ALAMODE Documentation

Release 1.0.2

Terumasa Tadano

Nov 10, 2018

Contents

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 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-consisitent phonon (SCPH) calculation

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

Copyright © 2014, 2015, 2016 Terumasa Tadano

This software is distributed under the MIT license. See the LICENSE.txt file for license rights and limitations.

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]

1.1.6 Acknowledgement

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

International Center for Young Scientists (ICYS),
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 (Either OpenMPI, MPICH2, or IntelMPI)
- Boost C++ library
- FFTW library
- Eigen3 library

In addition to the above requirements, users have to get and install a first-principles package (such as VASP, Wien2k, QUANTUM-ESPRESSO, 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 (Python 2) 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

0. Install the LAPACK, MPI, FFTW, Boost C++, and Eigen3 libraries.

   To install the Boost C++ library, please download a source file from the webpage and unpack the file. Then, copy the 'boost' subdirectory to the include folder in the home directory (or anywhere you like). 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 made a symbolic link to the 'boost' subdirectory 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 .
   ```

1. Download the package of ALAMODE from the download page or clone from the git repository.

2. Change directory to the location of the file and untar the file as follows:

   ```
   $ tar -xvzf alamode-x.y.z.tar.gz
   ```

   This will create a directory alamode-x.y.z containing the following subdirectories:

   - alm/ : Source files for alm (force constant calculation)

   - anphon/ : Source files for anphon (phonon calculation)

   - 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

3. Edit the Makefiles

   In directories alm/ and anphon/, we provide sample Makefiles for Linux (with Intel compiler) and Mac OSX (with gcc). Please copy one of them as `Makefile` and modify it appropriately for your environment.

   Here's a typical setting for Linux with Intel compiler:

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

```
CXXL = ${CXX}
LDFLAGS = -mkl

LAPACK =
LIBS = ${LAPACK}
```

To enable OpenMP parallelization, please add the `-openmp` (Intel) or `-fopenmp` (gcc) option in `CXXFLAGS`. In addition, the directory containing the boost/ and Eigen/ subdirectories must be given in `INCLUDE`.

4. Make executables by `make` command.

   If the compilation is successful, the binary file named **alm** (**anphon**) is created in the alm/ (anphon/) directory. To use some auxiliary scripts for post-processing and data conversion, please compile the programs in the tools directory as well. See README.md in the tools directory for details about the auxiliary programs.

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 a least-squares fitting

   In order to perform a fitting, please change the variable `MODE` of the input file *alm.in* to `MODE = fitting`. In addition please add the `&fitting` entry with appropriate `NDATA`, `DFILE`, and `FFILE`. (`DFILE` should be like `DFILE = disp_all.dat`.) 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 being 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 Tutorial

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

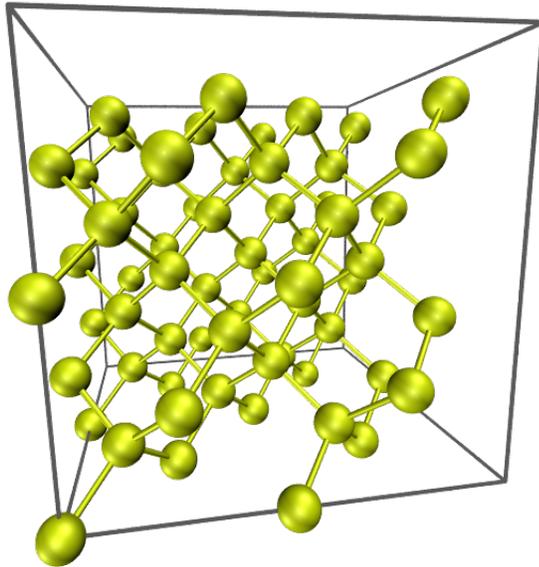1. *Silicon*

2. *Silicon with LAMMPS*

Fig. 1: Silicon. 2x2x2 conventional supercell

1.5.1 Silicon

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
```

(continues on next page)

```
/

&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=0.02 si222.pattern_HARMONIC
```

VASP

```
$ python displace.py --VASP=POSCAR.orig --mag=0.02 si222.pattern_HARMONIC
```

xTAPP

```
$ python displace.py --xTAPP=si222.cg --mag=0.02 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`, 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 --get=disp *.pw.out > disp.dat
$ python extract.py --QE=si222.pw.in --get=force *.pw.out > force.dat
```

VASP

```
$ python extract.py --VASP=POSCAR.orig --get=disp vasprun*.xml > disp.dat
$ python extract.py --VASP=POSCAR.orig --get=force vasprun*.xml > force.dat
```

xTAPP

```
$ python extract.py --xTAPP=si222.cg --get=disp *.str > disp.dat
$ python extract.py --xTAPP=si222.cg --get=force *.str > force.dat
```

In the above examples, atomic displacements of all the configurations are merged as *disp.dat*, and the corresponding atomic forces are saved in the file *force.dat*. These files will be used in the following fitting procedure as `DFILE` and `FFILE`. (See *Format of DFILE and FFILE*).

Note: For your convenience, we provide the disp.dat and force.dat files in the reference/ subdirectory. These files were 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 = fitting` as follows:

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

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

```
&fitting
  NDATA = 1
  DFILE = disp.dat
  FFILE = force.dat
/
```

Then, execute **alm** again

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

This time **alm** extract harmonic IFCs from the given displacement-force data set (disp.dat and force.dat above).

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     2.7617453e-01   1     1x     1x       0.000
       2       2    -1.2195057e-03   2     1x     2x      10.203
       3       3    -6.2496693e-04   2     1x    33x      10.203
       4       4     8.5935598e-03   1     1x     3x       7.215
       5       5     1.9655898e-03   1     1x     3y       7.215
       6       6    -4.2490872e-03   1     1x    17x       7.215
       7       7    -3.9172885e-03   1     1x    17z       7.215
       8       8     7.9671376e-03   4     1x     6x      14.429
       9       9    -2.3601630e-04   4     1x    34x      14.429
      10      10    -6.7104831e-02   1     1x     9x       4.418
      11      11    -4.7380801e-02   1     1x     9y       4.418
      12      12     1.3282532e-03   1     1x    10x       8.460
      13      13    -8.2116209e-04   1     1x    10y       8.460
      14      14    -6.9561306e-04   1     1x    10z       8.460
```

(continues on next page)

```
       15        15     4.5271736e-04   1    1x    26x      8.460
       16        16    -4.0571255e-03   1    1x    11x     11.118
       17        17     5.0517278e-04   1    1x    11y     11.118
       18        18    -2.0691115e-04   1    1x    25x     11.118
       19        19    -4.7362015e-04   1    1x    25z     11.118
       20        20    -1.0042599e-03   2    1x    20x     12.496
       21        21     1.2863153e-03   2    1x    20y     12.496
       22        22    -1.6874372e-04   2    1x    20z     12.496
       23        23    -2.0122242e-04   2    1x    35x     12.496
       24        24     6.0327036e-04   1    1x    28x     13.254
       25        25    -5.1714558e-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 (%) : 1.47766
```

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
  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   5.053714e+02   5.053714e+02   ␣
↪5.053714e+02
0.012316   6.132848e+00   6.132848e+00   1.033887e+01   5.052979e+02   5.052979e+02   ␣
↪5.053619e+02
0.024633   1.225469e+01   1.225469e+01   2.066997e+01   5.050779e+02   5.050779e+02   ␣
↪5.053331e+02
0.036949   1.835448e+01   1.835448e+01   3.098558e+01   5.047130e+02   5.047130e+02   ␣
↪5.052842e+02
0.049265   2.442116e+01   2.442116e+01   4.127802e+01   5.042056e+02   5.042056e+02   ␣
↪5.052137e+02
0.061582   3.044357e+01   3.044357e+01   5.153973e+01   5.035592e+02   5.035592e+02   ␣
↪5.051196e+02
0.073898   3.641049e+01   3.641049e+01   6.176326e+01   5.027783e+02   5.027783e+02   ␣
↪5.049995e+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:
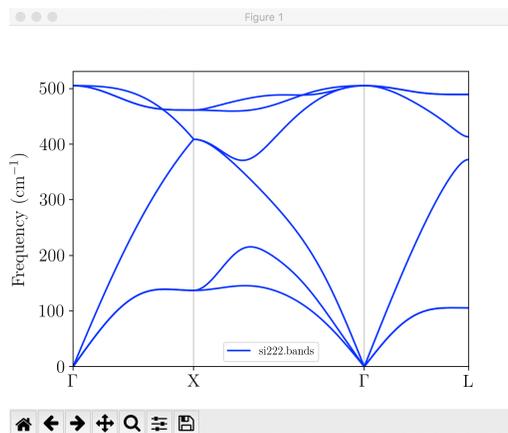
You can save the figure as png, eps, or other formats from this window. You can also change the energy unit of phonon frequency from cm^{-1} to THz or meV by the `--unit` option. For more detail of the usage of plotband.py, type

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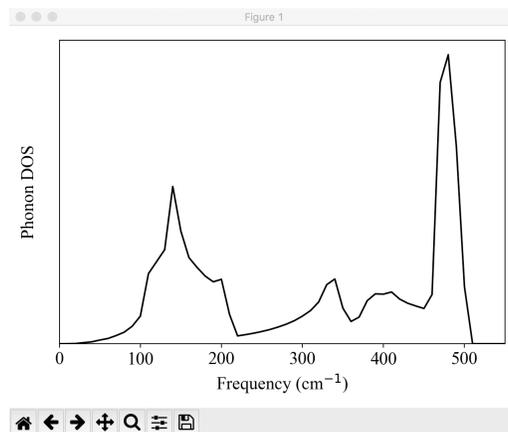
To improve the resolution of DOS, try again with a denser k grid and a smaller `DELTA_E` value.

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 (force) data to a file disp3.dat (force3.dat) as you did for harmonic IFCs in Steps 3 and 4.

Note: Since making disp3.dat and force3.dat requires moderate computational resources, you can skip this procedure by copying files reference/disp3.dat and reference/force3.dat to the working directory. The files we provide were generated by the Quantum-ESPRESSO package with `--mag=0.04`.

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

```
&fitting
  NDATA = 20
  DFILE = disp3.dat
  FFILE = force3.dat
  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 disp.dat and disp3.dat (and force files) as

```
$ cat disp.dat disp3.dat > disp_merged.dat
$ cat force.dat force3.dat > force_merged.dat
```

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

```
&fitting
  NDATA = 21
  DFILE = disp_merged.dat
  FFILE = force_merged.dat
/
```

---

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
```
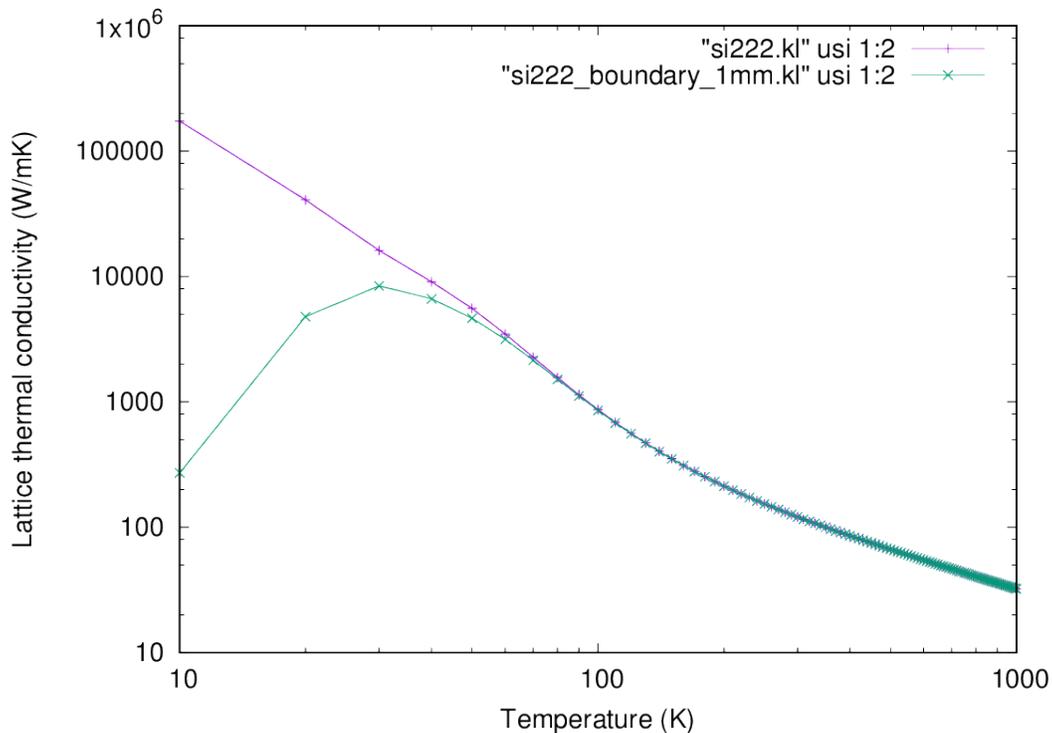

Fig. 2: Calculated lattice thermal conductivity of Si (click to enlarge)

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 `--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.dat
$ gnuplot
gnuplot> set xrange [1:]
gnuplot> set logscale y
gnuplot> set xlabel "Phonon frequency (cm^{-1})"
gnuplot> set ylabel "Phonon lifetime (ps)"
gnuplot> plot "tau300K.dat" using 3:4 w p
```

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.dat
$ gnuplot
```
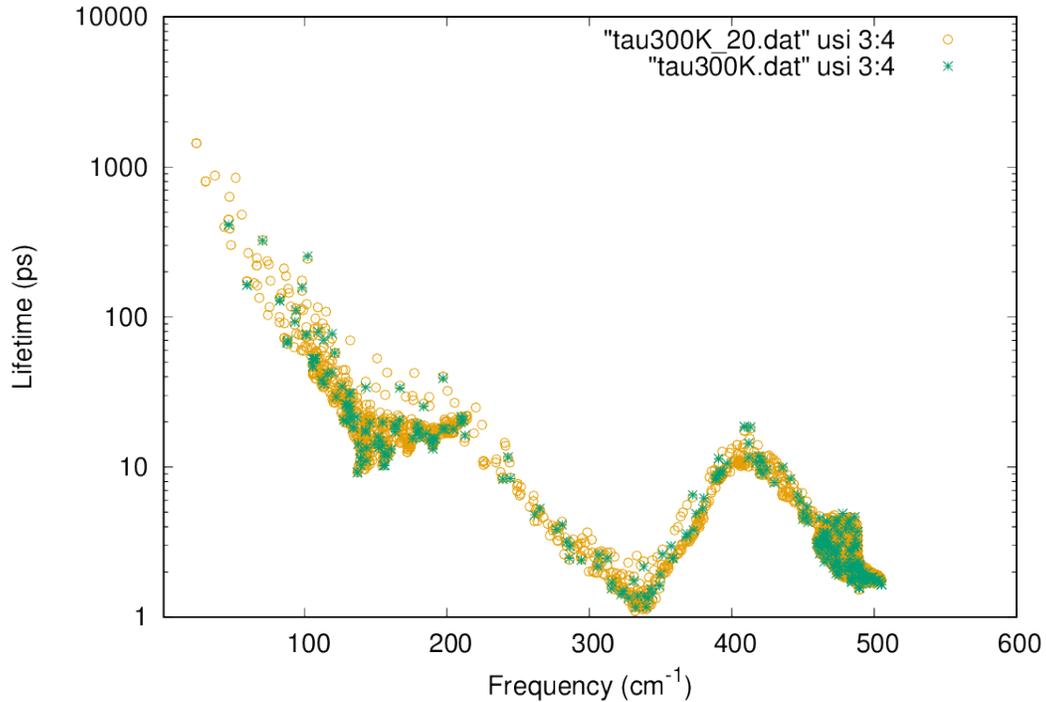
<div align="right">(continues on next page)</div>

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

```
gnuplot> set logscale x
gnuplot> set xlabel "L (nm)"
gnuplot> set ylabel "Cumulative kappa (W/mK)"
gnuplot> plot "cumulative_300K.dat" using 1:2 w lp
```

To draw a smooth line, you will 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:

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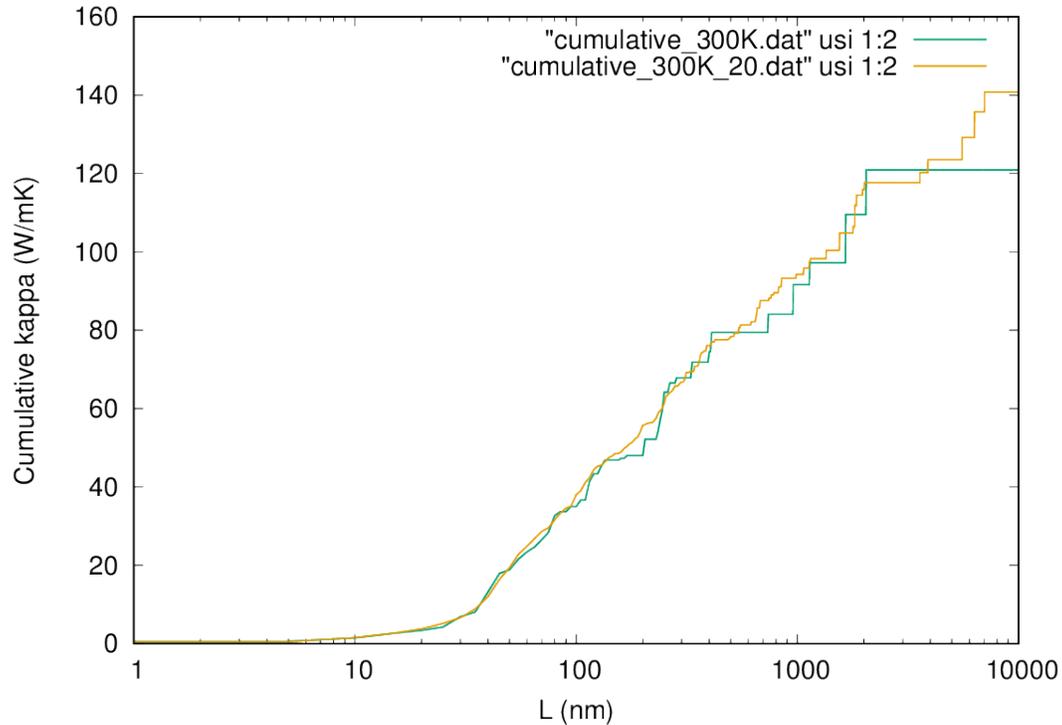
Fig. 4: Cumulative thermal conductivity of Si at 300 K (click to enlarge)

```
 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.kl_spec
$ gnuplot
gnuplot> set xlabel "Frequency (cm^{-1})"
gnuplot> set ylabel "Spectrum of kappa (W/mK/cm^{-1})"
gnuplot> plot "si222_300K.kl_spec" using 2:3 w l lt 2 lw 2
```

Fig. 5: Spectrum of thermal conductivity of Si at 300 K (click to enlarge)
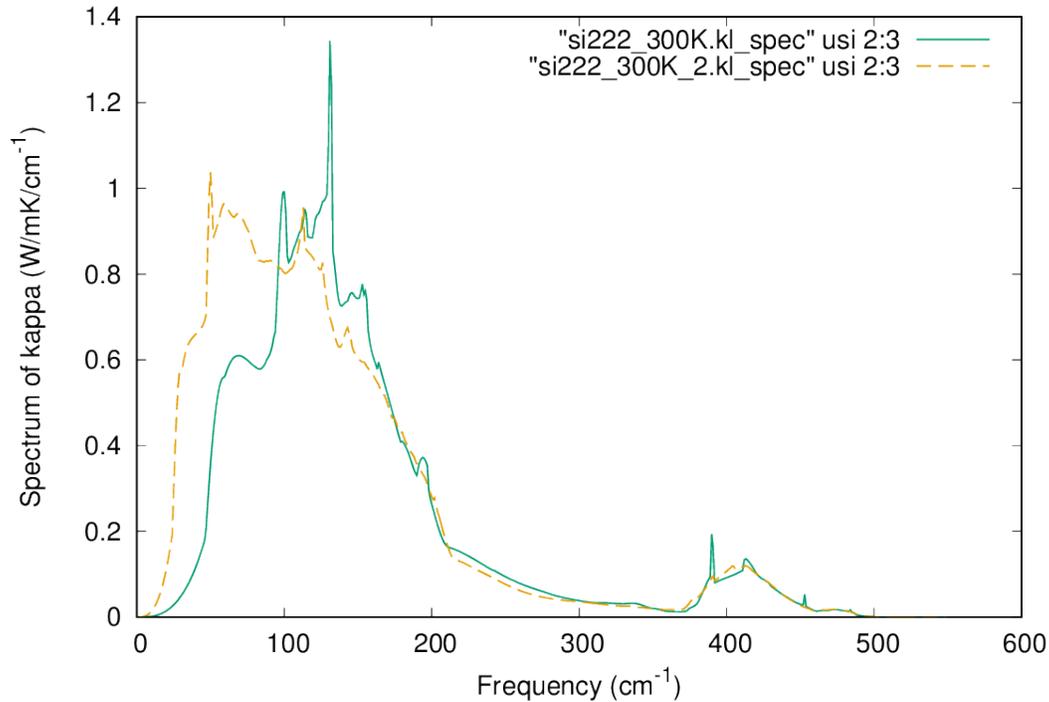
In the above figure, the computational result with $20 \times 20 \times 20$ q points is also shown by dashed line. From the figure, we can see that low-energy phonons below 200 cm^{-1} account for more than 80% of the total thermal conductivity at 300 K.

1.5.2 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 FORCE fx fy fz
dump            2 all custom 1 COORD xu yu zu
dump_modify     1 format float "%20.15f"
dump_modify     2 format float "%20.15f"
```

(continues on next page)

```
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 DISP DISP.harm1
cp FORCE FORCE.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 DISP DISP.cubic${num}
    cp FORCE FORCE.cubic${num}
done
```

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

```
$ python extract.py --LAMMPS=Si222.lammps --get=disp DISP.harm1 > disp.dat
$ python extract.py --LAMMPS=Si222.lammps --get=force FORCE.harm1 > force.dat

$ python extract.py --LAMMPS=Si222.lammps --get=disp DISP.cubic* > disp3.dat
$ python extract.py --LAMMPS=Si222.lammps --get=force FORCE.cubic* > force3.dat
```

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.

1.6 Making input files for *alm*

1.6.1 Format of input files

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

&general, **&interaction**, **&cutoff**, **&cell**, **&position**, and **&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 ("#") will be neglected. Therefore, the above example is equivalent to the following:

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

Each variable should be written inside the appropriate entry field.

1.6.2 List of input variables

"&general"-field

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

    Default None

> **Type** String

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

fitting	
	Perform fittings to estimate harmonic and anharmonic IFCs. This mode requires appropriate DFILE and FFILE.
suggest	
	This mode 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`.

- NSYM-tag = 0 | 1 | nsym

0	The program automatically generates the crystal symmetry operations (rotational and translational parts). When `PRINTSYM = 1`, symmetry operations will be saved in the file "SYMM_INFO".
1	Only the identity operation will be considered.
nsym	"nsym" symmetry operations will be read from "SYMM_INFO" file.

> **Default** 0
>
> **Type** Integer

- 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"
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` should be 1 for harmonic calculations, and 2 to include cubic terms.

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

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

    Type Array of integers

    Example If one wants to exclude three-body interactions from cubic force constants, one should explicitly give `NBODY` = 2 2.

"&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 radii of 10 a_0 will be 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 will be overwritten by a new entry that comes later.

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

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

"&fitting"-field

This field is necessary when MODE = fitting.

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

    Default None

    Type String

    Description The format of DFILE can be found *here*

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

    Default None

    Type String

    Description The format of FFILE can be found *here*

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

    Default None

    Type Integer

    Description DFILE and FFILE should contain at least NDATA× 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.

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

0	No constraints
1	Constraints for translational invariance will be imposed between IFCs.
2	In addition to `ICONST = 1`, constraints for rotational invariance will be imposed up to (`NORDER` + 1)th order.
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.
11	Same as `ICONST = 1` but the constraint is imposed algebraically rather than numerically.

Default 1

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 will be fixed upon fitting

    Default None

    Type String

    Description When `FC2XML`-tag is given, harmonic force constants will be 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 will be fixed upon fitting

Default None

Type String

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

1.6.3 Format of DFILE and FFILE

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` × `NDATA` lines in the `DFILE` without blank lines. In `FFILE`, please specify the corresponding atomic forces **in units of Ryd/Bohr**.

1.7 Making input files for *anphon*

1.7.1 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*.

1.7.2 List 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** None
  >
  > **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 will be 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.

- NSYM-tag = 0 | 1 | nsym

0	The program automatically generates the crystal symmetry operations (rotational and translational parts). When `PRINTSYM = 1`, symmetry operations will be saved in the file "SYMM_INFO_PRIM".
1	Only the identity operation will be considered.
nsym	"nsym" symmetry operations will be read from "SYMM_INFO_PRIM" file.

    Default 0

    Type Integer

- 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** None
>
> **Type** Double
>
> **Description** This entry is necessary 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 of 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 formlae. 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 = 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 = 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 = 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 = 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 = 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 = 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 = 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 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 corresponding selfenergy will be 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 0

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).

1.7.3 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.

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

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

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

1.8 List of output files

1.8.1 Output files of alm

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

    In these files, displacement patterns are printed in units of $e_{x,y,z}$. 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 ($\pm$1). Created when `MODE = fitting`.

- `PREFIX.xml`

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

1.8.2 Output files of anphon

- `PREFIX.bands`

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

- `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 $|v|$ 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 will be 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. The format is same as the `PREFIX.bands`.

- `PREFIX.scph_fc2_correction`

  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*.

1.9 Formalism for program alm

1.9.1 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

$$
U - U_0 = \sum_{n=1}^{N} U_n = U_1 + U_2 + U_3 + \cdots,
$$

$$
U_n = \frac{1}{n!} \sum_{\substack{\ell_1 \kappa_1, \ldots, \ell_n \kappa_n \\ \mu_1, \ldots, \mu_n}} \Phi_{\mu_1 \ldots \mu_n}(\ell_1 \kappa_1; \ldots; \ell_n \kappa_n) \, u_{\mu_1}(\ell_1 \kappa_1) \cdots u_{\mu_n}(\ell_n \kappa_n).
$$

(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).

1.9.2 Symmetry relationship between IFCs

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

- Permutation

  Firstly, IFCs should be invariant under the exchange of 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

  Secondly, since IFCs should depend on interatomic distances, they are invariant under a translation in units of 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:

$$
\sum_{\nu_1, \ldots, \nu_n} \Phi_{\nu_1 \ldots \nu_n}(L_1 K_1; \ldots; L_n K_n) O_{\nu_1 \mu_1} \cdots O_{\nu_n \mu_n} = \Phi_{\mu_1 \ldots \mu_n}(\ell_1 \kappa_1; \ldots; \ell_n \kappa_n),
$$

(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 current implementation of *alm*, 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

$$
\Phi_{\nu_1' \ldots \nu_n'}(L_1 K_1; \ldots; L_n K_n) = s \Phi_{\mu_1 \ldots \mu_n}(\ell_1 \kappa_1; \ldots; \ell_n \kappa_n),
$$

(1.3)

where $s = \pm 1$. 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.

1.9.3 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

$$\sum_{\ell' \kappa'} (\Phi_{\mu_1 \ldots \mu_n \nu}(\ell_1 \kappa_1; \ldots; \ell_n \kappa_n; \ell' \kappa') r_\mu(\ell' \kappa') - \Phi_{\mu_1 \ldots \mu_n \mu}(\ell_1 \kappa_1; \ldots; \ell_n \kappa_n; \ell' \kappa') r_\nu(\ell' \kappa'))$$

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

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

$$\sum_{\ell_2 \kappa_2} (\Phi_{\mu_1 \nu}(\ell_1 \kappa_1; \ell_2 \kappa_2) r_\mu(\ell_2 \kappa_2) - \Phi_{\mu_1 \mu}(\ell_1 \kappa_1; \ell_2 \kappa_2) r_\nu(\ell_2 \kappa_2))$$

$$+ \Phi_\nu(\ell_1 \kappa_1) \delta_{\mu,\mu_1} - \Phi_\mu(\ell_1 \kappa_1) \delta_{\nu,\mu_1} = 0,$$

and

$$\sum_{\ell_3 \kappa_3} (\Phi_{\mu_1 \mu_2 \nu}(\ell_1 \kappa_1; \ell_2 \kappa_2; \ell_3 \kappa_3) r_\mu(\ell_3 \kappa_3) - \Phi_{\mu_1 \mu_2 \mu}(\ell_1 \kappa_1; \ell_2 \kappa_2; \ell_3 \kappa_3) r_\nu(\ell_3 \kappa_3))$$

$$+ \Phi_{\nu \mu_2}(\ell_1 \kappa_1; \ell_2 \kappa_2) \delta_{\mu,\mu_1} - \Phi_{\mu \mu_2}(\ell_1 \kappa_1; \ell_2 \kappa_2) \delta_{\nu,\mu_1}$$

$$+ \Phi_{\mu_1 \nu}(\ell_1 \kappa_1; \ell_2 \kappa_2) \delta_{\mu,\mu_2} - \Phi_{\mu_1 \mu}(\ell_1 \kappa_1; \ell_2 \kappa_2) \delta_{\nu,\mu_2} = 0.$$

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`).

1.9.4 Estimate IFCs by least-square fitting

The code **alm** extracts harmonic and anharmonic IFCs from a displacement-force data set by solving the following linear least-square problem:

$$\text{minimize } \chi^2 = \sum_{t}^{m} \sum_{i} \| F_{i,t}^{\text{DFT}} - F_{i,t}^{\text{ALM}} \|^2. \tag{1.5}$$

Here, m is the number of atomic configurations and the index $i = (\ell, \kappa, \mu)$ is the triplet of coordinates. The model force $F_{i,t}^{\mathrm{ALM}}$ is a linear function of IFCs $\{\Phi\}$ which can be obtained by differentiating U (Eq. (1.1)) by u_i. The parameters (IFCs) are determined so as to best mimic the atomic forces obtained by DFT calculations, $F_{i,t}^{\mathrm{DFT}}$.

To evaluate goodness of fit, **alm** reports the relative error σ defined by

$$\sigma = \sqrt{\frac{\chi^2}{\sum_t^m \sum_i (F_{i,t}^{\mathrm{DFT}})^2}}, \tag{1.6}$$

where the numerator is the residual of fit and the denominator is the square sum of DFT forces.

1.10 Formalism for program anphon

1.10.1 Dynamical matrix

The dynamical matrix is given by

$$D_{\mu\nu}(\kappa\kappa'; \boldsymbol{q}) = \frac{1}{\sqrt{M_\kappa M_{\kappa'}}} \sum_{\ell'} \Phi_{\mu\nu}(\ell\kappa; \ell'\kappa') \exp\left[i\boldsymbol{q} \cdot (\boldsymbol{r}(\ell') - \boldsymbol{r}(\ell))\right], \tag{1.7}$$

where M_κ is the atomic mass of atom κ. By diagonalizing the dynamical matrix, one can obtain m $(= 3N_\kappa)$ eigenvalues $\omega_{\boldsymbol{q}j}^2$ $(j = 1, 2, \ldots, m)$ 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 (1.7), the eigenvalues may be written as

$$\omega_{\boldsymbol{q}j}^2 = (\boldsymbol{e}_{\boldsymbol{q}j}^*)^{\mathrm{T}} D(\boldsymbol{q}) \boldsymbol{e}_{\boldsymbol{q}j}. \tag{1.8}$$

Next, we introduce $m \times m$ matrices Λ and W which are defined as $\Lambda(\boldsymbol{q}) = \mathrm{diag}(\omega_{\boldsymbol{q}1}^2, \ldots, \omega_{\boldsymbol{q}m}^2)$ and $W(\boldsymbol{q}) = (\boldsymbol{e}_{\boldsymbol{q}1}, \ldots, \boldsymbol{e}_{\boldsymbol{q}m})$, respectively. Then, equation (1.8) 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

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

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

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

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

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

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.

---

[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).

1.10.2 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.

1.10.3 Thermodynamics functions

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

$$U = \frac{1}{N_q} \sum_{\boldsymbol{q},j} \hbar\omega_{\boldsymbol{q}j} \left[\frac{1}{e^{\hbar\omega_{\boldsymbol{q}j}/kT} - 1} + \frac{1}{2} \right],$$

$$C_{\mathrm{v}} = \frac{k}{N_q} \sum_{\boldsymbol{q},j} \left(\frac{\hbar\omega_{\boldsymbol{q}j}}{2kT} \right)^2 \mathrm{cosech}^2 \left(\frac{\hbar\omega_{\boldsymbol{q}j}}{2kT} \right),$$

$$S = \frac{k}{N_q} \sum_{\boldsymbol{q},j} \left[\frac{\hbar\omega_{\boldsymbol{q}j}}{kT} \frac{1}{e^{\hbar\omega_{\boldsymbol{q}j}/kT} - 1} - \log\left(1 - e^{-\hbar\omega_{\boldsymbol{q}j}/kT}\right) \right],$$

$$F = \frac{1}{N_q} \sum_{\boldsymbol{q},j} \left[\frac{\hbar\omega_{\boldsymbol{q}j}}{2} + kT \log\left(1 - e^{-\hbar\omega_{\boldsymbol{q}j}/kT}\right) \right].$$

Here, k is the Boltzmann constant. These quantities will be saved in the `PREFIX`.thermo file.

1.10.4 Mean square displacement

The mean square displacement tensor of atom κ is given by

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

$$\times \coth\left(\frac{\hbar\omega_{\boldsymbol{q}j}}{2kT} \right). \tag{1.9}$$

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

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

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

1.10.5 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

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

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

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

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.

1.10.6 (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)

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

- Atomic participation ratio (APR)

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

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.

1.10.7 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

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

---

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

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.

1.10.8 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

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

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

$$\delta D_{\mu\nu}(\kappa\kappa'; \boldsymbol{q}) = \frac{1}{\sqrt{M_\kappa M_{\kappa'}}} \sum_{\ell'} \delta\Phi_{\mu\nu}(\ell\kappa; \ell'\kappa') \exp\left[i\boldsymbol{q} \cdot (\boldsymbol{r}(\ell') - \boldsymbol{r}(\ell))\right], \tag{1.10}$$

$$\delta\Phi_{\mu\nu}(\ell\kappa; \ell'\kappa') = \sum_{\ell'', \kappa'', \lambda} \Phi_{\mu\nu\lambda}(\ell\kappa; \ell'\kappa'; \ell''\kappa'') r_\lambda(\ell''\kappa''). \tag{1.11}$$

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

1.10.9 Anharmonic self-energy

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

$$\Sigma_{\boldsymbol{q}j}(i\omega_m) = \frac{1}{2\hbar^2} \sum_{\boldsymbol{q}_1, \boldsymbol{q}_2} \sum_{j_1, j_2} |V_{-\boldsymbol{q}j, \boldsymbol{q}_1 j_1, \boldsymbol{q}_2 j_2}^{(3)}|^2$$

$$\times \left[\frac{n_1 + n_2 + 1}{i\omega_m + \omega_1 + \omega_2} - \frac{n_1 + n_2 + 1}{i\omega_m - \omega_1 - \omega_2} + \frac{n_1 - n_2}{i\omega_m - \omega_1 + \omega_2} - \frac{n_1 - n_2}{i\omega_m + \omega_1 - \omega_2} \right],$$

where $i\omega_m$ is the Matsubara frequency. In equation (1.12), we simply denoted $\omega_{\boldsymbol{q}_i j_i}$ as ω_i. The matrix element $V^{(3)}$ is given by

$$V_{\boldsymbol{q}j, \boldsymbol{q}'j', \boldsymbol{q}''j''}^{(3)} = \left(\frac{\hbar}{2N_q}\right)^{\frac{3}{2}} \frac{1}{\sqrt{\omega_{\boldsymbol{q}n}\omega_{\boldsymbol{q}'j'}\omega_{\boldsymbol{q}''j''}}} \sum_{\ell, \ell', \ell''} \exp\left[\mathrm{i}(\boldsymbol{q}\cdot\boldsymbol{r}(\ell) + \boldsymbol{q}'\cdot\boldsymbol{r}(\ell') + \boldsymbol{q}''\cdot\boldsymbol{r}(\ell''))\right]$$

$$\times \sum_{\kappa, \kappa', \kappa''} \frac{1}{\sqrt{M_\kappa M_{\kappa'} M_{\kappa''}}} \sum_{\mu, \nu, \lambda} \Phi_{\mu\nu\lambda}(\ell\kappa; \ell'\kappa'; \ell''\kappa'') e_\mu(\kappa; \boldsymbol{q}j) e_\nu(\kappa'; \boldsymbol{q}'j') e_\lambda(\kappa''; \boldsymbol{q}''j''),$$

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

$$\Gamma_{\boldsymbol{q}j}^{\mathrm{anh}}(\omega) = \frac{\pi}{2\hbar^2} \sum_{\boldsymbol{q}_1, \boldsymbol{q}_2} \sum_{j_1, j_2} |V_{-\boldsymbol{q}j, \boldsymbol{q}_1 j_1, \boldsymbol{q}_2 j_2}^{(3)}|^2$$

$$\times \left[-(n_1 + n_2 + 1)\delta(\omega + \omega_1 + \omega_2) + (n_1 + n_2 + 1)\delta(\omega - \omega_1 - \omega_2) \right.$$

$$\left. -(n_1 - n_2)\delta(\omega - \omega_1 + \omega_2) + (n_1 - n_2)\delta(\omega + \omega_1 - \omega_2) \right].$$

The computation of equation (1.12) 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`.

1.10.10 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

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

where g_2 is a dimensionless factor given by

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

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.

1.10.11 Lattice thermal conductivity

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

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

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

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

The lattice thermal conductivity will be written to the file `PREFIX`.kl.

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

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

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

1.10.12 Cumulative thermal conductivity

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

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

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:

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

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.

---

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

1.10.13 Delta function

In order 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 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.

1.10.14 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]:

$$V_{\boldsymbol{q}ij}^{[n]} = \omega_{\boldsymbol{q}i}^2 \delta_{ij} + \frac{1}{2} \sum_{\boldsymbol{q}_1,k,\ell} F_{\boldsymbol{q}\boldsymbol{q}_1,ijk\ell} \mathcal{K}_{\boldsymbol{q}_1,k\ell}^{[n-1]}. \tag{1.12}$$

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

$$\Lambda_{\boldsymbol{q}}^{[n]} = C_{\boldsymbol{q}}^{[n]\dagger} V_{\boldsymbol{q}}^{[n]} C_{\boldsymbol{q}}^{[n]},$$

where $\omega_{\boldsymbol{q}j}^{[n]} = (\Lambda_{\boldsymbol{q}jj}^{[n]})^{\frac{1}{2}}$ and $C_{\boldsymbol{q}}^{[n]}$ is the unitary matrix that transforms the harmonic phonon eigenvectors into anharmonic ones as $W_{\boldsymbol{q}}^{[n]} = W_{\boldsymbol{q}} C_{\boldsymbol{q}}^{[n]}$. The matrix $\mathcal{K}$ in Eq. (1.12) is defined as

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

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

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

The SCPH equation is solved on the irreducible q grid defined by the `KMESH_INTERPOLATE` tag. The $\boldsymbol{q}_1$ grid in Eq. (1.12), given by the `KMESH_SCPH` tag, can be finer than the q grid. After the SCPH iteration converges, the code

---

[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).

computes the anharmonic correction to the harmonic force constant $\Delta D(\boldsymbol{r}(\ell))$ as follows:

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

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

$$D_{\mathrm{SCPH}}(\boldsymbol{q}) = W_{\boldsymbol{q}} C_{\boldsymbol{q}}^{[n]} \Lambda_{\boldsymbol{q}}^{[n]} C_{\boldsymbol{q}}^{[n]\dagger} W_{\boldsymbol{q}}^{\dagger}.$$

$\Delta D(\boldsymbol{r}(\ell))$ is saved in `PREFIX.scph_fc2_correction`.

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 $\mathcal{O}(N_q^{\mathrm{irred.}} N_{q_1} m^4)$.

Indices and tables

- genindex
- modindex
- search