ase-espresso Documentation

Release 0.3.4

Lukasz Mentel

Feb 27, 2021

Contents

ase-espresso is a python interface for Quantum Espresso using the Atomic Simulation Environment (ASE).

The main purpose of the ase-espresso interface is to allow for python-controlled ionic updates (e.g. ase-based structural relaxation) and to provide post-processed Quantum Espresso output (e.g. charge densities, DOS) as numpy arrays. While the ase-espresso interface can be used to create input files for Quantum Espresso only, there are alternative python interfaces for input file generation (or for running static calculations ionic step by ionic step):

- ase_qe_intrfce,
- PWscfInput,
- qecalc,
- qecalc by P. T. Jochym,
- espresso,

Contents:

CHAPTER 1

Dependencies

- ASE
- numpy
- pexpect
- future
- path.py
- hostname

CHAPTER 2

Installation

The recommended installation method is with pip and can be installed directly from the ase_espresso repository:

```
pip install git+git://github.com/lmmentel/ase-espresso.git
```

or cloned first

```
git clone https://github.com/lmmentel/ase-espresso.git
```

and installed via

```
pip install ./ase-espresso
```

You can verify that your installation was successful by opening a python console and trying to import `Espresso`:

```
>>> from espresso import Espresso
```

CHAPTER 3

Configuration

To run properly ase-espresso requires that the Quantum Espresso code is properly compiled and the executables are available to the shell. You can to that by extending the `PATH` variable with the location of your Quantum Espresso

```
export PATH=$PATH:/path/to/your/quantum-espresso/executables
```

Another thing that is required is setting the environmental variable with the path to the directory containing pseudopotential files

```
export ESP_PSP_PATH=/path/to/pseudo/pseudopotentials
```

CHAPTER 4

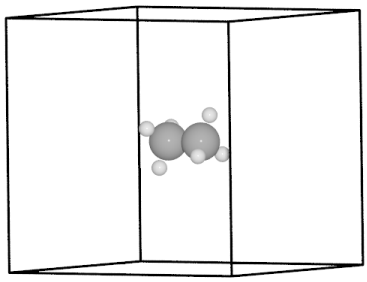
NEBEspresso

Derived from the `ase.neb.NEB` class, `NEBEspresso` orchestrates (i)Espresso calculators of individual images in a NEB calculation. This way a `NEBEspresso` object facilitates parallel NEB jobs, where each image calculator uses a subset of the pool of CPUs available to the job.

This tutorial explains the use of `NEBEspresso` to do a NEB calculation on methyl group rotation in ethane, built on this ASE example.

4.1 Initial and final structures of the NEB

As opposed to the Effective Medium Potential used in the original example, Quantum Espresso needs periodic boundaries defined by a unit cell. That is defined in the `ethane.traj` trajectory file containing the optimized structure. The final structure is created simply by permuting the H atoms in one of the methyl groups.



```
from espresso import iEspresso
from ase.io import read

initial = read('ethane.traj')
initial.get_potential_energy()
final = initial.copy()
final.positions[2:5] = initial.positions[[3, 4, 2]]
final.get_potential_energy()
```

4.2 Assigning calculators to intermediate images

Initially, images are created as copies of the initial structure and are each asssigned a calculator. Here we use the interactive `iEspresso` calculator, but the non-interactive `Espresso` calculator is equally valid.

```
images = [initial]
for _ in range(7):
    image = initial.copy()
    image.set_calculator(iEspresso(pw=300, dw=4000,kpts='gamma',xc='PBE'))
    images.append(image)
images.append(final)
```

4.3 Running the NEB calculation and analyzing the results

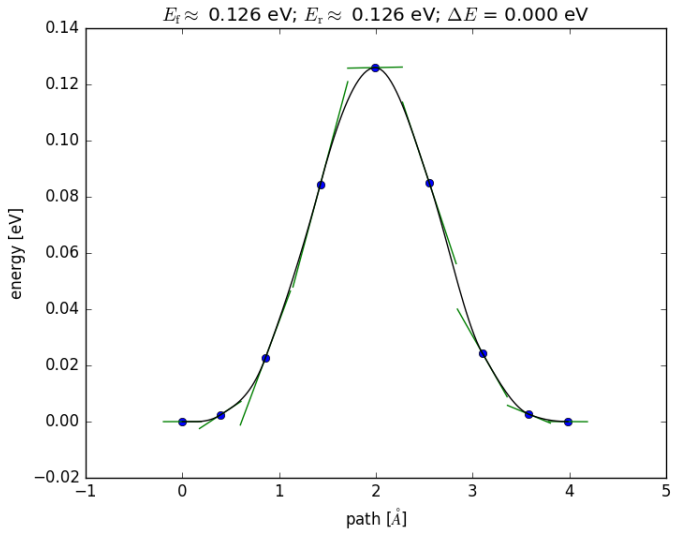
The `NEBEspresso` class is instantiated with the list of images and is used just as the `ase.neb.NEB` super class, with one exception: the `parallel` keyword. `NEBEspresso` ignores `'parallel'=False` and will always attempt to distribute the image calculators over the CPU pool available to the job.

```
from espresso.nebespresso import NEBEspresso
from ase.optimize.fire import FIRE as QuasiNewton

neb = NEBEspresso(images)
neb.interpolate('idpp')

qn = QuasiNewton(neb, logfile='ethane_linear.log', trajectory='neb.traj')
qn.run(fmax=0.05)

from ase.neb import NEBTools
nt = NEBTools(neb.images)
fig = nt.plot_band()
fig.savefig('rotation-barrier.png')
```



CHAPTER 5

API Reference

CHAPTER 6

Indices and tables

- genindex
- modindex
- search