phonoLAMMPS Documentation

Abel Carreras

Sep 01, 2023

Table of contents

A simple LAMMPS interface with phonopy.

CHAPTER 1

Introduction

PhonoLAMMPS is a python software designed to interface between *LAMMPS* and *phonopy*. With this software allows to calculate the 2nd order interatomic force constants with phonopy using the forces obtained by LAMMPS. For this purpose phonoLAMMPS uses the official LAMMPS python API to link both LAMMPS & phonopy. This is done in a clean way without generating temporal intermediate files on the disk.

PhonoLAMMPS can be used either as a python module with a very similar interface to phonopy or via a command line interface using a provided general python script written using argparse.

PhonoLAMMPS has been tested with the following LAMMPS versions: - 16 March 2018 - 15 May 2019 - 29 Oct 2020.

1.1 Main features

- Command line interface (phonopy like style)
- Python API fully compatible with phonopy
- Use of official LAMMPS python interface
- Simple and easy to use
- New! Finite temperature force constants using DynaPhoPy

CHAPTER 2

Requirements

phonoLAMMPS requires a few python modules to be installed in your computer in order to work properly. Additionally seekpath and matplotlib are necessary to show the preview of the phonon band structure. All this packages can be downloaded easily using PyPI or conda repositories.

2.1 Mandatory requirements

- python 2.7.x/3.x
- numpy
- phonopy (https://atztogo.github.io/phonopy/)
- LAMMPS python interface (https://lammps.sandia.gov/doc/Python_library.html)

Note: LAMMPS python interface may need to be installed manually from LAMMPS source. Check LAMMPS manual for further information

2.2 Optional requirements for phonon band structure preview

- matplotlib
- seekpath (https://github.com/giovannipizzi/seekpath)

2.3 Optional requirements for finite temperature FC calculations

- DynaPhoPy (https://github.com/abelcarreras/DynaPhoPy)

CHAPTER 3

Installation

phonoLAMMPS can be installed directly from the source code (python package) or via PyPI repository.

1) From source code

```
python setup.py install --user --prefix=
```

2) From PyPI repository

```
pip install phonoLAMMPS --user
```

For convenience, you may want to copy (or link) the files inside scripts folder to a location included in **$PATH** environment variable.

Note: Depending on your configuration or OS this may be done automatically.

CHAPTER 4

Get started

4.1 Python API

phonoLAMMPS has been designed to have a very similar python interface to phonopy. For this reason the python objects generated by phonoLAMMPS can be used with phonopy functions. The procedure to obtain the force constants from LAMMPS forces is the following:

1) Loading phonoLAMMPS module

```
from phonolammps import Phonolammps
```

Creating an instance of the main phonoLAMMPS class. In this call you have to introduce a lammps input that contains the definition of the crystal unit cell the *unitcell* and *empirical potential/s*. For this you have two options:

2a. Use a LAMMPS input file (in.lammps)

```
phlammps = Phonolammps('in.lammps',
                       supercell_matrix=[[2, 0, 0],
                                         [0, 2, 0],
                                         [0, 0, 2]],
                       primitive_matrix=[[0.0, 0.5 ,0.5],
                                         [0.5, 0.0, 0.5],
                                         [0.5, 0.5, 0.0]])
```

Where *supercell_matrix* defines the supercell expansion used to calculate the force constants using the finite displacements method, and *primitive_matrix* (optional) defines the primitive cell axis matrix. If *primitive_matrix* is not defined, identity matrix is used (primitive cell = unit cell).

2b. Use a python list containing LAMMPS input commands (one command per line)

```
list_of_commands = open('in.lammps').read().split('\n')
phlammps = Phonolammps(list_of_commands,
                       supercell_matrix=[[2, 0, 0],
                                         [0, 2, 0],
                                         [0, 0, 2]])
```

(continues on next page)

(continued from previous page)

```
                        primitive_matrix=[[0.0, 0.5 ,0.5],
                                          [0.5, 0.0, 0.5],
                                          [0.5, 0.5, 0.0]])
```

This second option may be handy if you want to generate/modify LAMMPS commands in a python script.

3) Get the data needed for phonopy

```
unitcell = phlammps.get_unitcell()
force_constants = phlammps.get_force_constants()
supercell_matrix = phlammps.get_supercell_matrix()
```

4) From this you have all the information you need for phonopy calculations

```
from phonopy import Phonopy
phonon = Phonopy(unitcell,
                 supercell_matrix)

phonon.set_force_constants(force_constants)
phonon.set_mesh([20, 20, 20])

phonon.set_total_DOS()
phonon.plot_total_DOS().show()

phonon.set_thermal_properties()
phonon.plot_thermal_properties().show()
```

4.2 Command line interface

To use phonoLAMMPS from command line interface you need a LAMMPS input file containing the definition of the unit cell and potentials.

example:

```
units           metal

boundary        p p p

box tilt large

atom_style      atomic

read_data       data.si

pair_style      tersoff
pair_coeff      * * SiCGe.tersoff  Si(C)

neighbor    0.3 bin
```

Notice that run command, as well as other MD related commands (thermostat, velocities, etc..) should not be included in the input file.

Note: In this example **read data** command is used to define the atoms coordinates in a different file (refer to

LAMMPS manual for further information).

Phonolammps script uses *argparse* to provide a clean command line interface using flags All options available are displayed by using **-h**

```
$ phonolammps -h
usage: phonolammps [-h] [-o file] [--dim N N N] [-p] [-c file]
                   [-pa F F F F F F F F F] [-t F] [--optimize] [--force_tol F]
                   [--amplitude F] [--total_time F] [--relaxation_time F]
                   [--timestep F] [--logshow] [--no_symmetrize] [--use_NAC]
                   [--write_force_sets]
                   lammps_file

phonoLAMMPS options

positional arguments:
  lammps_file           lammps input file

optional arguments:
  -h, --help            show this help message and exit
  -o file               force constants output file [default: FORCE_CONSTANTS]
  --dim N N N           dimensions of the supercell
  -p                    plot phonon band structure
  -c file, --cell file  generates a POSCAR type file containing the unit cell
  -pa F F F F F F F F F, --primitive_axis F F F F F F F F F
                        primitive axis
  -t F                  temperature in K
  --optimize            optimize atoms position of unitcell
  --force_tol F         forces tolerance for optimization
  --amplitude F         displacement distance [default: 0.01 angstrom]
  --total_time F        total MD time in picoseconds [default: 20 ps]
  --relaxation_time F   MD relaxation time in picoseconds [default: 5 ps]
  --timestep F          MD time step in picoseconds [default: 0.001 ps]
  --logshow             show LAMMPS & dynaphopy log on screen
  --no_symmetrize       deactivate force constant symmetrization
  --use_NAC             include non analytical corrections (Requires BORN file
                        in work directory)
  --write_force_sets    write FORCE_SETS file
  --version             print version
```

A simple example for crystalline silicon using a 2x2x2 supercell would be

```
phonolammps in.lammps --dim 2 2 2 -pa 0.0 0.5 0.5 0.5 0.0 0.5 0.5 0.5 0.0 -c POSCAR_
↪unitcell -p
```

where **in.lammps** is a LAMMPS input containing the unit cell, *–dim* defines the supercell, *–pa* are the primitive axis in matrix format written in one line (phonopy style), *-c FILENAME* (optional) requests to write the unitcell (the same written in LAMMPS input) in VASP format on the disk to be used in phonopy calculations, and *-p* requests to show a preview of the phonon band structure in a matplotlib plot. The output of this script is a file named **FORCE_CONSTANTS** that contains the interatomic 2nd order force constants in phonopy format.

To use the obtained **FORCE_CONSTANTS** in more advanced calculations, or to have more control on the displayed data it is recommended to use PHONOPY by reading the **FORCE_CONSTANTS** file using **–readfc** option

```
phonopy --dim="2 2 2" --pa="0.0 0.5 0.5 0.5 0.0 0.5 0.5 0.5 0.0" --readfc -c POSCAR_
↪unitcell band.conf
```

CHAPTER 5

Advanced topics

5.1 Use non-analytical corrections (NAC) in phonon band structure

Non-analytical corrections (NAC) can be used in phonoLAMMPS during the plot of the phononn band structure preview. To use them a file named **BORN** containing the Born charges anf the dielectric tensor should exist in the working directory. This file is formated in phonopy format (check phonopy documentation for further information). To activate this options **–use_NAC** flag is used

```
$ phonolammps in.lammps --dim 2 2 2 --use_NAC -p
```

Notice that this option does not modify the calculated force constants written in **FORCE_CONSTANTS** file. It is only used in the phonon band structure plot, therefore it makes no effect without **-p** flag.

5.2 Atomic position optimization

The atomic positions of the unit cell can be optimized using **–optimization** flag. This uses LAMMPS minimize to perform a minimization of atomic forces at constant volume. The forces tolerance for the convergence criterion is defined by **–force_tol** flag.

```
$ phonolammps in.lammps --dim 2 2 2 --optimization --force_tol 1e-10 -p
```

This option performs a simple minimization. If a more sophisticated optimization is required then use LAMMPS directly. This option is best used to refine the atomic positions of an already optimized unit cell.

5.3 Finite temperature force constants

Finite temperature force constants can be calculated from molecular dynamics using dynaphopy software (http://abelcarreras.github.io/DynaPhoPy). This software implements the normal mode projection technique to obtain the renormalized force constants at finite temperature based on quasiparticle theory. Phonolammps provide a minimum

functionality to make this process automatic using both LAMMPS and dynaphopy python API. To use this feature dynaphopy must be installed (for further details check dynaphopy documentation).

In command line script temperature is defined by **-t** flag. By default this value is 0 and usual 2n order force constants are calculated. If the temperature is higher than 0 then a molecular dynamics (MD) simulation is calculated with LAMMPS using a supercell defined by **–dim** flag. By default the length of the MD if 20 ps with time step of 0.001 ps and a relaxation time of 5 ps, but these parameters can be tweaked using **–total_time**, **–relaxation_time** and **–relaxation_time** flags.

example for silicon:

```
$ phonolammps in.lammps --dim 2 2 2 -pa 0.0 0.5 0.5 0.5 0.0 0.5 0.5 0.5 0.0 -t 300 -p
```

to have more control over the simulation and the renormalization precedure you will have to use the two software separately.

CHAPTER 6

API

This is the Python API for phonoLAMMPS

class phonolammps.**PhonoBase**

Base class for PhonoLAMMPS This class is not designed to be called directly. To use it make a subclass and implement the following methods:

- __init__()
- get_forces()

get_force_constants(*include_data_set=False*)

calculate the force constants with phonopy using lammps to calculate forces

Returns ForceConstants type object containing force constants

get_path_using_seek_path()

Obtain the path in reciprocal space to plot the phonon band structure

Returns dictionary with list of q-points and labels of high symmetry points

get_phonopy_phonon()

Return phonopy phonon object with unitcell, primitive cell and the force constants set.

Returns

get_supercell_matrix()

Get the supercell matrix

Return supercell the supercell 3x3 matrix (list of lists)

get_unitcell()

Get unit cell structure

Return unitcell unit cell 3x3 matrix (lattice vectors in rows)

plot_phonon_dispersion_bands(*bands_and_labels=None*)

Plot phonon band structure using seekpath automatic k-path Warning: The labels may be wrong if the structure is not standarized

Parameters **bands_and_labels** – Custom energy band path

write_force_constants(*filename='FORCE_CONSTANTS'*, *hdf5=False*)
Write the force constants in a file in phonopy plain text format

Parameters **filename** – Force constants filename

write_force_sets(*filename='FORCE_SETS'*)
Write the force sets in a file in phonopy plain text format

Parameters **filename** – Force sets filename

write_unitcell_POSCAR(*filename='POSCAR'*)
Write unit cell in VASP POSCAR type file

Parameters **filename** – POSCAR file name (Default: POSCAR)

class phonolammps.**PhonoGromacs**(*gro_file, supercell_matrix=array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]]), primitive_matrix=array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]]), displacement_distance=0.01, show_log=False, show_progress=False, use_NAC=False, symmetrize=True, gmx_params=None, base_ff='charmm27.ff/forcefield.itp', itp_file=None, silent=True, omp_num_threads=1*)

cell_angles(*cell*)
Get the angles between the cell lattice vectors in degrees.

cell_lengths(*cell*)
Get the lengths of cell lattice vectors in angstroms.

get_forces(*cell_with_disp*)
Calculate the forces of a supercell using tinker :param cell_with_disp: supercell (PhonopyAtoms) from which determine the forces :return array: numpy array matrix with forces of atoms [Natoms x 3]

class phonolammps.**PhonoTinker**(*txyz_input_file, key_input_file, force_field_file, supercell_matrix=array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]]), primitive_matrix=array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]]), displacement_distance=0.01, show_log=False, show_progress=False, use_NAC=False, symmetrize=True*)

get_forces(*cell_with_disp*)
Calculate the forces of a supercell using tinker :param cell_with_disp: supercell (PhonopyAtoms) from which determine the forces :return array: numpy array matrix with forces of atoms [Natoms x 3]

class phonolammps.**Phonolammps**(*lammps_input, supercell_matrix=array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]]), primitive_matrix=array([[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]]), displacement_distance=0.01, show_log=False, show_progress=False, use_NAC=False, symmetrize=True*)

get_forces(*cell_with_disp*)
Calculate the forces of a supercell using lammps

Parameters **cell_with_disp** – supercell from which determine the forces

Returns numpy array matrix with forces of atoms [Natoms x 3]

get_units(*commands_list*)
Get the units label for LAMMPS "units" command from a list of LAMMPS input commands

Parameters **commands_list** – list of LAMMPS input commands (strings)

Return units string containing the units

optimize_unitcell(*energy_tol=0*, *force_tol=1e-10*, *max_iter=1000000*, *max_eval=1000000*)

Optimize atoms position of the unitcell using lammps minimizer. Check https://docs.lammps.org/minimize.html for details

Parameters

- **energy_tol** – stopping tolerance for energ
- **force_tol** – stopping tolerance for force (force units)
- **max_iter** – max iterations of minimizer
- **max_eval** – max number of force/energy evaluations

end

CHAPTER 7

Troubleshooting

7.1 Check LAMMPS python API

If there is some problem with LAMMPS installation this is a simple script that can help you to find it out. Run this script inside one of the example folders (where in.lammps file is placed)

```
from lammps import lammps

lmp1 = lammps()
lmp1.file("in.lammps")
lmp1.close()
```

if everything works as expected you should get an output like this

```
LAMMPS (15 May 2019)
OMP_NUM_THREADS environment is not set. Defaulting to 1 thread. (src/comm.cpp:88)
  using 1 OpenMP thread(s) per MPI task
Lattice spacing in x,y,z = 4.785 2.76262 5.189
Created triclinic box = (0 0 0) to (3.19 2.76262 5.189) with tilt (-1.595 0 0)
WARNING: Triclinic box skew is large (src/domain.cpp:194)
  1 by 1 by 1 MPI processor grid
Created 4 atoms
  create_atoms CPU = 0.00049852 secs
Reading potential file GaN.tersoff with DATE: 2007-10-22
Total wall time: 0:00:00
```

otherwise there may be some trouble with LAMMPS python interface. Check LAMMPS manual for further information.

7.2 Check LAMMPS calculations log

By default LAMMPS logs are deactivated and not shown during the calculation. If issues appear it may be useful to check LAMMPS force calculations logs. This is done by using **-logshow** flag. Ex:

```
$ phonolammps in.lammps --dim 2 2 2 --logshow
```

Python Module Index

p

Index