pytint

Sarath Menon

Mar 13, 2022
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Installing LAMMPS</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Installing pytint</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Install dependencies</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Install pytint</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Publications</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Documentation</td>
<td>11</td>
</tr>
<tr>
<td>5.1</td>
<td>Input file</td>
<td>11</td>
</tr>
<tr>
<td>5.2</td>
<td>pytint package</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Indices and tables</td>
<td>15</td>
</tr>
</tbody>
</table>
Python library for free energy calculations using thermodynamic integration
pytint works with the standard version of LAMMPS. Currently supported pair styles are pace (works with lammps-ace) and eam (with standard LAMMPS). If you want use other pair styles such as snap or sw, please contact.

pytint needs LAMMPS compiled as a library with Python support. It can be done by the following instructions:

```
cd lammps
mkdir build_lib
cd build_lib
make 
make # -j${NUM_CPUS}
cp liblammps${SHLIB_EXT}* ../src
cd ../src
make install-python
```

The include files and compiled files should be available in the paths. A full set of instructions can be found here.
2.1 Install dependencies

The following packages need to be installed.

- numpy (conda install -c conda-forge numpy)
- scipy (conda install -c conda-forge scipy)
- pyyaml (conda install -c conda-forge pyyaml)
- mendeleev (conda install -c conda-forge mendeleev)
- pylammpsmpi (conda install -c conda-forge pylammpsmpi)
- pyscal (conda install -c conda-forge pyscal)
After installing the requirements, pytint can be installed by,

```shell
git clone https://git.noc.ruhr-uni-bochum.de/atomicclusterexpansion/pytint.git
cd pytint
python setup.py install

cd pytint/docs
pip install -r requirements.txt
make html
```

The files will be in pytint/docs/build/html.

pytint can be run as both a Python library and as a command line tool. The recommended way to use pytint is through the command line. After installation, pytint can be accessed from the terminal using,

```
tint --help
```

The main option one needs to specify is the `--input` or `-i`. This keyword species the location of the input file. The format of the input file is discussed in detail here.

```
tint -i input.yaml
```

Such a command will read the input file and start NEHI calculations for each temperature mentioned in the input file. Alternatively, one can use the `--mode` option to launch a reversible scaling calculation.

```
tint -i input.yaml -m rs
```

In this case, one NEHI calculation is done for the first temperature mentioned in the input file, and then a reversible scaling calculation is done to extend the free energy up to the last temperature specified in the input file.


5.1 Input file

pytint uses a yaml file for specifying the input options. In this section, the various blocks of the input file is discussed. A complete sample input file is also provided in the Examples section.

5.1.1 main block

main block consists of the major options that the user has to provided. A sample block is shown below.

```yaml
main:
  temperature: [1000, 1400]
  pressure: [0]
  element: 'Cu'
  lattice: [FCC, LQD]
  nsims: 3
```

- **temperature**: List of temperatures at which NEHI calculations have to be done. In case of regular use, one calculation will be started for each temperature. In case of `--mode rs`, one NEHI calculation will be done for the first temperature. After that, reversible scaling calculation will be done to extend the free energy upto the last temperature specified.

- **pressure**: Pressure for the calculation. Currently only zero pressure (Helmholtz free energy) is supported.

- **element**: The chemical symbol of the element used.

- **lattice**: The lattices for which the free energy calculations have to be done. Supported lattices are BCC, FCC, HCP, DIA, SC and LQD.

- **nsims**: The number of independent calculations to be carried out for finding the error in the estimated free energy.
### 5.1.2 md block

```python
md:
    timestep: 0.001
    pair_style: pace
    pair_coeff: "* * Cu.ace Cu"
    mass: 63.546
    tdamp: 0.1
    pdamp: 0.1
    nx: 5
    ny: 5
    nz: 5
    te: 25000
    ts: 50000
```

- **timestep**: Timestep in ps for the md simulations
- **pair_style**: Pair style used in LAMMPS. Supported pair styles are `pace`, `eam` and its variants, `sw` and `snap`.
- **pair_coeff**: Pair coefficient command used in LAMMPS. The relative/full path of the input potential file is also specified here.
- **mass**: Atomic mass of the element.
- **tdamp**: Thermostat damping in units of time.
- **pdamp**: Barostat damping in units of time.
- **nx**: Number of units cells in the 001 direction.
- **ny**: Number of units cells in the 010 direction.
- **nz**: Number of units cells in the 100 direction.
- **te**: Number of time steps for equilibration runs.
- **ts**: Number of time steps for switching runs.

### 5.1.3 queue block

This block specifies the input parameters for submitting the job on a cluster

```python
queue:
    scheduler: slurm
    cores: 40
    jobname: cu
    walltime: "23:50:00"
    queue: shorttime
    memory: 3GB
    modules:
        - anaconda/4
    commands:
        - source .bashrc
        - conda activate py3
        #any other extra options
        #options:
        #  - "-j Y"
```
• scheduler: The scheduler to be used for calculations. Supported options are local, slurm or sge.
• cores: Number of cores to be used for the md runs.
• jobname: Name of the job. Ignored for local.
• walltime: Walltime for the job. Ignored for local.
• queuename: Name of the submission queue. Ignored for local.
• memory: Total memory requested per core. Ignored for local.
• modules: Name of module that need to be loaded.
• commands: Extra commands that will be run in the beginning of the submission script. If a conda environment is used, the activate statements will be here.
• options: Further special options for the submission script.

5.2 pytint package

5.2.1 Submodules

5.2.2 pytint.fitting module

5.2.3 pytint.input module

5.2.4 pytint.integrators module

5.2.5 pytint.kernel module

5.2.6 pytint.lattice module

5.2.7 pytint.liquid module

5.2.8 pytint.queue module

5.2.9 pytint.queuekernel module

5.2.10 pytint.reversiblescaling module

5.2.11 pytint.solid module

5.2.12 pytint.splines module

5.2.13 Module contents
INDICES AND TABLES

• genindex
• modindex
• search