

   To perform phonon calculations and thermal conductivity calculations, one needs to prepare another input file, say *anphon.in*, for the program *anphon*.

   If one wants to perform (harmonic) phonon calculations, one should write `MODE = phonons` in the `&general` entry of anphon.in. Please make sure that `FCSXML` variable is set to the XML file generated by *alm*.

   If one wants to conduct thermal conductivity calculations instead of usual phonon calculations, please switch to `MODE = RTA` with appropriate `FCSXML` containing cubic IFCs.

For details of input variables of anphon, please refer to the *list of input variables for anphon*.

2. Execute *anphon*

   Phonon properties and lattice thermal conductivity can be calculated via executing

```
$ anphon anphon.in > anphon.log
```

   or

```
$ mpirun -np NPROCS anphon anphon.in > anphon.log
```

   Here, `NPROCS` is the number of MPI threads. If the code is compiled with the OpenMP option, the OpenMP parallelization can also be used by setting the `OMP_NUM_THREADS` variable as

```
$ export OMP_NUM_THREADS=16
```

   The number 16 should be modified appropriately for your environment.

   Note: MPI+OpenMP hybrid parallelization is available when calculating thermal conductivity with `MODE = RTA`, in which anharmonic self-energies of all $N_{\boldsymbol{q},irred} \times N_j$ phonon modes need to be calculated. Here $N_{\boldsymbol{q},irred}$ and N_j are the number of irreducible q points and the number of phonon branches, respectively. These phonon modes are distributed across `NPROCS` MPI threads, and phonon self-energies are calculated in parallel. OpenMP is used for the double loop over the N_j branches inside the calculation of each phonon self-energy. Therefore, a good performance is expected when `OMP_NUM_THREADS` is a divisor of N_j^2.

   When the calculation finishes normally, various files are generated in the working directory.

   - `PREFIX`.bands : Phonon dispersion along designated Brillouin zone paths
   - `PREFIX`.dos : (Atom projected) phonon DOS
   - `PREFIX`.thermo : Thermodynamic functions
   - `PREFIX`.msd : Mean-square displacement of atoms
   - ...

   The complete list of output files can be found *here*.

3. Analyze the result

   One can plot the phonon dispersion relation or phonon DOS using gnuplot. Alternatively, one can use a small script in the `tools/` directory for visualizing these results. For example,

```
$ plotband.py target.bands
```

   shows the phonon dispersion relation. Available command line options can be displayed by

```
$ plotband.py -h
```

   We also provide a similar script for phonon DOS. Another script `analyze_phonons.py` may be useful to analyze the result of thermal conductivity calculations. For example, phonon lifetimes and mean-free-path at 300 K can be extracted by

```
$ analyze_phonons.py --calc tau --temp 300 target.result
```

   It can also estimate a cumulative thermal conductivity by

```
$ analyze_phonons.py --calc cumulative --temp 300 --direction 1 target.result
```

For details, see the *tutorial*.

1.5 ALM: Force constant calculator

1.5.1 ALM: Input files

Format of input files

Each input file should consist of entry fields. Available entry fields are

&general, **&interaction**, **&cutoff**, **&cell**, **&position**, and **&optimize** (**&fitting**).

Each entry field starts from the key label **&field** and ends at the terminate character "/". (This is equivalent to Fortran namelist.)

For example, &general entry field of program *alm* should be given like

```
&general
  # Comment line
  PREFIX = prefix
  MODE = fitting
/
```

Multiple entries can be put in a single line. Also, characters put on the right of sharp ("#") are neglected. Therefore, the above example is equivalent to the following:

```
&general
  PREFIX = prefix; MODE = fitting  # Comment line
/
```

Each variable must be given inside the appropriate entry field.

List of supported input variables

&general				
HESSIAN	*KD*	*MODE*	*NAT*	*NKD*
PERIODIC	*PREFIX*	*PRINTSYM*	*TOLERANCE*	
&interaction				
NBODY	*NORDER*			
&optimize				
CONV_TOL	*CV*	*CV_MINALPHA*	*DEBIAS_OLS*	*DFILE*
DFSET	*DFSET_CV*	*ENET_DNORM*	*FC2XML*	*FC3XML*
FFILE	*ICONST*	*L1_ALPHA*	*L1_RATIO*	*LMODEL*
MAXITER	*NDATA*	*NDATA_CV*	*NSTART NEND*	*NSTART_CV NEND_CV*
ROTAXIS	*SKIP*	*SOLUTION_PATH*	*SPARSE*	*SPARSESOLVER*
STANDARDIZE				

Description of input variables

"&general"-field

- **PREFIX**-tag : Job prefix to be used for names of output files

  Default None

  Type String

- **MODE**-tag = optimize | suggest | fitting

optimize (>= 1.1.0)	Estimate harmonic and anharmonic IFCs. This mode requires an appropriate &optimize field.
fitting (deprecated)	An alias of `MODE = optimize`
suggest	Suggests the displacement patterns necessary to estimate harmonic and anharmonic IFCS.

  Default None

  Type String

- **NAT**-tag : Number of atoms in the supercell

  Default None

  Type Integer

- **NKD**-tag : Number of atomic species

  Default None

  Type Integer

- **KD**-tag = Name[1], . . . , Name[`NKD`]

  Default None

  Type Array of strings

  Example In the case of GaAs with `NKD = 2`, it should be `KD = Ga As`.

- TOLERANCE-tag : Tolerance for finding symmetry operations

Default 1.0e-3

Type Double

- PRINTSYM-tag = 0 | 1

0	Symmetry operations won't be saved in "SYMM_INFO"
1	Symmetry operations will be saved in "SYMM_INFO"

Default 0

type Integer

- PERIODIC-tag = PERIODIC[1], PERIODIC[2], PERIODIC[3]

0	Do not consider periodic boundary conditions when searching for interacting atoms.
1	Consider periodic boundary conditions when searching for interacting atoms.

Default 1 1 1

type Array of integers

Description This tag is useful for generating interacting atoms in low dimensional systems. When `PERIODIC[i]` is zero, periodic boundary condition is turned off along the direction of the lattice vector $\boldsymbol{a}_i$.

- HESSIAN-tag = 0 | 1

0	Do not save the Hessian matrix
1	Save the entire Hessian matrix of the supercell as PREFIX.hessian.

Default 0

type Integer

"&interaction"-field

- **NORDER**-tag : The order of force constants to be calculated. Anharmonic terms up to $(m+1)$th order will be considered with `NORDER` = m.

Default None

Type Integer

Example `NORDER = 1` for calculate harmonic terms only, `NORDER = 2` to include cubic terms as well, and so on.

- NBODY-tag : Entry for excluding multiple-body interactions from anharmonic force constants

  Default `NBODY` = [2, 3, 4, . . . , `NORDER` + 1]

  Type Array of integers

  Description This tag may be useful for excluding multi-body clusters which are supposedly less important. For example, a set of fourth-order IFCs $\{\Phi_{ijkl}\}$, where i, j, k, and l label atoms in the supercell, can be categorized into four different subsets; **on-site**, **two-body**, **three-body**, and **four-body** terms. Neglecting the Cartesian coordinates of IFCs for simplicity, each subset contains the IFC elements shown as follows:

on-site	$\{\Phi_{iiii}\}$
two-body	$\{\Phi_{iijj}\}$, $\{\Phi_{iiij}\}$ $(i \neq j)$
three-body	$\{\Phi_{iijk}\}$ $(i \neq j, i \neq k, j \neq k)$
four-body	$\{\Phi_{ijkl}\}$ (all subscripts are different from each other)

  Since the four-body clusters are expected to be less important than the three-body and less-body clusters, you may want to exclude the four-body terms from the Taylor expansion potential because the number of such terms is huge. This can be done by setting the `NBODY` tag as `NBODY = 2 3 3` together with `NORDER = 3`.

  More examples `NORDER = 2; NBODY = 2 2` includes harmonic and cubic IFCs but excludes three-body clusters from the cubic terms.

  `NORDER = 5; NBODY = 2 3 3 2 2` includes anharmonic terms up to the sixth-order, where the four-body clusters are excluded from the fourth-order IFCs, and the multi (≥ 3)-body clusters are excluded from the fifth- and sixth-order IFCs.

"&cutoff"-field

In this entry field, one needs to specify cutoff radii of interaction for each order in units of Bohr. In the current implementation, cutoff radii should be defined for every possible pair of atomic elements. For example, the cutoff entry for a harmonic calculation (`NORDER = 1`) of Si (`NKD = 1`) should be like

```
&cutoff
 Si-Si 10.0
/
```

This means that the cutoff radius of 10 a_0 is used for harmonic Si-Si terms. Please note that the first column should be two character strings, which are contained in the KD-tag, connected by a hyphen ('-').

When one wants to consider cubic terms (NORDER = 2), please specify the cutoff radius for cubic terms in the third column as the following:

```
&cutoff
 Si-Si 10.0 5.6 # Pair r_{2} r_{3}
/
```

Instead of giving specific cutoff radii, one can write "None" as follows:

```
&cutoff
 Si-Si None 5.6
/
```

which means that all possible harmonic terms between Si-Si atoms will be included.

Caution: Setting 'None' for anharmonic terms can greatly increase the number of parameters and thereby increase the computational cost.

When there are more than two atomic elements, please specify the cutoff radii between every possible pair of atomic elements. In the case of MgO (NKD = 2), the cutoff entry should be like

```
&cutoff
 Mg-Mg 8.0
 O-O 8.0
 Mg-O 10.0
/
```

which can equivalently be written by using the wild card ('*') as

```
&cutoff
 *-* 8.0
 Mg-O 10.0 # Overwrite the cutoff radius for Mg-O harmonic interactions
/
```

Important: Cutoff radii specified by an earlier entry are overwritten by a new entry that comes later.

Once the cutoff radii are properly given, harmonic force constants $\Phi_{i,j}^{\mu,\nu}$ satisfying $r_{ij} \leq r_c^{\mathrm{KD}[i]-\mathrm{KD}[j]}$ will be searched.

In the case of cubic terms, force constants $\Phi_{ijk}^{\mu\nu\lambda}$ satisfying $r_{ij} \leq r_c^{\mathrm{KD}[i]-\mathrm{KD}[j]}$, $r_{ik} \leq r_c^{\mathrm{KD}[i]-\mathrm{KD}[k]}$, and $r_{jk} \leq r_c^{\mathrm{KD}[j]-\mathrm{KD}[k]}$ will be searched and determined by fitting.

"&cell"-field

Please give the cell parameters in this entry in units of Bohr as the following:

```
&cell
 a
 a11 a12 a13
 a21 a22 a23
 a31 a32 a33
/
```

The cell parameters are then given by $\vec{a}_1 = a \times (a_{11}, a_{12}, a_{13})$, $\vec{a}_2 = a \times (a_{21}, a_{22}, a_{23})$, and $\vec{a}_3 = a \times (a_{31}, a_{32}, a_{33})$.

"&position"-field

In this field, one needs to specify the atomic element and fractional coordinate of atoms in the supercell. Each line should be

```
ikd xf[1] xf[2] xf[3]
```

where *ikd* is an integer specifying the atomic element (*ikd* = 1, ..., NKD) and *xf[i]* is the fractional coordinate of an atom. There should be NAT such lines in the &position entry field.

"&optimize"-field ("&fitting"-field)

This field is necessary when MODE = optimize (or a deprecated option MODE = fitting).

- LMODEL-tag : Choice of the linear model used for estimating force constants

"least-squares", "LS", "OLS", 1	Ordinary least square
"elastic-net", "enet", 2	Elastic net

Default least-squares

Type String

Description When LMODEL = ols, the force constants are estimated from the displacement-force datasets via the ordinary least-squares (OLS), which is usually sufficient to calculate harmonic and third-order force constants.

The elastic net (LMODEL = enet) should be useful to calculate the fourth-order (and higher-order) force constants. When the elastic net is selected, the users have to set the following related tags: CV, L1_RATIO, L1_ALPHA, CV_MAXALPHA, CV_MINALPHA, CV_NALPHA, STANDARDIZE, ENET_DNORM, MAXITER, CONV_TOL, NWRITE, SOLUTION_PATH, DEBIAS_OLS

- **DFSET**-tag: File name containing displacement-force datasets for training

  New in version 1.1.0.

  Default None

  Type String

  Description The format of DFSET can be found *here*

- DFILE-tag: File name containing atomic displacements in Cartesian coordinate

  Deprecated since version 1.1.0: Use `DFSET` instead.

  Default None

  Type String

  Description The format of `DFILE` can be found *here*. This tag is deprecated and will be removed in a future major release. Please use `DFSET` instead.

- FFILE-tag: File name containing atomic forces in Cartesian coordinate

  Deprecated since version 1.1.0: Use `DFSET` instead.

  Default None

  Type String

  Description The format of `FFILE` can be found *here*. This tag is deprecated and will be removed in a future major release. Please use `DFSET` instead.

- NDATA-tag : Number of displacement-force data sets

  Default None

  Type Integer

  Description If `NDATA` is not given, the code reads all lines of `DFSET` (excluding comment lines) and estimates `NDATA` by dividing the line number by `NAT`. If the number of lines is not divisible by `NAT`, an error is raised. `DFSET` should contain at least `NDATA`$\times$ `NAT` lines.

- NSTART, NEND-tags : Specifies the range of data to be used for fitting

  Default `NSTART = 1`, `NEND = NDATA`

  Type Integer

  Example To use the data in the range of [20:30] out of 50 entries, the tags should be `NSTART = 20` and `NEND = 30`.

- SKIP-tag : Specifies the range of data to be skipped for training

  Default None

  Type Two integers connected by a hyphen

  Description `SKIP` =i-j skips the data in the range of [i:j]. The i and j must satisfy $1 \leq i \leq j \leq$ `NDATA`. This option may be useful when doing cross-validation manually (`CV=-1`).

- ICONST-tag = 0 | 1 | 2 | 3 | 11

0	No constraints
1	Constraint for translational invariance is imposed between IFCs. Available only when `LMODEL = ols`.
11	Same as `ICONST = 1` but the constraint is imposed *algebraically* rather than numerically. Select this option when `LMODEL = enet`.
2	In addition to `ICONST = 1`, constraints for rotational invariance will be imposed up to (`NORDER` + 1)th order. Available only when `LMODEL = ols`.
3	In addition to `ICONST = 2`, constraints for rotational invariance between (`NORDER` + 1)th order and (`NORDER` + 2)th order, which are zero, will be considered. Available only when `LMODEL = ols`.

Default 11

Type Integer

Description See *this page* for the numerical formulae.

- ROTAXIS-tag : Rotation axis used to estimate constraints for rotational invariance. This entry is necessary when `ICONST = 2, 3`.

  Default None

  Type String

  Example When one wants to consider the rotational invariance around the x-axis, one should give `ROTAXIS = x`. If one needs additional constraints for the rotation around the y-axis, `ROTAXIS` should be `ROTAXIS = xy`.

- FC2XML-tag : XML file to which the harmonic terms are fixed upon fitting

  Default None

  Type String

  Description When `FC2XML`-tag is given, harmonic force constants are fixed to the values stored in the `FC2XML` file. This may be useful for optimizing cubic and higher-order terms

without changing the harmonic terms. Please make sure that the number of harmonic terms in the new computational condition is the same as that in the `FC2XML` file.

- FC3XML-tag : XML file to which the cubic terms are fixed upon fitting

  Default None

  Type String

  Description Same as the `FC2XML`-tag, but `FC3XML` is to fix cubic force constants.

- SPARSE-tag = 0 | 1

0	Use a direct solver (SVD or QRD) to estimate force constants
1	Use a sparse solver to estimate force constants

  Default 0

  Type Integer

  Description When you want to calculate force constants of a large system and generate training datasets by displacing only a few atoms from equilibrium positions, the resulting sensing matrix becomes large but sparse. For such matrices, a sparse solver is expected to be more efficient than SVD or QRD in terms of both memory usage and computational time. When `SPARSE = 1` is set, the code uses a sparse solver implemented in Eigen3 library. You can change the solver type via `SPARSESOLVER`. Effective when `LMODEL = ols`.

- SPARSESOLVER-tag : Type of the sparse solver to use

  Default SimplicialLDLT

  Type String

  Description Currently, only the sparse solvers of Eigen3 library can be used. Available options are *SimplicialLDLT*, *SparseQR*, *ConjugateGradient*, *LeastSquaresConjugateGradient*, and *BiCGSTAB*. When an iterative algorithm (conjugate gradient) is selected, a stopping criterion can be specified by the `CONV_TOL` and `MAXITER` tags. Effective when `LMODEL = ols` and `SPARSE = 1`.

  See also:

  Eigen documentation page: Solving Sparse Linear Systems

- MAXITER-tag : Number of maximum iterations in iterative algorithms

  Default 10,000

  Type Integer

  Description Effective when an iterative solver is selected via `SPARSESOLVER` (when `LMODEL = ols`) or when `LMODEL = enet`.

- CONV_TOL-tag : Convergence criterion of iterative algorithms

  Default 1.0e-8

Type Double

Description When `LMODEL = ols` and an iterative solver is selected via `SPARSESOLVER`, `CONV_TOL` value is passed to the Eigen3 function via *setTolerance()*. When `LMODEL = enet`, the coordinate descent iteration stops at ith iteration if $\sqrt{\frac{1}{N}|\mathbf{\Phi}_i - \mathbf{\Phi}_{i-1}|_2^2} <$ `CONV_TOL` is satisfied, where N is the length of the vector $\mathbf{\Phi}$.

See also:

Eigen documentation page: IterativeSolverBase

- L1_RATIO-tag : The ratio of the L1 regularization term

  Default 1.0 (LASSO)

  Type Double

  Description The `L1_RATIO` changes the regularization term as `L1_ALPHA` $\times$ [`L1_RATIO` $|\mathbf{\Phi}|_1$ + $\frac{1}{2}$ (1-`L1_RATIO`) $|\mathbf{\Phi}|_2^2$]. Therefore, `L1_RATIO = 1` corresponds to LASSO. `L1_RATIO` must be `0 < L1_ratio <= 1`. Effective when `LMODEL = enet`. See also *here*.

- L1_ALPHA-tag : The coefficient of the L1 regularization term

  Default 0.0

  Type Double

  Description This tag is used only when `LMODEL = enet` and `CV = 0`. See also *here*.

- CV-tag : Cross-validation mode for elastic net

0	Cross-validation mode is off. The elastic net optimization is solved with the given L1_ALPHA value. The force constants are written to PREFIX.fcs and PREFIX.xml.
>= 2	CV-fold cross-validation is performed *automatically*. NDATA training datasets are divided into CV subsets, and CV different combinations of training-validation datasets are created internally. For each combination, the elastic net optimization is solved with the various L1_ALPHA values defined by the CV_MINALPHA, CV_MAXALPHA, and CV_NALPHA tags. The result of each cross-validation is stored in PREFIX.enet_cvset[1, . . . , CV], and their average and deviation are stored in PREFIX.cvscore.
-1	The cross-validation is performed *manually*. The Taylor expansion potential is trained by using the training datasets in DFSET, and the validation score is calculated by using the data in DFSET_CV for various L1_ALPHA values defined the CV_MINALPHA, CV_MAXALPHA, and CV_NALPHA tags. After the calculation, the fitting and validation errors are stored in PREFIX.enet_cv. This option may be convenient for a large-scale problem since multiple optimization tasks with different training-validation datasets can be done in parallel.

Default 0

Type Integer

Description This tag is used only when `LMODEL = enet`.

- DFSET_CV-tag : File name containing displacement-force datasets used for manual cross-validation

  Default `DFSET_CV = DFSET`

  Type String

Description This tag is used only when `LMODEL = enet` and `CV = -1`.

- NDATA_CV-tag : Number of displacement-force validation datasets

  Default None

  Type Integer

  Description This tag is used only when `LMODEL = enet` and `CV = -1`.

- NSTART_CV, NEND_CV-tags : Specifies the range of data to be used for validation

  Default `NSTART_CV = 1`, `NEND_CV = NDATA_CV`

  Type Integer

  Example This tag is used only when `LMODEL = enet` and `CV = -1`.

- CV_MINALPHA, CV_MAXALPHA, CV_NALPHA-tags : Options to specify the `L1_ALPHA` values used in cross-validation

  Default `CV_MINALPHA = 1.0e-4`, `CV_MAXALPHA = 1.0`, `CV_NALPHA = 1`

  Type Double, Double, Integer

  Description `CV_NALPHA` values of `L1_ALPHA` are generated from `CV_MINALPHA` to `CV_MAXALPHA` in logarithmic scale. A recommended value of `CV_MAXALPHA` is printed out to the log file. This tag is used only when `LMODEL = enet` and the cross-validation mode is on (`CV > 0` or `CV = -1`).

- STANDARDIZE-tag = 0 | 1

0	Do not standardize the sensing matrix
1	Each column of the sensing matrix is standardized in such a way that its mean value becomes 0 and standard deviation becomes 1.

  Default 1

  Type Integer

  Description This option influences the optimal `L1_ALPHA` value. So, if you change the `STANDARDIZE` option, you have to rerun the cross-validation. Effective only when `LMODEL = enet`.

- ENET_DNORM-tag : Normalization factor of atomic displacements

  Default 1.0

  Type Double

Description The normalization factor of atomic displacement u_0 in units of Bohr. When $u_0(\neq 1)$ is given, the displacement data are scaled as $u_i \to u_i/u_0$ before constructing the sensing matrix. This option influences the optimal `L1_ALPHA` value. So, if you change the `ENET_DNORM` value, you will have to rerun the cross-validation. Effective only when `LMODEL = enet` and `STANDARDIZE = 0`.

- SOLUTION_PATH-tag = 0 | 1

0	Do not save the solution path.
1	Save the solution path of each cross-validation combination in `PREFIX`.solution_path.

Default 0

Type Integer

Description Effective when `LMODEL = enet` and the cross-validation mode is on.

- DEBIAS_OLS-tag = 0 | 1

0	Save the solution of the elastic net problem to `PREFIX`.fcs and `PREFIX`.xml.
1	After the solution of the elastic net optimization problem is obtained, only non-zero coefficients are collected, and the ordinary least-squares fitting is solved again with the non-zero coefficients before saving the results to `PREFIX`.fcs and `PREFIX`.xml. This might be useful to reduce the bias of the elastic net solution.

Default 0

Type Integer

Description Effective when `LMODEL = enet` and `CV = 0`.

How to make a DFSET file

Format of `DFSET` (recommended as of ver. 1.1.0)

The displacement-force data sets obtained by first-principles (or classical force-field) calculations have to be saved to a file, say *DFSET*. Then, the force constants are estimated by setting `DFSET` = *DFSET* and with `MODE = optimize`.

The *DFSET* file must contain the atomic displacements and corresponding forces in Cartesian coordinate for at least `NDATA` structures (displacement patterns) in the following format: Structure number 1 (this is just a comment line)

$$u_x(1) \qquad u_y(1) \qquad u_z(1) \qquad\qquad\qquad\qquad \mathrm{f}_x(1)$$

$$
\begin{array}{cccc}
f_y(1) & f_z(1) & & \\
u_x(2) & u_y(2) & u_z(2) & \mathrm{f}_x(2) \\
f_y(2) & f_z(2) & & \\
 & \vdots & & \\
\vdots & & & \\
u_x(\mathrm{NAT}) & u_y(\mathrm{NAT}) & u_z(\mathrm{NAT}) & \mathrm{f}_x(\mathrm{NAT}) \\
f_y(\mathrm{NAT}) & f_z(\mathrm{NAT}) & &
\end{array}
$$

Structure number 2

$$
\begin{array}{cccc}
u_x(1) & u_y(1) & u_z(1) & \mathrm{f}_x(1) \\
f_y(1) & f_z(1) & & \\
 & \vdots & & \\
\vdots & & &
\end{array}
$$

Here, `NAT` is the number of atoms in the supercell. The unit of displacements and forces must be **Bohr** and **Ryd/Bohr**, respectively.

Format of `DFILE` and `FFILE` (deprecated)

Deprecated since version 1.1.0: Use `DFSET` instead.

The displacement-force data sets obtained by first-principles (or classical force-field) calculations have to be saved to `DFILE` and `FFILE` to estimate IFCs with `MODE = fitting`. In `DFILE`, please explicitly specify the atomic displacements $u_\alpha(\ell\kappa)$ **in units of Bohr** as follows:

$$
\begin{array}{ccc}
u_x(1) & u_y(1) & u_z(1) \\
u_x(2) & u_y(2) & u_z(2) \\
 & \vdots & \\
u_x(\mathrm{NAT}) & u_y(\mathrm{NAT}) & u_z(\mathrm{NAT})
\end{array}
$$

When there are `NAT` atoms in the supercell and `NDATA` data sets, there should be `NAT` $\times$ `NDATA` lines in the `DFILE` without blank lines. In `FFILE`, please specify the corresponding atomic forces **in units of Ryd/Bohr**.

1.5.2 ALM: Output files

- `PREFIX`.pattern_HARMONIC, `PREFIX`.pattern_ANHARM?

  In these files, displacement patterns are printed in units of $\boldsymbol{e}_{x,y,z}$. These files are created when `MODE = suggest`. Patterns for anharmonic force constants are printed only when `NORDER > 1`.

- `PREFIX`.fcs

  Harmonic and anharmonic force constants in Rydberg atomic units. In the first section, only symmetry-reduced force constants are printed. All symmetry-related force constants are shown in the following section with the symmetry prefactor (± 1). Created when `MODE = fitting`.

- `PREFIX`.xml

  An XML file containing the necessary information for performing phonon calculations. The files can be read by *anphon* using the `FCSXML`-tag. Created when `MODE = fitting`.

1.5.3 ALM: Theoretical background

Interatomic force constants (IFCs)

The starting point of the computational methodology is to approximate the potential energy of interacting atoms by a Taylor expansion with respect to atomic displacements by

$$\begin{aligned} U - U_0 &= \sum_{n=1}^{N} U_n = U_1 + U_2 + U_3 + \cdots, \\ U_n &= \frac{1}{n!} \sum_{\substack{\ell_1\kappa_1,\ldots,\ell_n\kappa_n \\ \mu_1,\ldots,\mu_n}} \Phi_{\mu_1\ldots\mu_n}(\ell_1\kappa_1;\ldots;\ell_n\kappa_n)\, u_{\mu_1}(\ell_1\kappa_1)\cdots u_{\mu_n}(\ell_n\kappa_n). \end{aligned} \tag{1.1}$$

Here, $u_\mu(\ell\kappa)$ is the atomic displacement of κth atom in the ℓth unit cell along μth direction, and $\Phi_{\mu_1\ldots\mu_n}(\ell_1\kappa_1;\ldots;\ell_n\kappa_n)$ is the nth-order interatomic force constant (IFC).

Symmetry relationship between IFCs

The are several relationships between IFCs which may be used to reduce the number of independence IFCs.

- Permutation

  IFC should be invariant under the exchange of the triplet (ℓ, κ, μ), e.g.,

  $$\Phi_{\mu_1\mu_2\mu_3}(\ell_1\kappa_1;\ell_2\kappa_2;\ell_3\kappa_3) = \Phi_{\mu_1\mu_3\mu_2}(\ell_1\kappa_1;\ell_3\kappa_3;\ell_2\kappa_2) = \ldots.$$

- Periodicity

  Since IFCs should depend on interatomic distances, they are invariant under a translation in units of the lattice vector, namely

  $$\Phi_{\mu_1\mu_2\ldots\mu_n}(\ell_1\kappa_1;\ell_2\kappa_2;\ldots;\ell_n\kappa_n) = \Phi_{\mu_1\mu_2\ldots\mu_n}(0\kappa_1;\ell_2-\ell_1\kappa_2;\ldots;\ell_n-\ell_1\kappa_n).$$

- Crystal symmetry

  A crystal symmetry operation maps an atom $\vec{r}(\ell\kappa)$ to another equivalent atom $\vec{r}(LK)$ by rotation and translation. Since the potential energy is invariant under any crystal symmetry operations, IFCs should transform under a symmetry operation as follows:

  $$\sum_{\nu_1,\ldots,\nu_n} \Phi_{\nu_1\ldots\nu_n}(L_1K_1;\ldots;L_nK_n) O_{\nu_1\mu_1}\cdots O_{\nu_n\mu_n} = \Phi_{\mu_1\ldots\mu_n}(\ell_1\kappa_1;\ldots;\ell_n\kappa_n), \tag{1.2}$$

  where O is the rotational matrix of the symmetry operation. Let N_s be the number of symmetry operations, there are N_s relationships between IFCs which may be used to find independent IFCs.

Note: In the implementation of ALM, symmetricaly independent IFCs are searched in Cartesian coordinate where the matrix element $O_{\mu\nu}$ is 0 or ±1 in all symmetry operations except for those of **hexagonal** (trigonal) lattice. Also, except for hexagonal (trigonal) systems, the product $O_{\nu_1\mu_1}\cdots O_{\nu_n\mu_n}$ in the left hand side of equation (1.2) becomes non-zero only for a specific pair of $\{\nu\}$ (and becomes 0 for all other $\{\nu\}$s). Therefore, let $\{\nu'\}$ be such a pair of $\{\nu\}$, the equation (1.2) can be reduced to

$$\Phi_{\nu'_1\ldots\nu'_n}(L_1K_1;\ldots;L_nK_n) = s\Phi_{\mu_1\ldots\mu_n}(\ell_1\kappa_1;\ldots;\ell_n\kappa_n), \tag{1.3}$$

where $s = \pm1$. The code employs equation (1.3) instead of equation (1.2) to reduce the number of IFCs. If IFCs of the left-hand side and the right-hand side of equation (1.3) are equivalent and the coupling

coefficient is $s = -1$, the IFC is removed since it becomes zero. For **hexagonal** (trigonal) systems, there can be symmetry operations where multiple terms in the left-hand side of equation (1.2) become non-zero. For such cases, equation (1.2) is not used to reduce the number of IFCs. Alternatively, the corresponding symmetry relationships are imposed as constraints between IFCs in solving fitting problems.

Constraints between IFCs

Since the potential energy is invariant under rigid translation and rotation, it may be necessary for IFCs to satisfy corresponding constraints.

The constraints for translational invariance are given by

$$\sum_{\ell_1\kappa_1} \Phi_{\mu_1\mu_2\ldots\mu_n}(\ell_1\kappa_1;\ell_2\kappa_2;\ldots;\ell_n\kappa_n) = 0, \tag{1.4}$$

which should be satisfied for arbitrary pairs of $\ell_2\kappa_2,\ldots,\ell_n\kappa_n$ and $\mu_1,\ldots,\mu_n$. The code **alm** imposes equation (1.4) by default (`ICONST = 1`).

The constraints for rotational invariance are

$$\begin{aligned}\sum_{\ell'\kappa'}&(\Phi_{\mu_1\ldots\mu_n\nu}(\ell_1\kappa_1;\ldots;\ell_n\kappa_n;\ell'\kappa')r_\mu(\ell'\kappa') - \Phi_{\mu_1\ldots\mu_n\mu}(\ell_1\kappa_1;\ldots;\ell_n\kappa_n;\ell'\kappa')r_\nu(\ell'\kappa'))\\ &+\sum_{\lambda=1}^{n}\sum_{\mu'_\lambda}\Phi_{\mu_1\ldots\mu'_\lambda\ldots\mu_n}(\ell_1\kappa_1;\ldots;\ell_\lambda\kappa_\lambda;\ldots;\ell_n\kappa_n)(\delta_{\mu,\mu_\lambda}\delta_{\nu,\mu'_\lambda}-\delta_{\nu,\mu_\lambda}\delta_{\mu,\mu'_\lambda}) = 0,\end{aligned}$$

which must be satisfied for arbitrary pairs of $(\ell_1\kappa_1,\ldots,\ell_n\kappa_n;\mu_1,\ldots,\mu_n;\mu,\nu)$. This is complicated since $(n+1)$th-order IFCs (first line) are related to nth-order IFCs (second line).

For example, the constraints for rotational invariance related to harmonic terms can be found as

$$\begin{aligned}\sum_{\ell_2\kappa_2}&(\Phi_{\mu_1\nu}(\ell_1\kappa_1;\ell_2\kappa_2)r_\mu(\ell_2\kappa_2) - \Phi_{\mu_1\mu}(\ell_1\kappa_1;\ell_2\kappa_2)r_\nu(\ell_2\kappa_2))\\ &+\Phi_\nu(\ell_1\kappa_1)\delta_{\mu,\mu_1} - \Phi_\mu(\ell_1\kappa_1)\delta_{\nu,\mu_1} = 0,\end{aligned}$$

and

$$\begin{aligned}\sum_{\ell_3\kappa_3}&(\Phi_{\mu_1\mu_2\nu}(\ell_1\kappa_1;\ell_2\kappa_2;\ell_3\kappa_3)r_\mu(\ell_3\kappa_3) - \Phi_{\mu_1\mu_2\mu}(\ell_1\kappa_1;\ell_2\kappa_2;\ell_3\kappa_3)r_\nu(\ell_3\kappa_3))\\ &+\Phi_{\nu\mu_2}(\ell_1\kappa_1;\ell_2\kappa_2)\delta_{\mu,\mu_1} - \Phi_{\mu\mu_2}(\ell_1\kappa_1;\ell_2\kappa_2)\delta_{\nu,\mu_1}\\ &+\Phi_{\mu_1\nu}(\ell_1\kappa_1;\ell_2\kappa_2)\delta_{\mu,\mu_2} - \Phi_{\mu_1\mu}(\ell_1\kappa_1;\ell_2\kappa_2)\delta_{\nu,\mu_2} = 0.\end{aligned}$$

When `NORDER = 1`, equation (1.5) will be considered if `ICONST = 2`, whereas equation (1.5) will be neglected. To further consider equation (1.5), please use `ICONST = 3`, though it may enforce a number of harmonic IFCs to be zero since cubic terms don't exist in harmonic calculations (`NORDER = 1`).

Estimate IFCs by linear regression

Basic notations

From the symmetrically independent set of IFCs and the constraints between them for satisfying the translational and/or rotational invariance, we can construct an irreducible set of IFCs $\{\Phi_i\}$. Let us denote a column vector comprising the N irreducible set of IFCs as $\boldsymbol{\Phi}$. Then, the Taylor expansion potential (TEP) defined by equation (1.1) is written as

$$U_{\mathrm{TEP}} = \boldsymbol{b}^T\boldsymbol{\Phi}.$$

Here, $\boldsymbol{b} \in \mathbb{R}^{1\times N}$ is a function of atomic displacements $\{u_i\}$ defined as $\boldsymbol{b} = \partial U/\partial \boldsymbol{\Phi}$. The atomic forces based on the TEP is then given as

$$\boldsymbol{F}_{\rm TEP} = -\frac{\partial U_{\rm TEP}}{\partial \boldsymbol{u}} = -\frac{\partial \boldsymbol{b}^T}{\partial \boldsymbol{u}}\boldsymbol{\Phi} = A\boldsymbol{\Phi}, \tag{1.5}$$

where $A \in \mathbb{R}^{3N_s\times N}$ with N_s being the number of atoms in the supercell, and $\boldsymbol{u}^T = (u_1^x, u_1^y, u_1^z, \ldots, u_{N_s}^x, u_{N_s}^y, u_{N_s}^z)$ is the vector comprising $3N_s$ atomic displacements in the supercell. Note that the matrix A and force vector $\boldsymbol{F}_{\rm TEP}$ depend on the atomic configuration of the supercell. To make this point clearer, let us denote them as $A(\boldsymbol{u})$ and $\boldsymbol{F}_{\rm TEP}(\boldsymbol{u})$.

To estimate the IFC vector $\boldsymbol{\Phi}$ by linear regression, it is usually necessary to consider several different displacement patterns. Let us suppose we have N_d displacement patterns and atomic forces for each pattern obtained by DFT. Then, equation (1.5) defined for each displacement pattern can be combined to a single equation as

$$\boldsymbol{F}_{\rm TEP} = \mathbb{A}\boldsymbol{\Phi},$$

where $\boldsymbol{F}^T = [\boldsymbol{F}^T(\boldsymbol{u}_1), \ldots, \boldsymbol{F}^T(\boldsymbol{u}_{N_d})]$ and $\mathbb{A}^T = [A^T(\boldsymbol{u}_1), \ldots, A^T(\boldsymbol{u}_{N_d})]$.

Ordinary least-squares

In the ordinary least-squares (`LMODEL = least-squares`), IFCs are estimated by solving the following problem:

$$\boldsymbol{\Phi}_{\rm OLS} = \underset{\boldsymbol{\Phi}}{\rm argmin}\, \frac{1}{2N_d}\|\boldsymbol{F}_{\rm DFT} - \boldsymbol{F}_{\rm TEP}\|_2^2 = \underset{\boldsymbol{\Phi}}{\rm argmin}\, \frac{1}{2N_d}\|\boldsymbol{F}_{\rm DFT} - \mathbb{A}\boldsymbol{\Phi}\|_2^2. \tag{1.6}$$

Therefore, the IFCs are determined so that the residual sum of squares (RSS) is minimized. To determine all elements of $\boldsymbol{\Phi}_{\rm OLS}$ uniquely, $\mathbb{A}^T\mathbb{A}$ must be full rank. When the fitting is successful, **alm** reports the relative fitting error σ defined by

$$\sigma = \sqrt{\frac{\|\boldsymbol{F}_{\rm DFT} - \mathbb{A}\boldsymbol{\Phi}\|_2^2}{\|\boldsymbol{F}_{\rm DFT}\|_2^2}}, \tag{1.7}$$

where the denominator is the square sum of the DFT forces.

Elastic-net regression

In the elasitc-net optimization (`LMODEL = elastic-net`), IFCs are estimated by solving the following optimization problem:

$$\boldsymbol{\Phi}_{\rm enet} = \underset{\boldsymbol{\Phi}}{\rm argmin}\, \frac{1}{2N_d}\|\boldsymbol{F}_{\rm DFT} - \mathbb{A}\boldsymbol{\Phi}\|_2^2 + \alpha\beta\|\boldsymbol{\Phi}\|_1 + \frac{1}{2}\alpha(1-\beta)\|\boldsymbol{\Phi}\|_2^2, \tag{1.8}$$

where α is a hyperparameter that controls the trade-off between the sparsity and accuracy of the model, and β ($0 < \beta \leq 1$) is a hyperparameter that controls the ratio of the L_1 and L_2 regularization terms. α and β must be given by input tags `L1_ALPHA` and `L1_RATIO`, respectively.

An optimal value of α can be estimated, for example, by cross-validation (CV). A n-fold CV can be performed by setting the `CV`-tag properly.

1.6 ANPHON: Anharmonic phonon calculator

1.6.1 ANPHON: Input files

Format of input files

Each input file should consist of entry fields. Available entry fields are

&general, **&cell**, **&analysis**, and **&kpoint**.

The format of the input file is the same as that of *alm* which can be found *here*.

List of supported input variables

&general				
BCONNECT	*BORNINFO*	*BORNSYM*	*CLASSICAL*	*EMIN*
EPSILON	*FC2XML*	*FCSXML*	*ISMEAR*	*KD*
MASS	*MODE*	*NA_SIGMA*	*NKD*	*NONANALYTIC*
PREFIX	*PRINTSYM*	*RESTART*	*TMIN*	*TOLERANCE*
TRISYM				
&scph				
IALGO	*KMESH_INTERPOLATE*	*KMESH_SCPH*	*LOWER_TEMP*	*MAXITER*
MIXALPHA	*RESTART_SCPH*	*SELF_OFFDIAG*	*TOL_SCPH*	*WARMSTART*
&analysis				
ANIME	*ANIME_CELLSIZE*	*GRUNEISEN*	*ISOFACT*	
ISOTOPE	*KAPPA_SPEC*	*PDOS*	*PRINTEVEC*	*PRINTMSD*
PRINTPR	*PRINTVEL*	*PRINTXSF*	*SPS*	*TDOS*

Description of input variables

"&general"-field

- **PREFIX**-tag : Job prefix to be used for names of output files

  Default None

  Type String

- **MODE**-tag = phonons | RTA

phonons	Calculate phonon dispersion relation, phonon DOS, Grüneisen parameters etc.
RTA	Calculate phonon lifetimes and lattice thermal conductivity based on the Boltzmann transport equation (BTE) with the relaxation time approximation (RTA).
SCPH	Calculate temperature dependent phonon dispersion curves by the self-consistent phonon method.

Default None

Type String

- **NKD**-tag : Number of atomic species

  Default None

  Type Integer

- **KD**-tag = Name[1], . . . , Name[`NKD`]

  Default None

  Type Array of strings

  Example In the case of GaAs with `NKD = 2`, it should be `KD = Ga As`.

- MASS-tag = mass[1], . . . , mass[`NKD`]

  Default Standard atomic weight of elements given by the `KD`-tag

  Type Array of double

  Example In the case of Bi_2Te_3 with `NKD = 2`, `MASS` should be `MASS = 208.98 127.60`.

- **FCSXML**-tag : XML file containing force constants generated by the program *alm*

  Default None

  Type String

- FC2XML-tag : XML file containing harmonic force constants for different size of supercell

  Default None

  Type String

  Description When `FC2XML` is given, the harmonic force constants in this file are used for calculating dynamical matrices. It is possible to use different size of supercell for harmonic and anharmonic terms, which are specified by `FC2XML` and `FCSXML` respectively.

- TOLERANCE-tag : Tolerance for finding symmetry operations

  Default 1.0e-6

  Type Double

- PRINTSYM-tag = 0 | 1

0	Symmetry operations won't be saved in "SYMM_INFO_PRIM"
1	Symmetry operations will be saved in "SYMM_INFO_PRIM"

  Default 0

  type Integer

- NONANALYTIC-tag = 0 | 1 | 2 | 3

0	Non-analytic correction is not considered.
1	Include the non-analytic correction by the damping method proposed by Parlinski.
2	Include the non-analytic correction by the mixed-space approach
3	Include the non-analytic correction by the Ewald method

  Default 0

  Type Integer

  Description When `NONANALYTIC > 0`, appropriate `NA_SIGMA` and `BORNINFO` have to be given.

- NA_SIGMA-tag : Damping factor for the non-analytic term

  Default 0.0

  Type Double

  Description Used when `NONANALYTIC = 1`. The definition of `NA_SIGMA` is described in the formalism section.

- BORNINFO-tag : File containing the macroscopic dielectric tensor and Born effective charges for the non-analytic correction

  Default None

  Type String

  Description The details of the file format can be found *here*.

- BORNSYM-tag = 0 | 1

0	Do not symmetrize Born effective charges
1	Symmetrize Born effective charges by using point group symmetry

  Default 0

  Type Integer

- TMIN, TMAX, DT-tags : Temperature range and its stride in units of Kelvin

  Default `TMIN = 0`, `TMAX = 1000`, `DT = 10`

  Type Double

- EMIN, EMAX, DELTA_E-tags : Energy range and its stride in units of kayser (cm^{-1})

  Default `EMIN = 0`, `EMAX = 1000`, `DELTA_E = 10`

  Type Double

- ISMEAR-tag = -1 | 0 | 1

-1	Tetrahedron method
0	Lorentzian smearing with width of `EPSILON`
1	Gaussian smearing with width of `EPSILON`

  Default -1

  Type Integer

  Description `ISMEAR` specifies the method for Brillouin zone integration

---

- EPSILON-tag : Smearing width in units of Kayser (cm^{-1})

  Default 10.0

Type Double

Description This variable is neglected when `ISMEAR = -1`

- BCONNECT-tag = 0 | 1 | 2

0	Phonon band is saved without change (sorted in order of energy)
1	Phonon band is connected by using the similarity of eigenvectors.
2	Same as `BCONNECT=1`. In addition, information about the connectivity is saved as `PREFIX.connection`.

Default 0

Type Integer

Description The algorithm for connecting a band structure is described here.

- CLASSICAL-tag = 0 | 1

0	Use quantum statistics (default)
1	Use classical statistics

Default 0

Type Integer

Description When `CLASSICAL = 1`, all thermodynamic functions including the occupation function, heat capacity, and mean square displacements are calculated using the classical formulae. This option may be useful when comparing the lattice dynamics and molecular dynamics results.

- TRISYM-tag : Flag to use symmetry operations to reduce the number of triples of k points for self-energy calculations

0	Symmetry will not be used
1	Use symmetry to reduce triples of k points

Default 1

Type Integer

Description This variable is used only when `MODE = RTA`.

Note: `TRISYM = 1` can reduce the computational cost, but phonon linewidth stored to the file `PREFIX`.result needs to be averaged at points of degeneracy. For that purpose, a subsidiary program *analyze_phonons.py** should be used.

- RESTART-tag : Flag to restart the calculation when `MODE = RTA`

0	Calculate from scratch
1	Restart from the existing file

Default 1 if there is a file named `PREFIX`.result; 0 otherwise

Type Integer

"&scph"-field (Read only when `MODE = SCPH`)

- KMESH_INTERPOLATE-tag = k1, k2, k3

Default None

Type Array of integers

Description In the iteration process of the SCPH equation, the interpolation is done using the k mesh defined by `KMESH_INTERPOLATE`.

- KMESH_SCPH-tag = k1, k2, k3

Default None

Type Array of integers

Description This k mesh is used for the inner loop of the SCPH equation. Each value of `KMESH_SCPH` must be equal to or a multiple of the number of `KMESH_INTERPOLATE` in the same direction.

- SELF_OFFDIAG-tag = 0 | 1

0	Neglect the off-diagonal elements of the loop diagram in the SCPH calculation
1	Consider the off-diagonal elements of the loop diagram in the SCPH calculation

Default 0

Type Integer

Description `SELF_OFFDIAG = 1` is more accurate, but expensive.

- TOL_SCPH-tag: Stopping criterion of the SCPH iteration

Default 1.0e-10

Type Double

Description The SCPH iteration stops when both $[\frac{1}{N_q}\sum_q(\Omega_q^{(i)} - \Omega_q^{(i-1)})^2]^{1/2} <$ `TOL_SCPH` and $(\Omega_q^{(i)})^2 \geq 0$ $(\forall q)$ are satisfied. Here, $\Omega_q^{(i)}$ is the anharmonic phonon frequency in the ith iteration and q is the phonon modes at the irreducible momentum grid of `KMESH_INTERPOLATE`.

- MIXALPHA-tag: Mixing parameter used in the SCPH iteration

  Default 0.1

  Type Double

- MAXITER-tag: Maximum number of the SCPH iteration

  Default 1000

  Type Integer

- LOWER_TEMP-tag = 0 | 1

0	The SCPH iteration start from `TMIN` to `TMAX`. (Raise the temperature)
1	The SCPH iteration start from `TMAX` to `TMIN`. (Lower the temperature)

  Default 1

  Type Integer

- WARMSTART-tag = 0 | 1

0	SCPH iteration is initialized by harmonic frequencies and eigenvectors
1	SCPH iteration is initialized by the solution of the previous temperature

  Default 1

  Type Integer

  Description `WARMSTART = 1` usually improves the convergence.

- IALGO-tag = 0 | 1

0	MPI parallelization for the k point
1	MPI parallelization for the phonon branch

  Default 0

  Type Integer

  Description Use `IALGO = 1` when the primitive cell contains many atoms and the number of k points is small.

- RESTART_SCPH-tag = 0 | 1

0	Perform a SCPH calculation from scratch
1	Skip a SCPH iteration by loading a precalculated result

Default 1 if the file `PREFIX.scph_dymat` exists in the working directory; 0 otherwise

Type Integer

"&cell"-field

Please specify the cell parameters of the *primitive cell* as:

```
&cell
 a
 a11 a12 a13
 a21 a22 a23
 a31 a32 a33
/
```

The cell parameters are then given by $\vec{a}_1 = a \times (a_{11}, a_{12}, a_{13})$, $\vec{a}_2 = a \times (a_{21}, a_{22}, a_{23})$, and $\vec{a}_3 = a \times (a_{31}, a_{32}, a_{33})$.

Note: The lattice constant a must be consistent with the value used for the program *alm*. For example, if one used $a = 20.4a_0$ for a 2x2x2 supercell of Si, one should use $a = 10.2a_0$ here for the primitive cell.

"&kpoint"-field

This entry field is used to specify the list of k points to be calculated. The first entry **KPMODE** specifies the types of calculation which is followed by detailed entries.

- **KPMODE = 0** : Calculate phonon frequencies at given k points

  For example, if one wants to calculate phonon frequencies at Gamma (0, 0, 0) and X (0, 1/2, 1/2) of an FCC crystal, the `&kpoint` entry should be written as

  ```
  &kpoint
   0
   0.000 0.000 0.000
   0.000 0.500 0.500
  /
  ```

- **KPMODE = 1** : Band dispersion calculation

  For example, if one wants to calculate phonon dispersion relations along G-K-X-G-L of a FCC crystal, the `&kpoint` entry should be written as follows:

  ```
  &kpoint
   1
   G 0.000 0.000 0.000  K 0.375 0.375 0.750 51
   K 0.375 0.375 0.750  X 0.500 0.500 1.000 51
   X 0.000 0.500 0.500  G 0.000 0.000 0.000 51
   G 0.000 0.000 0.000  L 0.500 0.500 0.500 51
  /
  ```

The 1st and 5th columns specify the character of Brillouin zone edges, which are followed by fractional coordinates of each point. The last column indicates the number of sampling points.

- **KPMODE = 2** : Uniform k grid for phonon DOS and thermal conductivity

  In order to perform a calculation with 20x20x20 k grid, the entry should be

```
&kpoint
 2
 20 20 20
/
```

"&analysis"-field

- GRUNEISEN-tag = 0 | 1

0	Grüneisen parameters will not be calculated
1	Grüneisen parameters will be stored

Default 0

Type Integer

Description When `MODE = phonons` and `GRUNEISEN = 1`, Grüneisen parameters will be stored in `PREFIX`.gru (*KPMODE* = 1) or `PREFIX`.gru_all (*KPMODE* = 2).

Note: To compute Grüneisen parameters, cubic force constants must be contained in the `FCSXML` file.

- PRINTEVEC-tag = 0 | 1

0	Do not print phonon eigenvectors
1	Print phonon eigenvectors in the `PREFIX`.evec file

Default 0

Type Integer

- PRINTXSF-tag = 0 | 1

0	Do not save an AXSF file
1	Create an AXSF file `PREFIX`.axsf

Default 0

Type Integer

Description This is to visualize the direction of vibrational modes at gamma (0, 0, 0) by XCrySDen. This option is valid only when `MODE = phonons`.

- PRINTVEL-tag = 0 | 1

0	Do not print group velocity
1	Store phonon velocities to a file

Default 0

Type Integer

Description When `MODE = phonons` and `PRINTVEL = 1`, group velocities of phonons will be stored in `PREFIX`.phvel (*KPMODE* = 1) or `PREFIX`.phvel_all (*KPMODE* = 2).

- PRINTMSD-tag = 0 | 1

0	Do not print mean-square-displacement (MSD) of atoms
1	Save MSD of atoms to the file `PREFIX`.mds

Default 0

Type Integer

Description This flag is available only when `MODE = phonons` and *KPMODE* = 2.

- PDOS-tag = 0 | 1

0	Only the total DOS will be printed in `PREFIX`.dos
1	Atom-projected phonon DOS will be stored in `PREFIX`.dos

Default 0

Type Integer

Description This flag is available only when `MODE = phonons` and *KPMODE* = 2.

- TDOS-tag = 0 | 1

0	Do not compute two-phonon DOS
1	Two-phonon DOSs will be stored in `PREFIX`.tdos

Default 0

Type Integer

Description This flag is available only when `MODE = phonons` and *KPMODE* = 2.

Note: Calculation of two-phonon DOS is computationally expensive.

- SPS-tag = 0 | 1 | 2

0	Do not compute scattering phase space
1	Total and mode-decomposed scattering phase space involving the three-phonon processes will be stored in PREFIX.sps
2	Three-phonon scattering phase space with the Bose factor will be stored in PREFIX.sps_Bose

Default 0

Type Integer

Description This flag is available only when `MODE = phonons` and *KPMODE* = 2.

- PRINTPR-tag = 0 | 1

0	Do not compute the (atomic) participation ratio
1	Compute participation ratio and atomic participation ratio, which will be stored in PREFIX.pr and PREFIX.apr respectively.

Default 0

Type Integer

Description This flag is available when `MODE = phonons`.

- KAPPA_SPEC-tag = 0 | 1

0	Do not compute the thermal conductivity spectra
1	Compute the thermal conductivity spectra, which will be stored in PREFIX.kappa_spec .

Default 0

Type Integer

Description This flag is available when `MODE = RTA`.

- ISOTOPE-tag = 0 | 1

0	Do not consider phonon-isotope scatterings
1	Consider phonon-isotope scatterings
2	Consider phonon-isotope scatterings as in `ISOTOPE = 1` and the calculated selfenergy is stored in `PREFIX`.gamma_isotope

Default 0

Type Integer

Description When `MODE = RTA` and `ISOTOPE = 1 or 2`, phonon scatterings due to isotopes will be considered perturbatively. `ISOFACT` should be properly given.

- ISOFACT-tag = isofact[1], ... , isofact[`NKD`]

  Default Automatically calculated from the `KD` tag

  Type Array of doubles

  Description Isotope factor is a dimensionless value defined by $\sum_i f_i(1 - m_i/\bar{m})^2$. Here, f_i is the fraction of the ith isotope of an element having mass m_i, and $\bar{m} = \sum_i f_i m_i$ is the average mass, respectively. This quantity is equivalent to g_2 appearing in the original paper by S. Tamura [Phys. Rev. B, 27, 858.].

- ANIME-tag = k1, k2, k3

  Default None

  Type Array of doubles

  Description This tag is to animate vibrational mode. k1, k2, and k3 specify the momentum of phonon modes to animate, which should be given in units of the reciprocal lattice vector. For example, `ANIME = 0.0 0.0 0.5` will animate phonon modes at (0, 0, 1/2). When `ANIME` is given, `ANIME_CELLSIZE` is also necessary. You can choose the format of animation files, either AXSF or XYZ, by `ANIME_FORMAT` tag.

- ANIME_CELLSIZE-tag = L1, L2, L3

  Default None

  Type Array of integers

  Description This tag specifies the cell size for animation. L1, L2, and L3 should be large enough to be commensurate with the reciprocal point given by the `ANIME` tag.

- ANIME_FORMAT = xsf | xyz

  Default xyz

  Type String

Description When `ANIME_FORMAT = xsf`, `PREFIX`.anime???.axsf files are created for XcrySDen. When `ANIME_FORMAT = xyz`, `PREFIX`.anime???.xyz files are created for VMD (and any other supporting software such as Jmol).

Format of BORNINFO

When one wants to consider the LO-TO splitting near the Γ point, it is necessary to set `NONANALYTIC = 1` and provide `BORNINFO` file containing dielectric tensor ϵ^{∞} and Born effective charge Z^*. In `BORNINFO` file, the dielectric tensor should be written in first 3 lines which are followed by Born effective charge tensors for each atom as the following.

$$
\begin{array}{ccc}
\epsilon^{\infty}_{xx} & \epsilon^{\infty}_{xy} & \epsilon^{\infty}_{xz} \\
\epsilon^{\infty}_{yx} & \epsilon^{\infty}_{yy} & \epsilon^{\infty}_{yz} \\
\epsilon^{\infty}_{zx} & \epsilon^{\infty}_{zy} & \epsilon^{\infty}_{zz} \\
Z^*_{1,xx} & Z^*_{1,xy} & Z^*_{1,xz} \\
Z^*_{1,yx} & Z^*_{1,yy} & Z^*_{1,zz} \\
Z^*_{1,zx} & Z^*_{1,zy} & Z^*_{1,zz} \\
 & \vdots & \\
Z^*_{N_p,xx} & Z^*_{N_p,xy} & Z^*_{N_p,xz} \\
Z^*_{N_p,yx} & Z^*_{N_p,yy} & Z^*_{N_p,zz} \\
Z^*_{N_p,zx} & Z^*_{N_p,zy} & Z^*_{N_p,zz}
\end{array}
$$

Here, N_p is the number of atoms contained in the *primitive cell*.

Attention: Please pay attention to the order of Born effective charges.

1.6.2 ANPHON: Output files

- `PREFIX`.bands

  Phonon dispersion along given k paths in units of cm $^{-1}$. Created when `MODE = phonons` with **KPMODE** = 1.

- `PREFIX`.dos

  Phonon density of states (DOS). Atom projected phonon DOSs are also printed when `PDOS = 1`. Created when `MODE = phonons` with **KPMODE** = 2.

- `PREFIX`.tdos

  Two-phonon density of states for all irreducible :math:'k' points. Created when `MODE = phonons` with **KPMODE** = 2 and `TDOS = 1`.

- `PREFIX`.thermo

  Constant volume heat capacity, vibrational entropy, internal energy, and vibrational free energy. Created when `MODE = phonons` with **KPMODE** = 2. When `FE_BUBBLE = 1` is set in the **&analysis** field, an additional bubble correction term to the vibrational free energy is also calculated.

- PREFIX.msd

  Mean-square-displacements of atoms. Created when MODE = phonons with **KPMODE** = 2 and PRINTMSD = 1.

- PREFIX.sps

  Total and mode-decomposed scattering phase space. Created when MODE = phonons with **KPMODE** = 2 and SPS = 1.

- PREFIX.pr

  Participation ratio of every phonon modes. Created when MODE = phonons and PRINTPR = 1.

- PREFIX.apr

  Atomic participation ratio of every phonon modes. Created when MODE = phonons and PRINTPR = 1.

- PREFIX.phvel

  Phonon group velocity along given k paths. Created when MODE = phonons with **KPMODE** = 1 and PRINTVEL = 1.

- PREFIX.phvel_all

  Magnitude of group velocity $|\boldsymbol{v}|$ of all phonon modes at the uniform k grid. Created when MODE = phonons with **KPMODE** = 2 and PRINTVEL = 1.

- PREFIX.evec

  Eigenvalues and eigenvectors of dynamical matrices. Eigenvalues are printed in Rydberg atomic units. Created when MODE = phonons with PRINTEVEC = 1.

- PREFIX.gru

  Grüneisen parameters along given k paths. Created when MODE = phonons with **KPMODE** = 1 and GRUNEISEN = 1.

- PREFIX.gru_all

  Grüneisen parameters of all phonon modes at the uniform k grid. Created when MODE = phonons with **KPMODE** = 2 and GRUNEISEN = 1.

- PREFIX.axsf

  Zone-center phonon modes with directions indicated by arrows. This file can be visualized by XcrySDen. Created when MODE = phonons with PRINTXSF = 1.

- PREFIX.anime???.axsf and PREFIX.anime???.xyz

  Files for animating phonon modes. ??? is the mode number. Created when MODE = phonons with a proper ANIME-tag. If ANIME_FORMAT = xsf, axsf files will be created which can be displayed by XcrySDen. If ANIME_FORMAT = xyz, xyz files will be created which can be visualized by VMD, Jmol, etc.

- PREFIX.result

  In this file, phonon frequency, group velocity, and anharmonic phonon linewidths are printed. This file is updated during thermal conductivity calculations (MODE = RTA). In addition, this file is read when the restart mode is turned on (RESTART = 1).

- PREFIX.kl

  Lattice thermal conductivity tensor. Created when MODE = RTA.

- `PREFIX`.kl_spec

  Spectra of lattice thermal conductivity. Only diagonal components are saved. Created when `MODE = RTA` and `KAPPA_SPEC = 1`.

- `PREFIX`.gamma_isotope

  Phonon selfenergy due to isotope scatterings calculated by the Tamura's formula. Created when `MODE = RTA` and `ISOTOPE = 2`.

- `PREFIX`.scph_dymat

  Anharmonic dynamical matrix calculated on the k grid defined by the `KMESH_INTERPOLATE` tag. This file is used to restart the SCPH calculation.

- `PREFIX`.scph_bands

  Anharmonic phonon dispersion curves.

- `PREFIX`.scph_dos

  Anharmonic phonon DOS. Created when `MODE = SCPH` and `DOS = 1` with **KPMODE** = 2.

- `PREFIX`.scph_thermo

  Constant volume heat capacity, vibrational entropy, and vibrational free energy calculated based on the self-consistent phonon calculation. Created when `MODE = SCPH` with **KPMODE** = 2.

- `PREFIX`.scph_msd

  Mean square displacement calculated within the SCPH theory. Created when `MODE = SCPH` and `PRINTMSD = 1` with **KPMODE** = 2.

- `PREFIX`.scph_dfc2

  This file contains $\Delta D(\boldsymbol{q}) = D_{\mathrm{SCPH}}(\boldsymbol{q}) - D_{\mathrm{Harmonic}}(\boldsymbol{q})$. For the definition, see the *formalism of the SCPH calculation*.

1.6.3 ANPHON: Theoretical background

Dynamical matrix

The dynamical matrix is given by

$$D_{\mu\nu}(\kappa\kappa';\boldsymbol{q}) = \frac{1}{\sqrt{M_\kappa M_{\kappa'}}} \sum_{\ell'} \Phi_{\mu\nu}(\ell\kappa;\ell'\kappa') \exp\left[i\boldsymbol{q}\cdot(\boldsymbol{r}(\ell') - \boldsymbol{r}(\ell))\right], \tag{1.9}$$

where M_κ is the atomic mass of atom κ. By diagonalizing the dynamical matrix, one can obtain $m\ (= 3N_\kappa)$ eigenvalues $\omega^2_{\boldsymbol{q}j}$ $(j = 1, 2, \ldots, m)$ and corresponding eigenvectors $\boldsymbol{e}_{\boldsymbol{q}j}$ for each $\boldsymbol{q}$ point. Here, $\boldsymbol{e}_{\boldsymbol{q}j}$ is a column vector consisting of atomic polarization $e_\mu(\kappa;\boldsymbol{q}j)$. Let $D(\boldsymbol{q})$ denote a matrix form of equation (1.9), the eigenvalues may be written as

$$\omega^2_{\boldsymbol{q}j} = (\boldsymbol{e}^*_{\boldsymbol{q}j})^{\mathrm{T}} D(\boldsymbol{q}) \boldsymbol{e}_{\boldsymbol{q}j}. \tag{1.10}$$

Next, we introduce $m \times m$ matrices Λ and W which are defined as $\Lambda(\boldsymbol{q}) = \mathrm{diag}(\omega^2_{\boldsymbol{q}1}, \ldots, \omega^2_{\boldsymbol{q}m})$ and $W(\boldsymbol{q}) = (\boldsymbol{e}_{\boldsymbol{q}1}, \ldots, \boldsymbol{e}_{\boldsymbol{q}m})$, respectively. Then, equation (1.10) can be denoted as

$$\Lambda(\boldsymbol{q}) = W^\dagger(\boldsymbol{q}) D(\boldsymbol{q}) W(\boldsymbol{q}).$$

When one needs to capture the LO-TO splitting near the zone-center by the supercell approach, it is necessary to add the non-analytic part of the dynamical matrix defined by

$$D^{\rm NA}_{\mu\nu}(\kappa\kappa';\boldsymbol{q}) = \frac{1}{\sqrt{M_\kappa M_{\kappa'}}}\frac{4\pi e^2}{\Omega}\frac{(Z^*_\kappa \boldsymbol{q})_\mu (Z^*_{\kappa'}\boldsymbol{q})_\nu}{\boldsymbol{q}\cdot\epsilon^\infty\boldsymbol{q}},$$

where Ω is the volume of the primitive cell, Z^*_κ is the Born effective charge tensor of atom κ, and ϵ^∞ is the dielectric constant tensor, respectively. In program *anphon*, either the Parlinski's way[1] or the mixed-space approach[2] can be used. In the Parlinski's approach (`NONANALYTIC = 1`), the total dynamical matrix is given by

$$D(\boldsymbol{q}) + D^{\rm NA}(\boldsymbol{q})\exp\left(-q^2/\sigma^2\right),$$

where σ is a damping factor. σ must be chosen carefully so that the non-analytic contribution becomes negligible at Brillouin zone boundaries. In the mixed-space approach (`NONANALYTIC = 2`), the total dynamical matrix is given by

$$D(\boldsymbol{q}) + D^{\rm NA}(\boldsymbol{q})\frac{1}{N}\sum_{\ell'}\exp\left[i\boldsymbol{q}\cdot(\boldsymbol{r}(\ell') - \boldsymbol{r}(\ell))\right].$$

The second term vanishes at commensurate $\boldsymbol{q}$ points other than Γ point ($\boldsymbol{q}=0$).

To include the non-analytic term, one needs to set `NONANALYTIC > 0` and give appropriate `BORNINFO` and `NA_SIGMA` tags.

Group velocity

The group velocity of phonon mode $\boldsymbol{q}j$ is given by

$$\boldsymbol{v}_{\boldsymbol{q}j} = \frac{\partial\omega_{\boldsymbol{q}j}}{\partial\boldsymbol{q}}.$$

To evaluate the group velocity numerically, we employ a central difference where $\boldsymbol{v}$ may approximately be given by

$$\boldsymbol{v}_{\boldsymbol{q}j} \approx \frac{\omega_{\boldsymbol{q}+\Delta\boldsymbol{q}j} - \omega_{\boldsymbol{q}-\Delta\boldsymbol{q}j}}{2\Delta\boldsymbol{q}}.$$

If one needs to save the group velocities, please turn on the `PRINTVEL`-tag.

Thermodynamics functions

The specific heat at constant volume $C_{\rm v}$, the internal energy U, the vibrational entropy S, and the Helmholtz free energy F of individual harmonic oscillator are given as follows:

$$\begin{aligned}
U &= \frac{1}{N_q}\sum_{\boldsymbol{q},j}\hbar\omega_{\boldsymbol{q}j}\left[\frac{1}{e^{\hbar\omega_{\boldsymbol{q}j}/kT}-1}+\frac{1}{2}\right],\\
C_{\rm v} &= \frac{k}{N_q}\sum_{\boldsymbol{q},j}\left(\frac{\hbar\omega_{\boldsymbol{q}j}}{2kT}\right)^2\mathrm{cosech}^2\left(\frac{\hbar\omega_{\boldsymbol{q}j}}{2kT}\right),\\
S &= \frac{k}{N_q}\sum_{\boldsymbol{q},j}\left[\frac{\hbar\omega_{\boldsymbol{q}j}}{kT}\frac{1}{e^{\hbar\omega_{\boldsymbol{q}j}/kT}-1} - \log\left(1-e^{-\hbar\omega_{\boldsymbol{q}j}/kT}\right)\right],\\
F &= \frac{1}{N_q}\sum_{\boldsymbol{q},j}\left[\frac{\hbar\omega_{\boldsymbol{q}j}}{2} + kT\log\left(1-e^{-\hbar\omega_{\boldsymbol{q}j}/kT}\right)\right].
\end{aligned}$$

[1] K. Parlinski, Z. Q. Li, and Y. Kawazoe, Phys. Rev. Lett. **81**, 3298 (1998).
[2] Y. Wang *et al.*, J. Phys.: Condens. Matter **22**, 202201 (2010).

Here, k is the Boltzmann constant. These quantities are saved in the PREFIX.thermo file.

When the self-consistent phonon mode (MODE = SCPH) is selected, the anharmonic free-energy defined by the following equation will be calculated and saved in the PREFIX.scph_thermo file:

$$F^{\rm SCP} = \frac{1}{N_q}\sum_{\boldsymbol{q},j}\left[\frac{\hbar\Omega_{\boldsymbol{q}j}}{2} + kT\log\left(1-e^{-\hbar\Omega_{\boldsymbol{q}j}/kT}\right)\right] - \frac{1}{N_q}\sum_{\boldsymbol{q},j}\left[\Omega^2_{\boldsymbol{q}j} - (C^\dagger_{\boldsymbol{q}}\Lambda^{(\rm HA)}_{\boldsymbol{q}}C_{\boldsymbol{q}})_{jj}\right]\times\frac{\hbar[1+2n_{\boldsymbol{q}j}]}{2\Omega_{\boldsymbol{q}j}}.$$

Details of the derivation of the above expression can be found in Ref.[7].

Mean square displacement

The mean square displacement tensor of atom κ is given by

$$\langle u_\mu(\kappa)u_\nu(\kappa)\rangle = \frac{\hbar}{2M_\kappa N_q}\sum_{\boldsymbol{q},j}\frac{1}{2\omega_{\boldsymbol{q}j}}\left(e_\mu(\kappa;\boldsymbol{q}j)e^*_\nu(\kappa;\boldsymbol{q}j) + e^*_\mu(\kappa;\boldsymbol{q}j)e_\nu(\kappa;\boldsymbol{q}j)\right) \times\coth\left(\frac{\hbar\omega_{\boldsymbol{q}j}}{2kT}\right). \tag{1.11}$$

When PRINTMSD is turned on, the code print the diagonal part of the mean square displacement tensor

$$\left\langle u^2_\mu(\kappa)\right\rangle = \frac{\hbar}{M_\kappa N_q}\sum_{\boldsymbol{q},j}\frac{1}{\omega_{\boldsymbol{q}j}}|e_\mu(\kappa;\boldsymbol{q}j)|^2\left(n_{\boldsymbol{q}j}+\frac{1}{2}\right),$$

where $n_{\boldsymbol{q}j} = 1/(e^{\hbar\omega_{\boldsymbol{q}j}/kT}-1)$ is the Bose-Einstein distribution function.

Phonon DOS

When *KPMODE* = 2, the program *anphon* saves the (one) phonon density of states (DOS) to the file PREFIX.dos. The one-phonon DOS is given by

$$\mathrm{DOS}(\omega) = \frac{1}{N_q}\sum_{\boldsymbol{q},j}\delta(\omega-\omega_{\boldsymbol{q}j}).$$

If PDOS = 1 is given, the program also prints the atom-projected phonon DOS which is given by

$$\mathrm{PDOS}(\kappa;\omega) = \frac{1}{N_q}\sum_{\boldsymbol{q},j}|\boldsymbol{e}(\kappa;\boldsymbol{q}j)|^2\delta(\omega-\omega_{\boldsymbol{q}j}).$$

In addition, TDOS-tag is available to compute the two-phonon DOS defined by

$$\mathrm{DOS2}(\omega;\boldsymbol{q};\pm) = \frac{1}{N_q}\sum_{\boldsymbol{q}_1,\boldsymbol{q}_2,j_1,j_2}\delta(\omega\pm\omega_{\boldsymbol{q}_1j_1}-\omega_{\boldsymbol{q}_2j_2})\delta_{\boldsymbol{q}\pm\boldsymbol{q}_1,\boldsymbol{q}_2+\boldsymbol{G}},$$

where $\boldsymbol{G}$ is a reciprocal lattice vector. The sign $\pm$ correspond to absorption and emission processes, respectively. Please note that the computation of the two-phonon DOS can be expensive especially when N_q or N_κ is large.

[7] Y.Oba, T. Tadano, R. Akashi, and S. Tsuneyuki, Phys. Rev. Materials **3**, 033601 (2019).

(Atomic) participation ratio

Participation ratio (PR) and atomic participation ratio (APR) defined in the following may be useful to analyze the localized nature of the phonon mode $\boldsymbol{q}j$.

- Participation ratio (PR)

$$PR_{\boldsymbol{q}j} = \left(\sum_{\kappa}^{N_\kappa} \frac{|\boldsymbol{e}(\kappa;\boldsymbol{q}j)|^2}{M_\kappa} \right)^2 \Bigg/ N_\kappa \sum_{\kappa}^{N_\kappa} \frac{|\boldsymbol{e}(\kappa;\boldsymbol{q}j)|^4}{M_\kappa^2}$$

- Atomic participation ratio (APR)

$$APR_{\boldsymbol{q}j,\kappa} = \frac{|\boldsymbol{e}(\kappa;\boldsymbol{q}j)|^2}{M_\kappa} \Bigg/ \left(N_\kappa \sum_{\kappa}^{N_\kappa} \frac{|\boldsymbol{e}(\kappa;\boldsymbol{q}j)|^4}{M_\kappa^2} \right)^{1/2}$$

For an extended eigenmode, the PR value is of order 1, whereas for a localized eigenmodes PR is of order $1/N_\kappa$[3]. APR is an atomic decomposition of PR that satisfies $PR_{\boldsymbol{q}j} = \sum_\kappa (APR_{\boldsymbol{q}j,\kappa})^2$. To print the PR and APR, please set `MODE = phonons` and `PRINTPR = 1` in the `&analysis` entry field.

Scattering phase space

When *KPMODE* = 2 and `SPS = 1`, the three-phonon scattering phase space P_3 is calculated and saved to the file `PREFIX`.sps. P_3 is defined as

$$P_3(\boldsymbol{q}j) = \frac{1}{3m^3}(2P_3^{(+)}(\boldsymbol{q}j) + P_3^{(-)}(\boldsymbol{q}j)),$$

where m is the number of phonon branches and

$$P_3^{(\pm)}(\boldsymbol{q}j) = \frac{1}{N_q} \sum_{\boldsymbol{q}_1,\boldsymbol{q}_2,j_1,j_2} \delta(\omega_{\boldsymbol{q}j} \pm \omega_{\boldsymbol{q}_1 j_1} - \omega_{\boldsymbol{q}_2 j_2}) \delta_{\boldsymbol{q}\pm\boldsymbol{q}_1,\boldsymbol{q}_2+\boldsymbol{G}}.$$

anphon also print the total scattering phase space

$$P_3 = \frac{1}{N_q} \sum_{\boldsymbol{q}j} P_3(\boldsymbol{q}j).$$

When `SPS = 2`, the three-phonon scattering phase space with the occupation factor $W_3^{(\pm)}$ will be calculated and saved to the file `PREFIX`.sps_Bose. $W_3^{(\pm)}$ is defined as

$$W_3^{(\pm)}(\boldsymbol{q}j) = \frac{1}{N_q} \sum_{\boldsymbol{q}_1,\boldsymbol{q}_2,j_1,j_2} \left\{ \begin{array}{c} n_2 - n_1 \\ n_1 + n_2 + 1 \end{array} \right\} \delta(\omega_{\boldsymbol{q}j} \pm \omega_{\boldsymbol{q}_1 j_1} - \omega_{\boldsymbol{q}_2 j_2}) \delta_{\boldsymbol{q}\pm\boldsymbol{q}_1,\boldsymbol{q}_2+\boldsymbol{G}}.$$

Here, $n_1 = n(\omega_{\boldsymbol{q}_1 j_1})$ and $n_2 = n(\omega_{\boldsymbol{q}_2 j_2})$ where $n(\omega) = \frac{1}{e^{\hbar\omega/k_B T}-1}$ is the Bose-Einstein distribution function. Since $n(\omega)$ is temperature dependent, $W_3^{(\pm)}$ is also temperature dependent. The file `PREFIX`.sps_Bose contains $W_3^{(\pm)}$ for all phonon modes at various temperatures specified with `TMIN`, `TMAX`, and `DT` tags.

Grüneisen parameter

The mode Grüneisen parameter, defined as $\gamma_{\boldsymbol{q}j} = -\frac{\partial \log \omega_{\boldsymbol{q}j}}{\partial \log V}$, is calculated by

$$\gamma_{\boldsymbol{q}j} = -\frac{(\boldsymbol{e}^*_{\boldsymbol{q}j})^{\mathrm{T}} \delta D(\boldsymbol{q}) \boldsymbol{e}_{\boldsymbol{q}j}}{6\omega_{\boldsymbol{q}j}},$$

[3] J. Hafner and M. Krajci, J. Phys.: Condens. Matter **5**, 2489 (1993).

where $\delta D(\boldsymbol{q})$ is a change in the dynamical matrix due to a volume change δV, which is given by

$$\delta D_{\mu\nu}(\kappa\kappa';\boldsymbol{q}) = \frac{1}{\sqrt{M_\kappa M_{\kappa'}}} \sum_{\ell'} \delta\Phi_{\mu\nu}(\ell\kappa;\ell'\kappa') \exp\left[i\boldsymbol{q}\cdot(\boldsymbol{r}(\ell') - \boldsymbol{r}(\ell))\right], \tag{1.12}$$

$$\delta\Phi_{\mu\nu}(\ell\kappa;\ell'\kappa') = \sum_{\ell'',\kappa'',\lambda} \Phi_{\mu\nu\lambda}(\ell\kappa;\ell'\kappa';\ell''\kappa'') r_\lambda(\ell''\kappa''). \tag{1.13}$$

Please set `GRUNEISEN = 1` and give an appropriate `FCSXML` file containing cubic IFCs to print Grüneisen parameters.

Anharmonic self-energy

The anharmonic self-energy due to cubic anharmonicity to the lowest order is given by

$$\begin{aligned}\Sigma_{\boldsymbol{q}j}(i\omega_m) = \frac{1}{2\hbar^2} \sum_{\boldsymbol{q}_1,\boldsymbol{q}_2} \sum_{j_1,j_2} |V^{(3)}_{-\boldsymbol{q}j,\boldsymbol{q}_1 j_1,\boldsymbol{q}_2 j_2}|^2 \\ \times \left[\frac{n_1+n_2+1}{i\omega_m+\omega_1+\omega_2} - \frac{n_1+n_2+1}{i\omega_m-\omega_1-\omega_2} + \frac{n_1-n_2}{i\omega_m-\omega_1+\omega_2} - \frac{n_1-n_2}{i\omega_m+\omega_1-\omega_2}\right],\end{aligned}$$

where $i\omega_m$ is the Matsubara frequency. In equation (1.14), we simply denoted $\omega_{\boldsymbol{q}_i j_i}$ as ω_i. The matrix element $V^{(3)}$ is given by

$$\begin{aligned}V^{(3)}_{\boldsymbol{q}j,\boldsymbol{q}'j',\boldsymbol{q}''j''} = \left(\frac{\hbar}{2N_q}\right)^{\frac{3}{2}} \frac{1}{\sqrt{\omega_{\boldsymbol{q}n}\omega_{\boldsymbol{q}'j'}\omega_{\boldsymbol{q}''j''}}} \sum_{\ell,\ell',\ell''} \exp\left[\mathrm{i}(\boldsymbol{q}\cdot\boldsymbol{r}(\ell) + \boldsymbol{q}'\cdot\boldsymbol{r}(\ell') + \boldsymbol{q}''\cdot\boldsymbol{r}(\ell''))\right] \\ \times \sum_{\kappa,\kappa',\kappa''} \frac{1}{\sqrt{M_\kappa M_{\kappa'} M_{\kappa''}}} \sum_{\mu,\nu,\lambda} \Phi_{\mu\nu\lambda}(\ell\kappa;\ell'\kappa';\ell''\kappa'') e_\mu(\kappa;\boldsymbol{q}j) e_\nu(\kappa';\boldsymbol{q}'j') e_\lambda(\kappa'';\boldsymbol{q}''j'') \,,\end{aligned}$$

which becomes zero unless $\boldsymbol{q}+\boldsymbol{q}'+\boldsymbol{q}''$ is an integral multiple of $\boldsymbol{G} = n_1\boldsymbol{b}_1 + n_2\boldsymbol{b}_2 + n_3\boldsymbol{b}_3$. Phonon linewidth $\Gamma_{\boldsymbol{q}j}$, which is the imaginary part of the phonon self-energy, can be obtained by the analytic continuation to the real axis ($i\omega_m \to \omega + i0^+$) as

$$\begin{aligned}\Gamma^{\mathrm{anh}}_{\boldsymbol{q}j}(\omega) = \frac{\pi}{2\hbar^2} \sum_{\boldsymbol{q}_1,\boldsymbol{q}_2} \sum_{j_1,j_2} |V^{(3)}_{-\boldsymbol{q}j,\boldsymbol{q}_1 j_1,\boldsymbol{q}_2 j_2}|^2 \\ \times \left[-(n_1+n_2+1)\delta(\omega+\omega_1+\omega_2) + (n_1+n_2+1)\delta(\omega-\omega_1-\omega_2)\right. \\ \left. -(n_1-n_2)\delta(\omega-\omega_1+\omega_2) + (n_1-n_2)\delta(\omega+\omega_1-\omega_2)\right].\end{aligned}$$

The computation of equation (1.14) is the most expensive part of the thermal conductivity calculations. Therefore, we employ the crystal symmetry to reduce the number of triplet pairs $(\boldsymbol{q}j, \boldsymbol{q}'j', \boldsymbol{q}''j'')$ of $V^{(3)}$ to calculate. To disable the reduction, please set `TRISYM = 0`.

Isotope scattering

The effect of isotope scatterings can be considered by the mass perturbation approach proposed by S. Tamura[4] by the `ISOTOPE`-tag. The corresponding phonon linewidth is given by

$$\Gamma^{\mathrm{iso}}_{\boldsymbol{q}j}(\omega) = \frac{\pi}{4N_q}\omega^2_{\boldsymbol{q}j} \sum_{\boldsymbol{q}_1,j_1} \delta(\omega-\omega_{\boldsymbol{q}_1 j_1}) \sum_\kappa g_2(\kappa) |\boldsymbol{e}^*(\kappa;\boldsymbol{q}_1\boldsymbol{j}_1)\cdot\boldsymbol{e}(\kappa;\boldsymbol{q}\boldsymbol{j})|^2,$$

where g_2 is a dimensionless factor given by

$$g_2(\kappa) = \sum_i f_i(\kappa)\left(1 - \frac{m_i(\kappa)}{M_\kappa}\right)^2.$$

[4] S. -I. Tamura, Phys. Rev. B **27**, 858 (1983).

Here, f_i is the fraction of the ith isotope of an element having mass m_i, and $M_\kappa = \sum_i f_i m_i(\kappa)$ is the average mass, respectively. The g_2 values should be provided by the `ISOFACT`-tag. The average mass M_κ is substituted by the value specified in the `MASS`-tag.

Lattice thermal conductivity

The lattice thermal conductivity tensor $\kappa_{\rm ph}^{\mu\nu}(T)$ is estimated within the relaxation-time approximation as

$$\kappa_{\rm ph}^{\mu\nu}(T) = \frac{1}{\Omega N_q} \sum_{\boldsymbol{q},j} c_{\boldsymbol{q}j}(T) v_{\boldsymbol{q}j}^{\mu} v_{\boldsymbol{q}j}^{\nu} \tau_{\boldsymbol{q}j}(T),$$

where $c_{\boldsymbol{q}j} = \hbar\omega_{\boldsymbol{q}j}\partial n_{\boldsymbol{q}j}/\partial T$ and $\tau_{\boldsymbol{q}j}(T)$ is the phonon lifetime. The phonon lifetime is estimated using the Matthiessen's rule as

$$\tau_{\boldsymbol{q}j}^{-1}(T) = 2(\Gamma_{\boldsymbol{q}j}^{\rm anh}(T) + \Gamma_{\boldsymbol{q}j}^{\rm iso}).$$

The lattice thermal conductivity is saved in the file `PREFIX`.kl.

The spectra of the lattice thermal conductivity $\kappa_{\rm ph}^{\mu\mu}(\omega)$ can also be calculated by setting `KAPPA_SPEC = 1` in the `&analysis` field. $\kappa_{\rm ph}^{\mu\mu}(\omega)$ is defined as

$$\kappa_{\rm ph}^{\mu\mu}(\omega) = \frac{1}{\Omega N_q} \sum_{\boldsymbol{q},j} c_{\boldsymbol{q}j} v_{\boldsymbol{q}j}^{\mu} v_{\boldsymbol{q}j}^{\mu} \tau_{\boldsymbol{q}j} \delta(\omega - \omega_{\boldsymbol{q}j}).$$

If we integrate this quantity over ω, we then obtain the bulk thermal conductivity, namely $\kappa_{\rm ph}^{\mu\mu} = \int_0^\infty \kappa_{\rm ph}^{\mu\mu}(\omega)\ {\rm d}\omega$.

Cumulative thermal conductivity

The accumulative lattice thermal conductivity $\kappa_{\rm ph,acc}^{\mu\nu}(L)$ is defined as

$$\kappa_{\rm ph,acc}^{\mu\mu}(L) = \frac{1}{\Omega N_q} \sum_{\boldsymbol{q},j} c_{\boldsymbol{q}j} v_{\boldsymbol{q}j}^{\mu} v_{\boldsymbol{q}j}^{\mu} \tau_{\boldsymbol{q}j} \Theta(L - |\boldsymbol{v}_{\boldsymbol{q}j}|\tau_{\boldsymbol{q}j}),$$

where $\Theta(x)$ is the step function. This quantity can be calculated by using the script `analyze_phonons.py` with `--calc cumulative` flag. One can also use another definition for the accumulative thermal conductivity:

$$\kappa_{\rm ph,acc}^{\mu\nu}(L) = \frac{1}{\Omega N_q} \sum_{\boldsymbol{q},j} c_{\boldsymbol{q}j} v_{\boldsymbol{q}j}^{\mu} v_{\boldsymbol{q}j}^{\nu} \tau_{\boldsymbol{q}j} \Theta(L - |v_{\boldsymbol{q}j}^{\mu}|\tau_{\boldsymbol{q}j}).$$

In this case, the contribution to the total thermal conductivity is limited only from phonon modes whose mean-free-path along the μ-direction is smaller than L. To calculate this, please use the `--calc cumulative2` flag and specify the direction μ by the `--direction` option.

Delta function

To compute the phonon DOSs and the imaginary part of phonon self-energies, it is necessary to evaluate the Brillouin-zone integration containing Dirac's delta function. For that purpose, we provide 3 options through the `ISMEAR`-tag.

When `ISMEAR = 0`, the delta function is replaced by the Lorentzian function as

$$\delta(\omega) \approx \frac{1}{\pi} \frac{\epsilon^2}{\omega^2 + \epsilon^2}.$$

When `ISMEAR = 1`, the delta function is replaced by the Gaussian function as

$$\delta(\omega) \approx \frac{1}{\sqrt{\pi}\epsilon} \exp\left(-\omega^2/\epsilon^2\right),$$

which decays faster than the Lorentzian function. For both cases, ϵ should be given by the `EPSILON`-tag, which must be chosen carefully to avoid any unscientific results. ϵ should be small enough to capture detailed phonon structures such as phonon DOS or energy conservation surface related to three-phonon processes, but it should be large enough to avoid unscientific oscillations. Choosing an appropriate value for ϵ is not a trivial task since it may depend on the phonon structure and the density of $\boldsymbol{q}$ points.

To avoid such issues, the program *anphon* employs the tetrahedron method[5] by default (`ISMEAR = -1`) for numerical evaluations of Brillouin zone integration containing $\delta(\omega)$. When the tetrahedron method is used, the `EPSILON`-tag is neglected. We recommend using the tetrahedron method whenever possible.

Self-consistent phonon (SCPH) calculation

The self-consistent phonon mode (`MODE = SCPH`) computes temperature-dependent phonon frequencies by solving the following equation self-consistently[6]:

$$V^{[n]}_{\boldsymbol{q}ij} = \omega^2_{\boldsymbol{q}i}\delta_{ij} + \frac{1}{2}\sum_{\boldsymbol{q}_1,k,\ell} F_{\boldsymbol{q}\boldsymbol{q}_1,ijk\ell}\mathcal{K}^{[n-1]}_{\boldsymbol{q}_1,k\ell}. \tag{1.14}$$

Here, $\omega_{\boldsymbol{q}j}$ is the harmonic phonon frequency and $F_{\boldsymbol{q}\boldsymbol{q}_1,ijk\ell} = \Phi(\boldsymbol{q}i; -\boldsymbol{q}j; \boldsymbol{q}_1k; -\boldsymbol{q}_1\ell)$ is the reciprocal representation of fourth-order force constants. The updated phonon frequency in the nth iteration is obtained by diagonalizing the matrix $V^{[n]}_{\boldsymbol{q}ij}$ as

$$\Lambda^{[n]}_{\boldsymbol{q}} = C^{[n]\dagger}_{\boldsymbol{q}} V^{[n]}_{\boldsymbol{q}} C^{[n]}_{\boldsymbol{q}},$$

where $\omega^{[n]}_{\boldsymbol{q}j} = (\Lambda^{[n]}_{\boldsymbol{q}jj})^{\frac{1}{2}}$ and $C^{[n]}_{\boldsymbol{q}}$ is the unitary matrix that transforms the harmonic phonon eigenvectors into anharmonic ones as $W^{[n]}_{\boldsymbol{q}} = W_{\boldsymbol{q}} C^{[n]}_{\boldsymbol{q}}$. The matrix $\mathcal{K}$ in Eq. (1.14) is defined as

$$\mathcal{K}^{[n]}_{\boldsymbol{q},ij} = \alpha K^{[n]}_{\boldsymbol{q},ij} + (1-\alpha)K^{[n-1]}_{\boldsymbol{q},ij},$$

$$K^{[n]}_{\boldsymbol{q},ij} = \sum_k C^{[n]}_{\boldsymbol{q},ki} C^{[n]*}_{\boldsymbol{q},kj} \frac{\hbar\left[1 + 2n(\omega^{[n]}_{\boldsymbol{q}_1 k})\right]}{2\omega^{[n]}_{\boldsymbol{q}_1 k}}.$$

α is the mixing parameter, which can be changed via the `MIXALPHA` tag.

The SCPH equation is solved on the irreducible $\boldsymbol{q}$ grid defined by the `KMESH_INTERPOLATE` tag. The $\boldsymbol{q}_1$ grid in Eq. (1.14), given by the `KMESH_SCPH` tag, can be finer than the $\boldsymbol{q}$ grid. After the SCPH iteration converges, the code computes the anharmonic correction to the harmonic force constant $\Delta D(\boldsymbol{r}(\ell))$ as follows:

$$\Delta D(\boldsymbol{r}(\ell)) = \frac{1}{N_q}\sum_{\boldsymbol{q}} \Delta D(\boldsymbol{q}) e^{-i\boldsymbol{q}\cdot\boldsymbol{r}(\ell)},$$

$$\Delta D(\boldsymbol{q}) = D_{\mathrm{SCPH}}(\boldsymbol{q}) - D_{\mathrm{Harmonic}}(\boldsymbol{q}),$$

$$D_{\mathrm{SCPH}}(\boldsymbol{q}) = W_{\boldsymbol{q}} C^{[n]}_{\boldsymbol{q}} \Lambda^{[n]}_{\boldsymbol{q}} C^{[n]\dagger}_{\boldsymbol{q}} W^{\dagger}_{\boldsymbol{q}}.$$

$\Delta D(\boldsymbol{r}(\ell))$ is saved in `PREFIX.scph_dfc2`.

The most computationally expensive part is the calculation of matrix elements of $F_{\boldsymbol{q}\boldsymbol{q}_1,ijk\ell}$. When `SELF_OFFDIAG = 0` (default), the code only computes the elements of $F_{\boldsymbol{q}\boldsymbol{q}_1,iikk}$. Therefore, the computational complexity is $\mathcal{O}(N_q^{\mathrm{irred.}} N_{q_1} m^2)$. When `SELF_OFFDIAG = 1`, the off-diagonal elements are also calculated, and the computational complexity is $\mathcal{O}(N_q^{\mathrm{irred.}} N_{q_1} m^4)$.

[5] P. E. Blöchl, O. Jepsen, and O. K. Andersen, Phys. Rev. B **49**, 1450555 (1994).
[6] T. Tadano and S. Tsuneyuki, Phys. Rev. B **92**, 054301 (2015).

1.7 Tutorial

Input files prepared for this tutorial are located in the **example/** directory of the ALAMODE package.

1. *Silicon*
2. *Silicon with LAMMPS*

1.7.1 Silicon

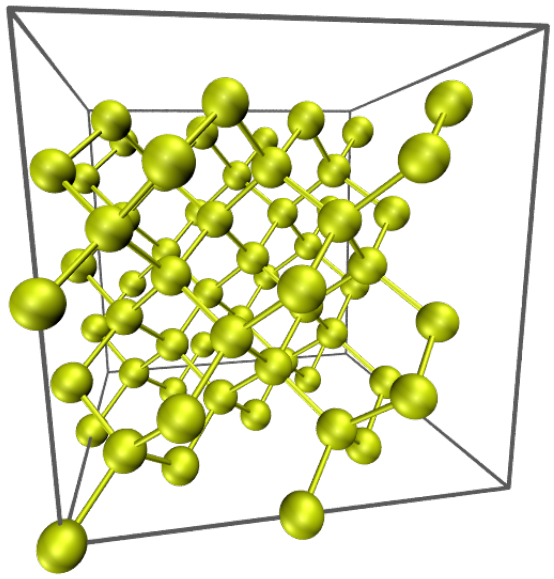


Fig. 1: Silicon. 2x2x2 conventional supercell

In the following, (anharmonic) phonon properties of bulk silicon (Si) are calculated by a 2x2x2 conventional cell containing 64 atoms.

1. *Get displacement pattern by alm*
2. *Calculate atomic forces for the displaced configurations*
3. *Estimate force constants by fitting*
4. *Calculate phonon dispersion and phonon DOS*
5. *Estimate anharmonic IFCs for thermal conductivity*
6. *Calculate thermal conductivity*
7. *Analyze results*

1. Get displacement patterns by alm

Change directory to **example/Si** and open file si_alm.in. This file is an input for the code **alm** which estimate interatomic force constants (**IFC**) by least square fitting. In the file, the crystal structure of a 2x2x2 conventional supercell of Si is specified in the **&cell** and the **&position** fields as the following:

```
&general
  PREFIX = si222
  MODE = suggest
  NAT = 64; NKD = 1
  KD = Si
/

&interaction
  NORDER = 1  # 1: harmonic, 2: cubic, ..
/

&cell
  20.406 # factor in Bohr unit
  1.0 0.0 0.0 # a1
  0.0 1.0 0.0 # a2
  0.0 0.0 1.0 # a3
/

&cutoff
  Si-Si None
/

&position
  1 0.0000000000000000 0.0000000000000000 0.0000000000000000
  1 0.0000000000000000 0.0000000000000000 0.5000000000000000
  1 0.0000000000000000 0.2500000000000000 0.2500000000000000
  1 0.0000000000000000 0.2500000000000000 0.7500000000000000
  1 0.0000000000000000 0.5000000000000000 0.0000000000000000
  1 0.0000000000000000 0.5000000000000000 0.5000000000000000
  1 0.0000000000000000 0.7500000000000000 0.2500000000000000
```

Replace the lattice constant of the supercell (20.406 Bohr) by your own value.

Then, execute **alm**

```
$ alm si_alm.in > si_alm.log1
```

which creates a file si222.pattern_HARMONIC in the working directory. In the pattern file, suggested displacement patterns are defined in Cartesian coordinates. As you can see in the file, there is only one displacement pattern for harmonic IFCs of bulk Si.

2. Calculate atomic forces for the displaced configurations

Next, calculate atomic forces for all the displaced configurations defined in si222.pattern_HARMONIC. To do so, you first need to decide the magnitude of displacements Δu, which should be small so that anharmonic contributions are negligible. In most cases, $\Delta u \sim 0.01$ Å is a reasonable choice.

Then, prepare input files necessary to run an external DFT code for each configuration. Since this procedure is a little tiresome, we provide a subsidiary Python script for VASP, Quantum-ESPRESSO (QE), and xTAPP. Using the script displace.py in the tools/ directory, you can generate the necessary input files as follows:

QE

```
$ python displace.py --QE=si222.pw.in --mag=0.01 si222.pattern_HARMONIC
```

VASP

```
$ python displace.py --VASP=POSCAR.orig --mag=0.01 si222.pattern_HARMONIC
```

xTAPP

```
$ python displace.py --xTAPP=si222.cg --mag=0.01 si222.pattern_HARMONIC
```

OpenMX

```
$ python displace.py --OpenMX=si222.dat --mag=0.01 si222.pattern_HARMONIC
```

The `--mag` option specifies the displacement length in units of Angstrom. You need to specify an input file with equilibrium atomic positions either by the `--QE`, `--VASP`, `--xTAPP`, `--OpenMX` or `--LAMMPS`.

Then, calculate atomic forces for all the configurations. This can be done with a simple shell script as follows:

```
#!/bin/bash

# Assume we have 20 displaced configurations for QE [disp01.pw.in,..., disp20.pw.in].

for ((i=1;i<=20;i++))
do
    num=`echo $i | awk '{printf("%02d",$1)}'`
    mkdir ${num}
    cd ${num}
    cp ../disp${num}.pw.in .
    pw.x < disp${num}.pw.in > disp${num}.pw.out
    cd ../
done
```

Important: In QE, you need to set tprnfor=.true. to print out atomic forces.

The next step is to collect the displacement data and force data by the Python script extract.py (also in the tools/ directory). This script can extract atomic displacements, atomic forces, and total energies from multiple output files as follows:

QE

```
$ python extract.py --QE=si222.pw.in *.pw.out > DFSET_harmonic
```

VASP

```
$ python extract.py --VASP=POSCAR.orig vasprun*.xml > DFSET_harmonic
```

xTAPP

```
$ python extract.py --xTAPP=si222.cg *.str > DFSET_harmonic
```

OpenMX

```
$ python extract.py --OpenMX=si222.dat *.out > DFSET_harmonic
```

In the above examples, atomic displacements and corresponding atomic forces of all the configurations are merged as DFSET_harmonic. These files will be used in the following fitting procedure as `DFSET` (See *Format of DFSET*).

Note: For your convenience, we provide the DFSET_harmonic file in the reference/ subdirectory. These files are generated by the Quantum-ESPRESSO package with `--mag=0.02`. You can proceed to the next step by copying these files to the working directory.

3. Estimate force constants by fitting

Edit the file si_alm.in to perform least-square fitting. Change the `MODE = suggest` to `MODE = optimize` as follows:

```
&general
  PREFIX = si222
  MODE = optimize   # <-- here
  NAT = 64; NKD = 1
  KD = Si
/
```

Also, add the **&optimize** field as:

```
&fitting
  DFSET = DFSET_harmonic
/
```

Then, execute **alm** again

```
$ alm si_alm.in > si_alm.log2
```

This time **alm** extract harmonic IFCs from the given displacement-force data set (`DFSET_harmonic`).

You can find files si222.fcs and si222.xml in the working directory. The file si222.fcs contains all IFCs in Rydberg atomic units. You can find symmetrically irreducible sets of IFCs in the first part as:

```
************************ Force Constants (FCs) ************************
*        Force constants are printed in Rydberg atomic units.       *
*        FC2: Ry/a0^2     FC3: Ry/a0^3     FC4: Ry/a0^4   etc.      *
*        FC?: Ry/a0^?     a0 = Bohr radius                           *
*                                                                    *
*        The value shown in the last column is the distance         *
*        between the most distant atomic pairs.                     *
**********************************************************************

----------------------------------------------------------------------
     Index               FCs          P        Pairs     Distance [Bohr]
(Global, Local)               (Multiplicity)
----------------------------------------------------------------------

 *FC2
      1       1     2.7613553e-01   1     1x     1x       0.000
      2       2    -9.5252308e-04   2     1x     2x      10.203
      3       3    -3.7574247e-04   2     1x    33x      10.203
      4       4     8.8561868e-03   1     1x     3x       7.215
      5       5     1.9633030e-03   1     1x     3y       7.215
      6       6    -3.9798276e-03   1     1x    17x       7.215
```

(continues on next page)

(continued from previous page)

```
       7       7    -3.9080855e-03   1     1x     17z       7.215
       8       8     8.2188178e-03   4     1x      6x      14.429
       9       9     4.3878651e-05   4     1x     34x      14.429
      10      10    -6.7358757e-02   1     1x      9x       4.418
      11      11    -4.7360832e-02   1     1x      9y       4.418
      12      12     1.0670028e-03   1     1x     10x       8.460
      13      13    -8.2098635e-04   1     1x     10y       8.460
      14      14    -6.9742830e-04   1     1x     10z       8.460
      15      15     2.1135669e-04   1     1x     26x       8.460
      16      16    -4.3031422e-03   1     1x     11x      11.118
      17      17     5.0759340e-04   1     1x     11y      11.118
      18      18    -4.7178438e-04   1     1x     25x      11.118
      19      19    -4.7518256e-04   1     1x     25z      11.118
      20      20    -7.4138963e-04   2     1x     20x      12.496
      21      21     1.2900836e-03   2     1x     20y      12.496
      22      22    -1.7263732e-04   2     1x     20z      12.496
      23      23     6.4515756e-05   2     1x     35x      12.496
      24      24     3.5369980e-04   1     1x     28x      13.254
      25      25    -5.1486930e-04   1     1x     28y      13.254
```

Harmonic IFCs $\Phi_{\mu\nu}(i,j)$ in the supercell are given in the third column and the multiplicity P is the number of times each interaction (i,j) occurs within the given cutoff radius. For example, $P = 2$ for the pair $(1x, 2x)$ because the distance $r_{1,2}$ is exactly the same as the distance $r_{1,2'}$ where the atom 2' is a neighboring image of atom 2 under the periodic boundary condition. If you compare the magnitude of IFCs, the values in the third column should be divided by P.

In the log file si_alm2.log, the *fitting error* is printed. Try

```
$ grep "Fitting error" si_alm2.log
Fitting error (%) : 0.567187
```

The other file si222.xml contains crystal structure, symmetry, IFCs, and all other information necessary for subsequent phonon calculations.

4. Calculate phonon dispersion and phonon DOS

Open the file si_phband.in and edit it for your system.

```
&general
  PREFIX = si222
  MODE   = phonons
  FCSXML = si222.xml

  NKD = 1; KD = Si
  MASS = 28.0855
/

&cell
  10.203
  0.0 0.5 0.5
  0.5 0.0 0.5
  0.5 0.5 0.0
/

&kpoint
```

(continues on next page)

(continued from previous page)

```
  1  # KPMODE = 1: line mode
  G 0.0 0.0 0.0 X 0.5 0.5 0.0 51
  X 0.5 0.5 1.0 G 0.0 0.0 0.0 51
  G 0.0 0.0 0.0 L 0.5 0.5 0.5 51
/
```

Please specify the XML file you obtained in step 3 by the `FCSXML`-tag as above. In the **&cell**-field, you need to define the lattice vector of a **primitive cell**. In phonon dispersion calculations, the first entry in the **&kpoint**-field should be 1 (**KPMODE** = 1).

Then, execute **anphon**

```
$ anphon si_phband.in > si_phband.log
```

which creates a file si222.bands in the working directory. In this file, phonon frequencies along the given reciprocal path are printed in units of cm^{-1} as:

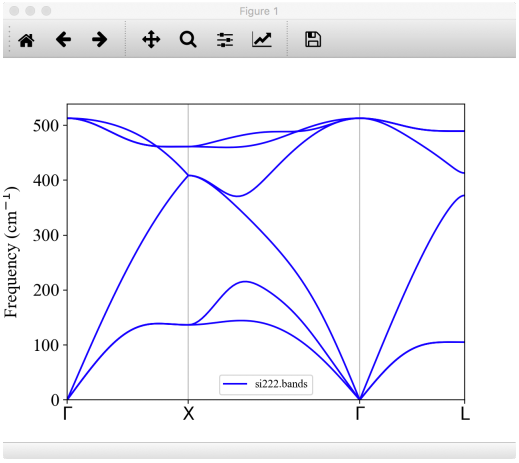
```
#  G X G L
# 0.000000 0.615817 1.486715 2.020028
# k-axis, Eigenvalues [cm^-1]
0.000000   1.097373e-10   1.097373e-10   1.097373e-10   5.129224e+02   5.129224e+02
↪5.129224e+02
0.012316   6.009520e+00   6.009520e+00   1.022526e+01   5.128286e+02   5.128286e+02
↪5.128923e+02
0.024633   1.200989e+01   1.200989e+01   2.044420e+01   5.125479e+02   5.125479e+02
↪5.128019e+02
0.036949   1.799191e+01   1.799191e+01   3.065053e+01   5.120824e+02   5.120824e+02
↪5.126510e+02
0.049265   2.394626e+01   2.394626e+01   4.083799e+01   5.114353e+02   5.114353e+02
↪5.124391e+02
0.061582   2.986346e+01   2.986346e+01   5.100034e+01   5.106114e+02   5.106114e+02
↪5.121656e+02
0.073898   3.573381e+01   3.573381e+01   6.113142e+01   5.096167e+02   5.096167e+02
↪5.118299e+02
```

You can plot the phonon dispersion relation with gnuplot or any other plot software.

For visualizing phonon dispersion relations, we provide a Python script plotband.py in the tools/ directory (Matplotlib is required.). Try

```
$ python plotband.py si222.bands
```

Then, the phonon dispersion is displayed as follows:



You can save the figure as png, eps, or other formats from this window. You can also change the energy unit of phonon frequency from cm^{-1} to THz or meV by the `--unit` option. For more detail of the usage of plotband.py, type

```
$ python plotband.py -h
```

Next, let us calculate the phonon DOS. Copy si_phband.in to si_phdos.in and edit the **&kpoint** field as follows:

```
&kpoint
  2   # KPMODE = 2: uniform mesh mode
  20 20 20
/
```

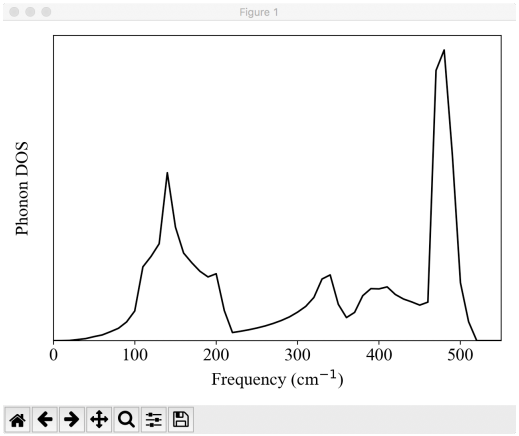
Then, execute **anphon**

```
$ anphon si_phdos.in > si_phdos.log
```

This time, **anphon** creates files si222.dos and si222.thermo in the working directory, which contain phonon DOS and thermodynamic functions, respectively. For visualizing phonon DOS and projected DOSs, we provide a Python script plotdos.py in the tools/ directory (Matplotlib is required.). The command

```
$ python plotdos.py --emax 550 --nokey si222.dos
```

will show the phonon DOS of Si by a pop-up window:



To improve the resolution of DOS, try again with a denser k grid and a smaller `DELTA_E` value.

5. Estimate cubic IFCs for thermal conductivity

Copy file si_alm.in to si_alm2.in. Edit the **&general**, **&interaction**, and **&cutoff** fields of si_alm2.in as the following:

```
&general
  PREFIX = si222_cubic
  MODE = suggest
  NAT = 64; NKD = 1
  KD = Si
/
```

Change the `PREFIX` from *si222* to *si222_cubic* and set `MODE` to *suggest*.

```
&interaction
  NORDER = 2
/
```

Change the `NORDER` tag from `NORDER = 1` to `NORDER = 2` to include cubic IFCs. Here, we consider cubic interaction pairs up to second nearest neighbors by specifying the cutoff radii as:

```
&cutoff
  Si-Si None 7.3
/
```

7.3 Bohr is slightly larger than the distance of second nearest neighbors (7.21461 Bohr). Change the cutoff value appropriately for your own case. (Atomic distance can be found in the file si_alm.log.)

Then, execute **alm**

```
$ alm si_alm2.in > si_alm2.log
```

which creates files si222_cubic.pattern_HARMONIC and si222_cubic.pattern_ANHARM3.

Then, calculate atomic forces of displaced configurations given in the file si222_cubic.pattern_ANHARM3, and collect the displacement and force datasets to a file DFSET_cubic as you did for harmonic IFCs in Steps 3 and 4.

Note: Since making DFSET_cubic requires moderate computational resources, you can skip this procedure by copying file reference/DFSET_cubic to the working directory. The file we provide is generated by the Quantum-ESPRESSO package with `--mag=0.04`.

In si_alm2.in, change `MODE = suggest` to `MODE = optimize` and add the following:

```
&optimize
  DFSET = DFSET_cubic
  FC2XML = si222.xml # Fix harmonic IFCs
/
```

By the `FC2XML` tag, harmonic IFCs are fixed to the values in si222.xml. Then, execute **alm** again

```
$ alm si_alm2.in > si_alm2.log2
```

which creates files si222_cubic.fcs and si222_cubic.xml. This time cubic IFCs are also included in these files.

Note: In the above example, we obtained cubic IFCs by least square fitting with harmonic IFCs being fixed to the value of the previous harmonic calculation. You can also estimate both harmonic and cubic IFCs simultaneously instead. To do this, merge DFSET_harmonic and DFSET_cubic as

```
$ cat DFSET_harmonic DFSET_cubic > DFSET_merged
```

and change the **&fitting** field as the following:

```
&fitting
  DFSET = DFSET_merged
/
```

6. Calculate thermal conductivity

Copy file si_phdos.in to si_RTA.in and edit the `MODE` and `FCSXML` tags as follows:

```
&general
  PREFIX = si222
  MODE = RTA
  FCSXML = si222_cubic.xml

  NKD = 1; KD = Si
  MASS = 28.0855
/
```

In addition, change the k grid density as:

```
&kpoint
  2
  10 10 10
/
```

Then, execute **anphon** as a background job

```
$ anphon si_RTA.in > si_RTA.log &
```

Please be patient. This can take a while. When the job finishes, you can find a file si222.kl in which the lattice thermal conductivity is saved. You can plot this file using gnuplot (or any other plotting softwares) as follows:

```
$ gnuplot
gnuplot> set logscale xy
gnuplot> set xlabel "Temperature (K)"
gnuplot> set ylabel "Lattice thermal conductivity (W/mK)"
gnuplot> plot "si222.kl" usi 1:2 w lp
```

As you can see, the thermal conductivity diverges in $T \to 0$ limit. This occurs because we only considered intrinsic phonon-phonon scatterings in the present calculation and neglected phonon-boundary scatterings which are dominant in the low-temperature range. The effect of the boundary scattering can be included using the python script `analyze_phonons.py` in the tools directory:

```
$ analyze_phonons.py --calc kappa_boundary --size 1.0e+6 si222.result > si222_
↪boundary_1mm.kl
```

In this script, the phonon lifetimes are altered using the Matthiessen's rule

$$\frac{1}{\tau_q^{total}} = \frac{1}{\tau_q^{p-p}} + \frac{2|\boldsymbol{v}_q|}{L}.$$

Here, the first term on the right-hand side of the equation is the scattering rate due to the phonon-phonon scattering and the second term is the scattering rate due to a grain boundary of size L. The size L must be specified using the

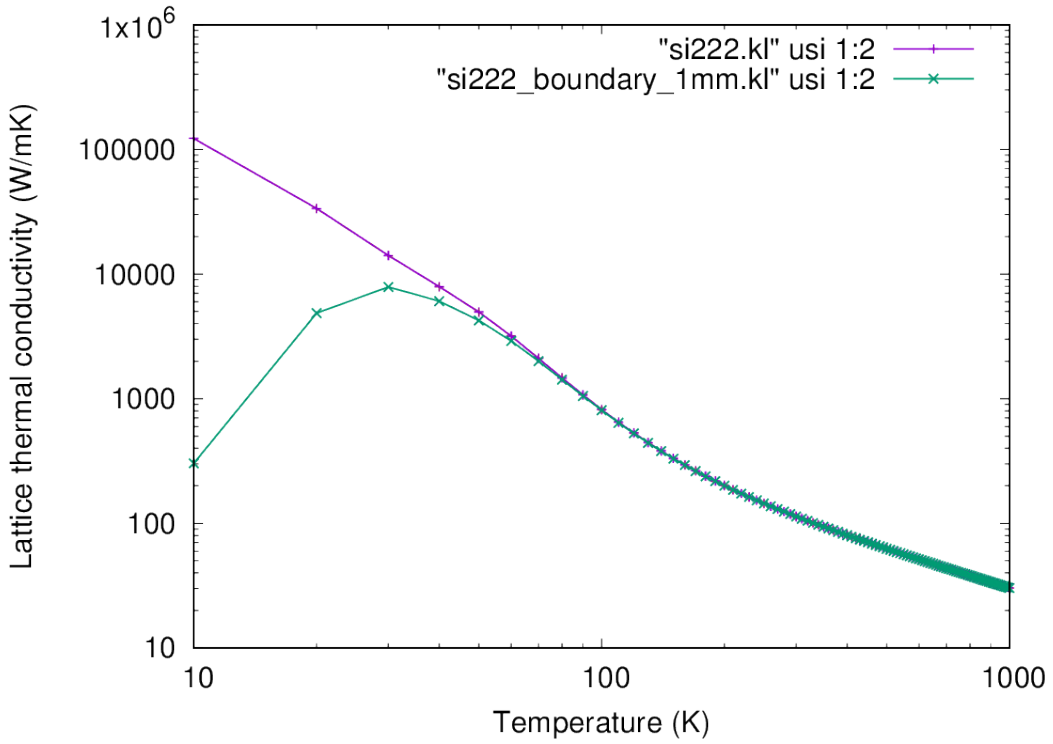


Fig. 2: Calculated lattice thermal conductivity of Si (click to enlarge)

`--size` option in units of nm. The result is also shown in the above figure and the divergence is cured with the boundary effect.

Note: When a calculation is performed with a smearing method (`ISMEAR=0 or 1`) instead of the tetrahedron method (`ISMEAR=-1`), the thermal conductivity may have a peak structure in the very low-temperature region even without the boundary effect. This peak occurs because of the finite smearing width ϵ used in the smearing methods. As we decrease the ϵ value, the peak value of κ should disappear. In addition, a very dense q grid is necessary for describing phonon excitations and thermal transport in the low-temperature region (regardless of the `ISMEAR` value).

---

7. Analyze results

There are many ways to analyze the result for better understandings of nanoscale thermal transport. Some selected features are introduced below:

Phonon lifetime

The file si222.result contains phonon linewidths at irreducible k points. You can extract phonon lifetime from this file as follows:

```
$ analyze_phonons.py --calc tau --temp 300 si222.result > tau300K_10.dat
$ gnuplot
gnuplot> set xrange [1:]
gnuplot> set logscale y
```

(continues on next page)

(continued from previous page)

```
gnuplot> set xlabel "Phonon frequency (cm^{-1})"
gnuplot> set ylabel "Phonon lifetime (ps)"
gnuplot> plot "tau300K_10.dat" using 3:4 w p
```

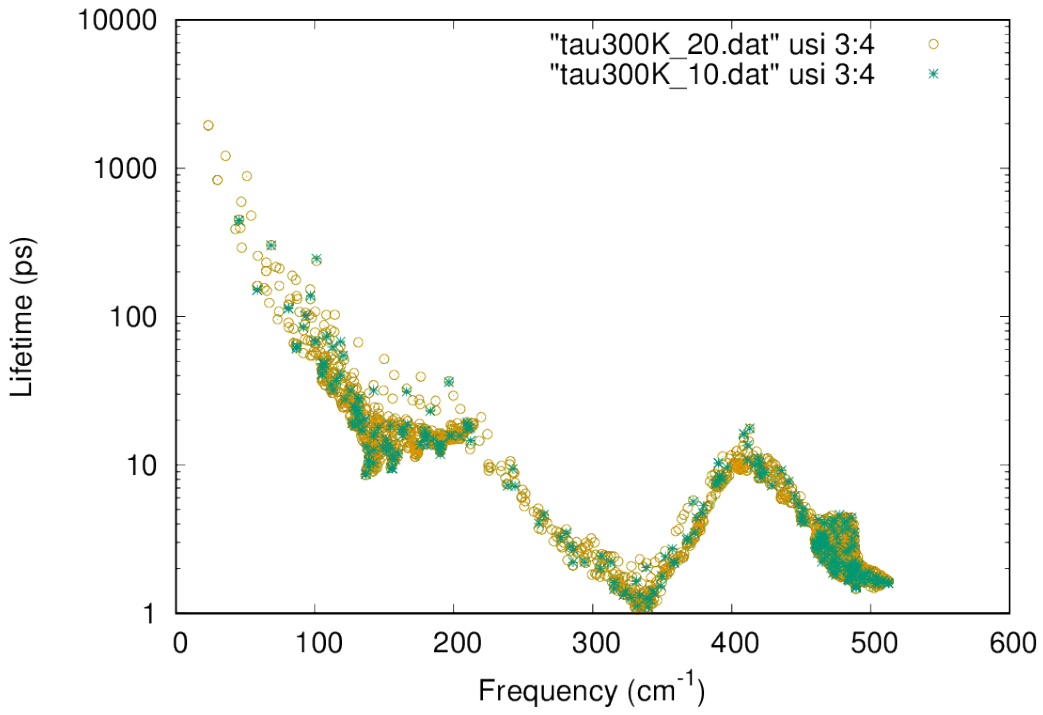


Fig. 3: Phonon lifetime of Si at 300 K (click to enlarge)

In the above figure, phonon lifetimes calculated with $20 \times 20 \times 20$ q points are also shown by open circles.

Cumulative thermal conductivity

Following the procedure below, you can obtain the *cumulative thermal conductivity*:

```
$ analyze_phonons.py --calc cumulative --temp 300 --length 10000:5 si222.result >␣
↪cumulative_300K_10.dat
$ gnuplot
gnuplot> set logscale x
gnuplot> set xlabel "L (nm)"
gnuplot> set ylabel "Cumulative kappa (W/mK)"
gnuplot> plot "cumulative_300K_10.dat" using 1:2 w lp
```

To draw a smooth line, you have to use a denser q grid as shown in the figure by the orange line, which are obtained with $20 \times 20 \times 20$ q points.

Thermal conductivity spectrum

To calculate the *spectrum of thermal conductivity*, modify the si_RTA.in as follows:

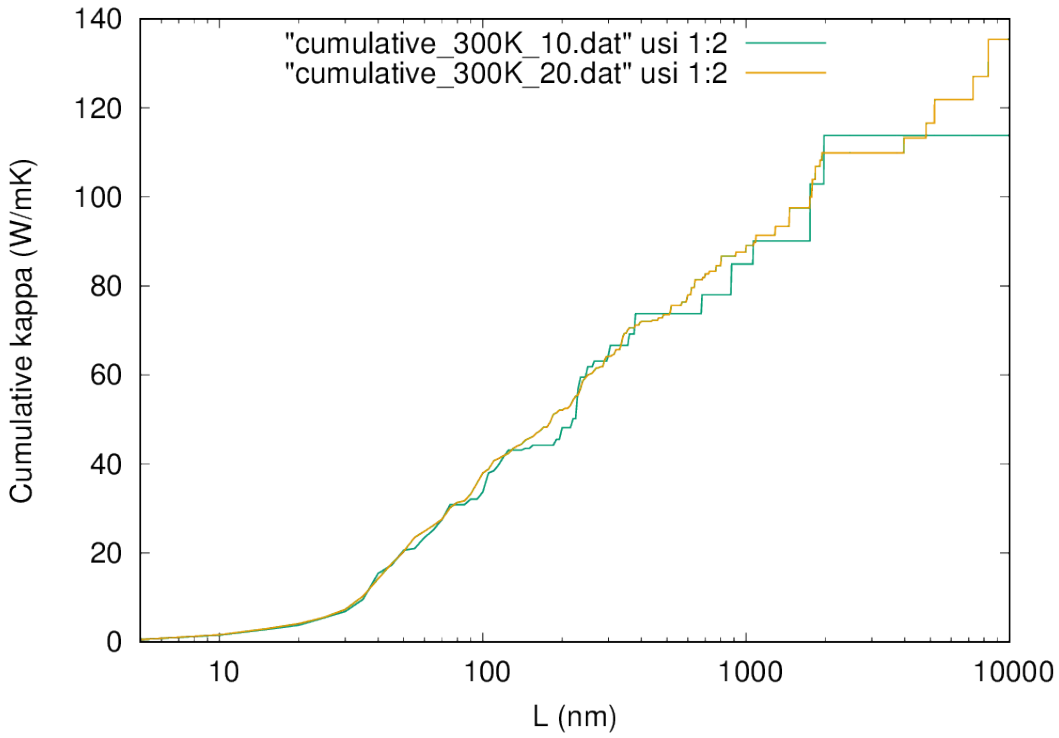


Fig. 4: Cumulative thermal conductivity of Si at 300 K (click to enlarge)

```
&general
  PREFIX = si222
  MODE = RTA
  FCSXML = si222_cubic.xml

  NKD = 1; KD = Si
  MASS = 28.0855

  EMIN = 0; EMAX = 550; DELTA_E = 1.0 # <-- frequency range
/

&cell
10.203
 0.0 0.5 0.5
 0.5 0.0 0.5
 0.5 0.5 0.0
/

&kpoint
 2
 10 10 10
/

&analysis
 KAPPA_SPEC = 1 # compute spectrum of kappa
/
```

The frequency range is specified with the `EMIN`, `EMAX`, and `DELTA_E` tags, and the `KAPPA_SPEC = 1` is set in the

`&analysis` field. Then, execute anphon again

```
$ anphon si_RTA.in > si_RTA2.log
```

After the calculation finishes, you can find the file si222.kl_spec which contains the spectra of thermal conductivity at various temperatures. You can plot the data at room temperature as follows:

```
$ awk '{if ($1 == 300.0) print $0}' si222.kl_spec > si222_300K_10.kl_spec
$ gnuplot
gnuplot> set xlabel "Frequency (cm^{-1})"
gnuplot> set ylabel "Spectrum of kappa (W/mK/cm^{-1})"
gnuplot> plot "si222_300K_10.kl_spec" using 2:3 w l lt 2 lw 2
```

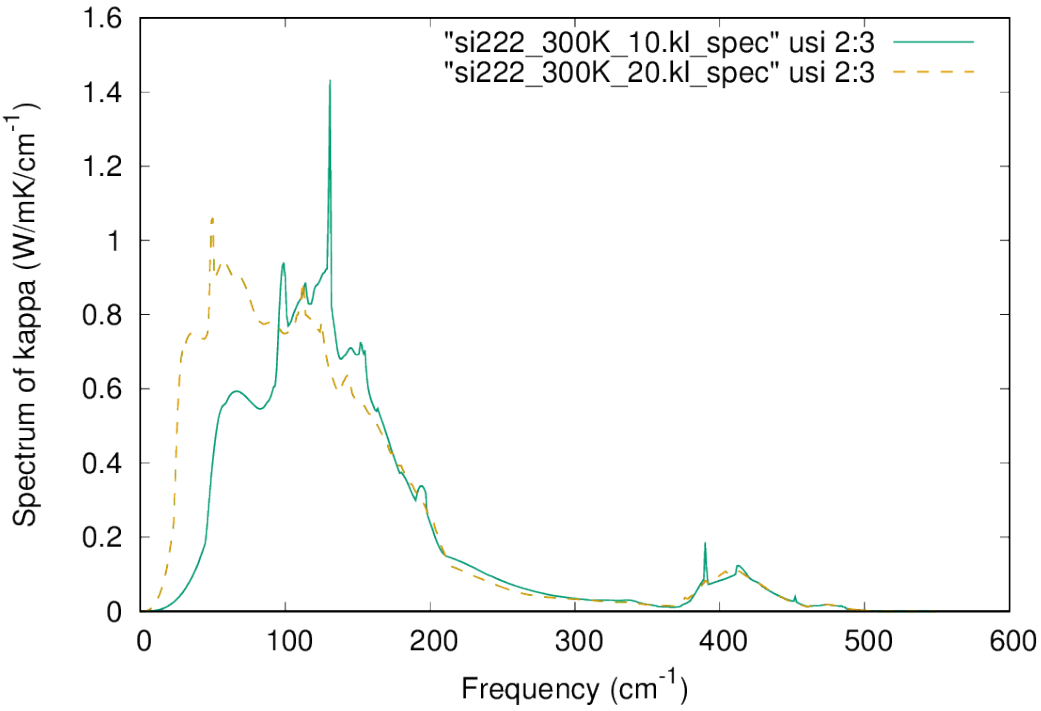


Fig. 5: The spectrum of thermal conductivity of Si at 300 K (click to enlarge)

In the above figure, the computational result with $20 \times 20 \times 20$ q points is also shown by the dashed line. From the figure, we can observe that low-energy phonons below 200 cm^{-1} account for more than 80% of the total thermal conductivity at 300 K.

1.7.2 Silicon with LAMMPS

Here, we demonstrate how to use ALAMODE together with LAMMPS. All input files can be found in the **example/Si_LAMMPS** directory. Before starting the tutorial, please build the LAMMPS code (e.g. `lmp_serial`).

As a simple example, we calculate phonon dispersion curves of Si using the Stillinger-Weber (SW) potential implemented in LAMMPS. First, you need to make two input files for LAMMPS: in.sw and Si222.lammps (file name is arbitrary, though). in.sw is the main input file for LAMMPS, in which the type of the empirical force field is defined as follows:

```
units           metal
atom_style      atomic
boundary        p p p

read_data       tmp.lammps

pair_style      sw
pair_coeff         * * Si.sw Si

dump            1 all custom 1 XFSET id xu yu zu fx fy fz
dump_modify     1 format float "%20.15f"
run             0
```

In the file Si222.lammps, the lattice vectors and atomic positions of a relaxed supercell structure are defined as follows:

```
# Structure data of Si (2x2x2 conventional)

64 atoms
1 atom types

0.000000   10.800000   xlo xhi
0.000000   10.800000   ylo yhi
0.000000   10.800000   zlo zhi
0.000000    0.000000    0.000000   xy xz yz

Masses

 1 28.085

Atoms

   1  1       0.000000 0.000000 0.000000
   2  1       0.000000 0.000000 5.400000
   3  1       0.000000 2.700000 2.700000
   4  1       0.000000 2.700000 8.100000
   5  1       0.000000 5.400000 0.000000
   6  1       0.000000 5.400000 5.400000
   7  1       0.000000 8.100000 2.700000
   8  1       0.000000 8.100000 8.100000
   9  1       1.350000 1.350000 1.350000
  10  1       1.350000 1.350000 6.750000
  11  1       1.350000 4.050000 4.050000
  12  1       1.350000 4.050000 9.450000
  13  1       1.350000 6.750000 1.350000
  14  1       1.350000 6.750000 6.750000
```

Next, please generate a set of structure files for displaced configurations using the python script:

```
$ python displace.py --LAMMPS=Si222.lammps --mag=0.01 --prefix harm  si222.pattern_
↪HARMONIC
$ python displace.py --LAMMPS=Si222.lammps --mag=0.04 --prefix cubic si222.pattern_
↪ANHARM3
```

The pattern files can be generated by the **alm** code as decribed *here*. The above commands create `harm1.lammps` and `cubic[01-20].lammps` structure files. Then, run the following script and calculate atomic forces for the generated structures. This should finish in a few seconds.

```
#!/bin/bash

cp harm1.lammps tmp.lammps
lmp_serial < in.sw > log.lammps
cp XFSET XFSET.harm1

for ((i=1;i<=20;i++))
do
    num=`echo $i | awk '{printf("%02d",$1)}'`
    cp cubic${num}.lammps tmp.lammps
    lmp_serial < in.sw > log.lammps
    cp XFSET XFSET.cubic${num}
done
```

After the force calculations are finished, displacement and force data sets can be generated as follows:

```
.. bash::
$ python extract.py --LAMMPS=Si222.lammps XFSET.harm1 > DFSET_harmonic

$ python extract.py --LAMMPS=Si222.lammps XFSET.cubic* > DFSET_harmonic
```

Then, using these files and following exactly the same procedure as the last tutorial section, you can calculate phonons and thermal conductivity of Si using the SW potential.

1.8 FAQ

CHAPTER 2

Indices and tables

- genindex
- modindex
- search