aiida-phonopy Documentation

Release 0.1

Abel Carreras

Nov 09, 2019

Contents

CHAPTER 1

Installation

```
pip install -e aiida-phonopy
```

1.1 Requirements

- AiiDA v1.0.0b6
- phonopy v2.3.2
- seekpath

Currently only aiida-vasp can be used as the calculator plugin.

CHAPTER 2

Introduction

This package is a collection of tools to perform phonon calculations using AiiDA. These tools include a phonopy plugin, data objects and workchains to calculate the phonon band structure, the thermal properties and mode Gruneisen parameters.

2.1 Get started

CHAPTER 3

Data objects

This is a collection of data objects for AiiDA used in phonopy plugin and WorkChains.

3.1 Band structure

This object contains the phonon band structure calculated by phonopy. Also it can store the mode Gruneisen parameters data.

Special setters are provided to store the data directly from phonopy objects:

Also special getters are provided to return data in phonopy format:

3.2 Force constants

This object contains the second order force constants as a numpy array.

Setters and getters are provided to store and get the data in phonopy format:

3.2.1 example of use

```
force_constants = phonon.get_force_constants()
force_constants_data = ForceConstantsData(data=force_constants)

...

phonon.set_force_constants(force_constants_data.get_data())
```

3.3 Force sets

This object contains the information about the supercells with displacements generated by phonopy (displacement, directions and atomic forces). Atomic forces are optional. This can be used to store either 2nd order or 3rd order displacements data. When containing 3rd order displacements data, it also can be used for 2nd order phonon calculations.

Setters and getters are provided to store and get the data in phonopy format:

3.3.1 example of use in phonopy

```
data_sets = phonon.get_displacement_dataset()
data_sets_data = ForceSetsData(data_sets=data_sets)

...

force_sets_data = ForceSetsData(data_sets=data_sets_data.get_data_sets())

forces = []
for i in range(force_sets_data.get_number_of_displacements()):
    forces.append(forces_of_supercell_with_displacement[i])

force_sets_data.set_forces(forces)

...

phonon.set_displacement_dataset(force_sets_data.get_force_sets())
phonon.produce_force_constants()
```

3.4 Phonon density of states

This object contains the phonon band structure calculated by phonopy. Also stores the mode Gruneisen parameters data in Gruneisen calculation.

Special setters are provided to store the data directly from phonopy objects:

3.5 Non-analytical corrections (NAC)

This object contains the information needed to calculate the non-analytical corrections in phonopy phonon calculations. This includes the crystal structure, Born effective charges and dielectric tensor.

Setters and getters are provided to store the data from aiida objects and get the data in phonopy format:

3.5.1 example of use

Setup the nac_data object from calculation output data

```
structure = StructureData()
born_charges = NumpyArray()
dielectric_tensor = NumpyArray()
```

(continues on next page)

(continued from previous page)

```
nac_data = NacData(structure=structure,
                   born_charges=born_charges,
                   epsilon=dielectric_tensor)
```

Use this object to setup a phonopy calculation

```
phonon = Phonopy()
primitive = phonon.get_primitive()
nac_parameters = nac_data.get_born_parameters_phonopy(primitive_cell=primitive.get_
↪cell())
phonon.set_nac_params(nac_parameters)
```

CHAPTER 4

Codes

These are the plugins for AiiDA to calculate phonon using phonopy.

4.1 Phonopy

This plugin is designed to calculate the harmonic force constants, thermal properties (entropy, free energy and heat capacity at constant volume) and density of states using phonopy.

PhonopyCalculation (*structure*, *parameters*, *data_sets*, *nac_data*)

Parameters

- **structure** – StructureData object that contains the crystal unit cell information
- **parameters** – Dict object that contains the phonopy input parameters
- **data_sets** – ForceSetsData object that contains the forces, directions and detail of all the supercells with displacements (equivalent to FORCE_SETS file in phonopy)
- **force_constants** – AiiDA ForceConstants object that contains the force constants
- **nac_data** – (optional) NacData object that contains the Born effective charges and the dielectric tensor

- input parameters should be dictionary with the following entries

```
parameters_dict = {'supercell': [[2, 0, 0],
                                 [0, 2, 0],
                                 [0, 0, 2]],
                   'primitive': [[1.0, 0.0, 0.0],
                                 [0.0, 1.0, 0.0],
                                 [0.0, 0.0, 1.0]],
                   'distance': 0.01,
                   'mesh': [40, 40, 40],
                   'symmetry_tolerance': 1e-5}
```

(continues on next page)

(continued from previous page)

```
Dict = DataFactory('dict')
parameters = Dict(dict=parameters_dict)
```

Either data_sets of force_constants should be used. If data_sets is used force constants will be calculated and returned as a calculation output. If force_constants is used the calculation will be faster.

- data_sets can be created from a phonopy FORCE_SETS file by

```
ForceSetsData = DataFactory('phonopy.force_sets')
force_sets = ForceSetsData()
force_sets.read_from_phonopy_file('FORCE_SETS')
```

- force_constants can be create from a phonopy FORCE_CONSTANTS file by

```
ForceConstantsData = DataFactory('phonopy.force_constants')
force_constants = ForceConstantsData()
force_constants.read_from_phonopy_file('FORCE_CONSTANTS')
```

- nac_data is an optional parameter and can be created from single point calculation, the required information is the crystal structure(StructureData) used in the calculation, born effective charges (numpy array) for all the atoms in the crystal structure and the dielectric tensor (numpy array [dim: Natoms x 3 x 3])

```
nac_data = NacData(structure=crystal_structure,
                   born_charges=born_charges_numpy_array,
                   epsilon=epsilon_numy_array)
```

The outputs of this plugin are:

- **force_constants**: ForceConstantsData object that contains the second order force constants.
- **dos**: DosData object that contains the full and partial density of states.
- **band_structure**: BandStructureData object that contains the phonon band structure.
- **thermal_properties**: ArrayData object that contains the entropy, free energy and heat capacity at constant volume.

Take a look at the examples in examples/plugins folder for reference

4.2 Phono3py

This plugin is designed to calculate the thermal conductivity using phono3py.

`Phono3pyCalculation` (*structure*, *parameters*, *data_sets*, *nac_data*)

Parameters

- **`structure`** – StructureData object that contains the crystal unit cell information
- **`parameters`** – Dict object that contains the phonopy input parameters
- **`data_sets`** – ForceSetsData object that contains the forces, directions and detail of all the supercells with displacements (equivalent to FORCE_SETS file in phonopy)
- **`force_constants`** – ForceConstants object that contains the 2nd order force constants
- **`force_constants_3`** – ForceConstants object that contains the 3rd order force constants

- **nac_data** – (optional) NacData object that contains the Born effective charges and the dielectric tensor

- input parameters should be dictionary with the following entries

```
parameters_dict = {'supercell': [[2, 0, 0],
                                 [0, 2, 0],
                                 [0, 0, 2]],
                   'primitive': [[1.0, 0.0, 0.0],
                                 [0.0, 1.0, 0.0],
                                 [0.0, 0.0, 1.0]],
                   'distance': 0.01,
                   'mesh': [40, 40, 40],
                   'symmetry_tolerance': 1e-5}
Dict = DataFactory('dict')
parameters = Dict(dict=parameters_dict)
```

Either data_sets of force_constants/force_constants_3 should be defined. If data_sets is used force constants will be calculated during the thermal conductivity calculation.

The outputs of this plugin are:

- **kappa**: ArrayData object that contains the results stored in kappa-mxxx-gx.hdf5 output file.

CHAPTER 5

Workchains

This is a collection of workchains for AiiDA to do phonon calculations using phonopy.

5.1 Phonon

This WorkChain performs a phonon calculation using Phonopy. This WorkChain requires at least one of the following AiiDA plugins to work: QuantumESPRESSO (https://github.com/aiidateam/aiida-quantumespresso), VASP (https://github.com/DropD/aiida-vasp) or LAMMPS (https://github.com/abelcarreras/aiida-lammps). At the present time Born effective charges are only calculated using VASP plugin. Phonopy can be used either locally and remotely. To use phonopy remotely phonopy code must be setup as described in AiiDA documentation (https://aiida-core.readthedocs.io/en/latest/get_started/index.html#code-setup-and-configuration). using the phonopy plugin provided in this package.

PhononPhonopy (*structure*, *ph_settings*, *es_settings*[, *optimize=True*, *pressure=0.0*, *use_nac=False*])

Parameters

- **structure** – StructureData object that contains the crystal unit cell structure.
- **ph_settings** – Dict object that contains the phonopy input parameters.
- **es_settings** – Dict object that contains the calculator input parameters. These parameters depends on the code used (see workchains/launcher examples)
- **optimize** – (optional) BooleanData object. Determines if a crystal unit cell optimization is performed or not before the phonon calculation. By default this option is True.
- **pressure** – (optional) FloatData object. If optimize is True, this sets the external pressure (in kB) at which the unit cell optimization is preformed. By default this option takes value 0 kB.
- **use_nac** – (optional) BooleanData object. Determines if non-analytical corrections will be included in the phonon calculation. By default this option is False.

- ph_settings: This object contains a dictionary with all input parameters for phonopy. See plugins section for more information. Additional dictionary entries can be added to request a remote phonopy calculation. See example in examples/workchains/launh_phonon_gan

```
code: phonopy@cluster
machine: machine_dict
```

machine_dict dictionary should contain the following entries. resources_dict may change depending on the scheduler

```
machine_dict = {'resources': resources_dict
                'max_wallclock_seconds': 3600 * 10 # in seconds
                }

resources_dic = {'num_machines': 1,
                 'parallel_env': 'mpi*',
                 'tot_num_mpiprocs': 16
                 }
```

- es_settings: This object contains the parameters for each specific software used as calculator (QE, VASP or LAMMPS). Each calculator uses a different dictionary structure (See examples at examples/workchains folder for the details). The common basic structure in all calculators is

```
settings_dict = {'code': {'optimize': 'vasp5@boston',
                          'forces': 'vasp4@boston',
                          'born': 'vasp4@boston'},

                 'parameters': parameters, # this depends on calculator (see␣
↪examples)
                 'machine': machine_dict, # same dictionary defined above
                 ...
                 }

es_settings = Dict(dict=settings_dict)
```

If the code used in all calculations types (optimize, forces and born) is the same, the dictionary can be written as

```
settings_dict = {'code': vasp@boston',
                 'parameters': parameters,
                 'machine': machine_dict,
                 ...
                 }
```

The results outputs of this WorkChain are the following :

- **force_constants**: ForceConstantsData object that contains the harmonic force constants. If Born effective charges are calculated this object also contains the dielectric tensor and the born effective charges of each atom in the unit cell
- **thermal_properties**: ArrayData object that contains the thermal properties calculated using phonopy. These include the entropy, free energy and heat capacity at constant volume.
- **dos**: PhononDosData object that contains the phonon full and partial density of states.
- **band_structure**: BandStructureData object that contains the harmonic phonon band structure.
- **final_structure**: StructureData object that contains the optimized unit cell. If no optimization is performed this is the same StructureData object provided as a input.

Each one of this objects has its own methods for extracting the information. Check the individual object documentation for details. **workchains/tools/plot_phonon.py** contains a complete example script showing how to extract the

information from these outputs.

5.1.1 example of use

```
PhononPhonopy = WorkflowFactory('phonopy.phonon')
```

Run in interactive

```
result = run(PhononPhonopy,
             structure=structure,
             es_settings=es_settings,
             ph_settings=ph_settings,
             pressure=Float(0.0),
             optimize=Bool(True),
             use_nac=Bool(True)
             )
```

Run by the deamon

```
future = submit(PhononPhonopy,
                structure=structure,
                es_settings=es_settings,
                ph_settings=ph_settings,
                pressure=Float(0),
                optimize=Bool(True),
                use_nac=Bool(True)
                )
print('Running WorkChain with pk={}'.format(future.pid))
```

5.2 Gruneisen

This WorkChain performs a mode Gruneisen parameters calculation using Phonopy. This WorkChain is designed to keep the compatibility with phonon WorkChain input structure. For this reason the input parameters in the WorkChain have the same structure. Please, check phonon Workchain documentation for detailed information. This WorkChain calculates the mode Gruneisen parameters by performing 3 different phonon calculations. One with the crystal structure optimized at a given external stress (default: 0 kB) and the other two with the unit cell optimized with a slightly higher and lower stress (defined by stress_displacement) obtaining a slightly smaller and larger unit cell respectively. Stress_displacement can be set as an optional argument, by default its value is 1e-2 kB.

GruneisenPhonopy (*structure*, *ph_settings*, *es_settings*[, *stress_displacement=1e-2*, *use_nac=False*])

Parameters

- **structure** – StructureData object that contains the crystal unit cell structure.
- **ph_settings** – Dict object that contains the phonopy input parameters.
- **es_settings** – Dict object that contains the calculator input parameters.
- **pressure** – (optional) FloatData object. This determines the absolute stress (in kBar) at which the reference crystal structure is optimized (default 0).
- **stress_displacement** – (optional) FloatData object. This determines the stress difference between the 3 phonon calculations (default 1e-2 kB).
- **use_nac** – (optional) BooleanData object. Determines if non-analytical corrections will be included in the phonon calculations. By default this option is False.

The outputs of this WorkChain are:

- **band_structure**: BandStructure object that contains the phonon band structure and the mode Gruneisen parameters.
- **mesh**: ArrayData object that contains the wave vectors sampling mesh and the mode Gruneises parameters at each wave vector.

Each one of this objects has its own methods for extracting the information. Check the individual object documentation for more details. **workchains/tools/plot_gruneisen.py** contains a complete example script showing how to extract the information from these outputs.

5.2.1 example of use

```
GruneisenPhonopy = WorkflowFactory('phonopy.gruneisen')
```

Run in interactive

```
result = run(GruneisenPhonopy,
             structure=structure,
             es_settings=es_settings,
             ph_settings=ph_settings,
             pressure=Float(0),
             stress_displacement=Float(1e-2),
             use_nac=Bool(True)
             )
```

Run by the deamon

```
future = submit(GruneisenPhonopy,
                structure=structure,
                es_settings=es_settings,
                ph_settings=ph_settings,
                pressure=Float(0),
                stress_displacement=Float(1e-2),
                use_nac=Bool(True)
                )

print('Running WorkChain with pk={}'.format(future.pid))
```

5.3 QHA

This WorkChain performs a quasi-harmonic approximation calculation using Phonopy. This WorkChain requires one of the plugins for VASP, QuantumESPRESSO and LAMMPS described in phonon WorkChain. Non-analytical corrections can be included from the Born effective charges and dielectric tensor which are only implemented for VASP plugin. Phonopy can be used either locally and remotely. To use phonopy remotely phonopy code must be setup as described in AiiDA documentation (https://aiida-core.readthedocs.io/en/latest/get_started/index.html#code-setup-and-configuration). using the phonopy plugin provided in this package.

QHAPhonopy (*structure*, *ph_settings*, *es_settings*[, *optimize=True*, *use_nac=False*, *num_expansions=10*])

Parameters

- **structure** – StructureData object that contains the crystal unit cell structure.
- **ph_settings** – Dict object that contains the phonopy input parameters.

- **es_settings** – Dict object that contains the calculator input parameters. These parameters depends on the code used (see workchains/launcher examples)
- **num_expansions** – (optional) IntData object. The number of volume expansions around the optimized structure at zero pressure to perform. By default the value is 10.
- **use_nac** – (optional) BooleanData object. Determines if non-analytical corrections will be included in the phonon calculations. By default this option is False.

The results outputs of this WorkChain are the following :

- **qha_results**: ArrayData object that contains the thermal properties at constant pressure calculated using phonopy. This ArrayData includes the following arrays: qha_temperatures, helmholtz_volume, thermal_expansion, volume_temperature, heat_capacity_P_numerical, volume_expansion and gibbs_temperature.

5.4 Phonon3

This WorkChain calculated the 3rd order force constants using phono3py. This WorkChain requires one of the plugins for VASP, QuantumESPRESSO and LAMMPS described in phonon WorkChain. Non-analytical corrections can be calculated from the Born effective charges and dielectric tensor which are only implemented for VASP plugin.

Phonon3py (*structure*, *ph_settings*, *es_settings*[, *optimize=True*, *use_nac=False*, *pressure= 0.0*, *calculate_fc=False*])

Parameters

- **structure** – StructureData object that contains the crystal unit cell structure.
- **ph_settings** – Dict data object that contains the phonopy input parameters.
- **es_settings** – Dict object that contains the calculator input parameters. These parameters depends on the code used (see workchains/launcher examples)
- **use_nac** – (optional) BooleanData object. Determines if non-analytical corrections will be included in the phonon calculations. By default this option is False.
- **optimize** – (optional) BooleanData object. Determines if a crystal unit cell optimization is performed or not before the phonon calculation. By default this option is True.
- **pressure** – (optional) FloatData object. If optimize is True, this sets the external pressure (in kB) at which the unit cell optimization is preformed. By default this option takes value 0 kB.
- **calculate_fc** – (optional) BooleanData object. Determines if the 2on and 3rd order force constants are calculated. By default this option is False.
- **chunks** – (optional) Int object that defines the maximum number of calculation to submit simultaneously. The next set of calculation will not be submitted until the previous set is finished.
- **data_sets** – (optional) ForceSets object that contains the forces and displacements of a previously calculation. This data_set can be the output of either phonon3 or phonon WorkChains.

The results outputs of this WorkChain are the following :

- **data_sets**: ForceSetsData object that contains the information of supercells with displacements and forces. Check
- **force_constants_2order**: ForceConstantsData object that contains the 2ond order force constants.
- **force_constants_3order**: ForceConstantsData object that contains the 3rd order force constants.

- **final_structure**: StructureData containing the optimized structure.

The ForceSetsData object obtained as a output of phonon3 WorkChain can be also in harmonic phonon calculation without modification.

5.5 Phono3pyDist

This WorkChain is designed to do a phono3py calculation using a distributed workload scheme as described in phono3py manual (https://atztogo.github.io/phono3py/workload-distribution.html). The input nodes have the same name as the usual phono3py calculation. An additional node that defines the number of computers in which distribute the calculation is defined in this WorkChain.

Phono3pyDist (*structure*, *parameters*[, *data_sets=None*, *nac_data=None*, *force_constants=None*, *force_constants_3=None*, *gp_chunks=10*])

Parameters

- **structure** – StructureData object that contains the crystal unit cell information
- **parameters** – Dict data object that contains the phonopy input parameters
- **data_sets** – ForceSetsData object that contains the forces, directions and detail of all the supercells with displacements (equivalent to FORCE_SETS file in phonopy)
- **force_constants** – ForceConstantsData object that contains the 2nd order force constants
- **force_constants_3** – ForceConstantsData object that contains the 3rd order force constants
- **nac_data** – (optional) NacData object that contains the Born effective charges and the dielectric tensor
- **gp_chunks** – (optional) Int object that defines the number of computers in which the calculation will be distributed

The results outputs of this WorkChain are the following :

- **kappa**: ArrayData object that contains the results stored in kappa-mxxx-gx.hdf5 output file.

5.6 Thermal_conductivity

This WorkChain is designed to calculate the thermal conductivity with phono3py using an interative procedure by increasing the cutoff distance. This WorkChain makes use of phonon WorkChain to calculate the harmonic band structure and DOS and determine the dynamical stability of the crystal structure. If the crystal structure is stable then it calculates the thermal conductivity using phono3py using a small cutoff distance (defined by the user) and successively increase it until reaching convergence. The convergence criteria are given by the user as atol and rtol parameters. The convergence criteria works in the same manner of numpy allclose function (https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.allclose.html), applied to all values of kappa vs temperature.

ThermalPhono3py (*structure*, *ph_settings*, *es_settings*[, *optimize=True*, *use_nac=False*, *pressure= 0.0*, *calculate_fc=False*, *gp_chunks=1*, *gp_chunks=10*, *initial_cutoff=2.0*, *step=1.0*, *atol=0.1*, *rtol=0.3*])

Parameters

- **structure** – StructureData object that contains the crystal unit cell structure.
- **ph_settings** – Dict object that contains the phonopy input parameters.

- **es_settings** – Dict object that contains the calculator input parameters. These parameters depends on the code used (see workchains/launcher examples)
- **use_nac** – (optional) BooleanData object. Determines if non-analytical corrections will be included in the phonon calculations. By default this option is False.
- **optimize** – (optional) BooleanData object. Determines if a crystal unit cell optimization is performed or not before the phonon calculation. By default this option is True.
- **pressure** – (optional) FloatData object. If optimize is True, this sets the external pressure (in kB) at which the unit cell optimization is preformed. By default this option takes value 0 kB.
- **chunks** – (optional) Int object that defines the maximum number of calculation to submit simultaneously. The next set of calculation will not be submitted until the previous set is finished.
- **gp_chunks** – (optional) Int object that defines the number of computers in which the calculation will be distributed (default: 1).
- **initial_cutoff** – (optional) Float object that defines initial cutoff distance for phono3py calculation.
- **step** – (optional) Float object that defines increment of cutoff distance in each iteration.
- **atol** – (optional) Float object that defines the convergence criteria of thermal conductivity, absolute value (thermal conductivity units).
- **rtol** – (optional) Float object that defines the convergence criteria of thermal conductivity, relative value.

The results outputs of this WorkChain are the following :

- **kappa**: ArrayData object that contains the results stored in kappa-mxxx-gx.hdf5 output file.
- **final_structure**: StructureData object that contains the optimized unit cell. If no optimization is performed this contains the same StructureData object provided as a input.
- **thermal_properties**: ArrayData object that contains the harmonic thermal properties calculated using phonopy. These include the entropy, free energy and heat capacity at constant volume.
- **dos**: PhononDosData object that contains the harmonic phonon full and partial density of states.
- **band_structure**: BandStructureData object that contains the harmonic phonon band structure.

Index