pyhma Documentation

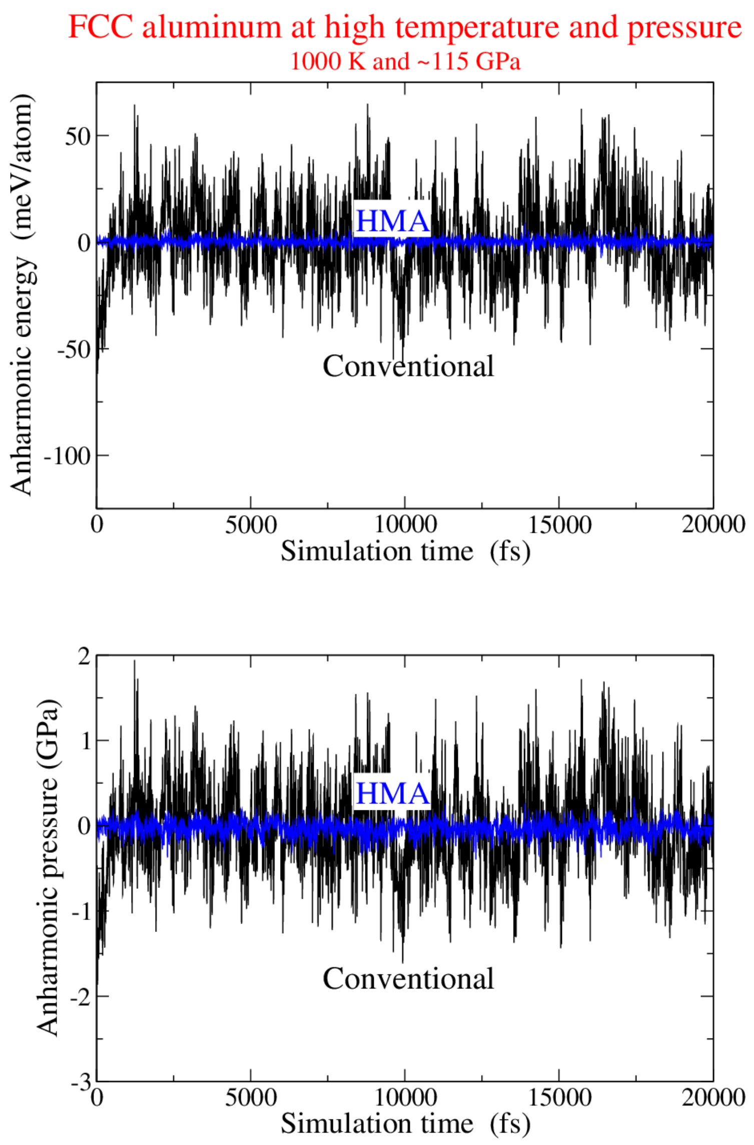
Release 0.0.1

Sabry Moustafa

Oct 08, 2020

Contents

This project is a Python implementation of the Mapped-Averaging method for precise estimation of ensemble averages using molecular simulation.



CHAPTER 1

Theory

1.1 Absolute free energy

The classical Helmholtz configurational free energy A of a system at temperature T and volume V is related to its configurational partition function Q via:

$$A = -k_{\mathrm{B}} T \ln Q \qquad \text{where} \qquad Q = \int_V e^{-\beta U(\mathbf{x})} \mathrm{d}\mathbf{x}$$

with $\mathbf{x}$ representing coordinates of all atoms. For simplicity, from now on we will be using unitless energy $\mathcal{U} \equiv \beta U$ and free energy $\mathcal{A} \equiv \beta A$ (and their derivatives; e.g., force), where $\beta = 1/k_{\mathrm{B}} T$.

1.2 Free energy derivatives

Derivative of free energy w.r.t external perturbation or distortion (e.g., temperature or volume) is related to material properties. For example, average energy U and pressure P are given by

$$U = \partial_\beta \mathcal{A} \qquad \text{and} \qquad P = -k_{\mathrm{B}} T \, \partial_V \mathcal{A}$$

(using ∂_ν to represent the derivative operator $\equiv \frac{\partial}{\partial \nu}$). To get general expression of free energy derivative, we will use the vector λ to represent all perturbations of interest; e.g., $\lambda = (\beta, V)$. The unitless free energy $\mathcal{A}$ at some λ is then given by

$$\mathcal{A}(\lambda) = -\ln Q(\lambda)$$

and its first and second derivatives are given by

$$\partial_\nu \mathcal{A} = -\frac{\partial_\nu Q}{Q} \qquad \text{and} \qquad \partial_{\mu\nu} \mathcal{A} = -\frac{\partial_{\mu\nu} Q}{Q} + \frac{\partial_\nu Q}{Q} \, \frac{\partial_\mu Q}{Q}$$

To get $Q(\lambda)$ derivatives, we need to recognize that, in general, the integration boundary Ω is a function of λ,

$$Q(\lambda) = \int_{\Omega(\lambda)} e^{-\mathcal{U}(\mathbf{y}, \lambda)} \mathrm{d}\mathbf{y}$$

where $\mathbf{y}$ represents the new coordinates that span the general phase space $\Omega\left(\lambda\right)$, at some general λ. However, we can use the change of variables technique to carry out the integration over some λ-independent phase-space boundary (we will use $\Omega(\lambda) = V$) and use the Jacobian determinant J to transform from the $\mathbf{x}$ configurations (which span V) to the mapped coordinates $\mathbf{y}(\mathbf{x}, \lambda)$, using $\mathrm{d}\mathbf{y} = J\mathrm{d}\mathbf{x}$

$$Q\left(\lambda\right) = \int_V e^{-\mathcal{U}(\mathbf{y}(\mathbf{x},\lambda),\lambda)} J\mathrm{d}\mathbf{x}$$

now, Q can be written as a function of the "normal" (current) coordinates $\mathbf{x}$

$$Q\left(\lambda\right) = \int_V e^{-\mathcal{U}'}\mathrm{d}\mathbf{x}$$

where we define a modified potential $\mathcal{U}' \equiv \mathcal{U} - \ln J$, which accounts for the mapping. Now, the Q derivatives can be easily evaluated

$$\partial_\nu Q = - \int_V e^{-\mathcal{U}'}\ \mathrm{D}_\nu\mathcal{U}'\ \ \mathrm{d}\mathbf{x}$$

$$\partial_{\mu\nu} Q = \int_V e^{-\mathcal{U}'}\ \left[\left(\mathrm{D}_\nu\mathcal{U}'\right)\left(\mathrm{D}_\mu\mathcal{U}'\right) - \mathrm{D}_{\mu\nu}\mathcal{U}'\right]\ \mathrm{d}\mathbf{x}$$

where we used the D_ν operator on some function $f(\mathbf{y}(\mathbf{x},\lambda),\lambda)$ to represent the total (Lagrangian or material) derivative (i.e., $\mathrm{D}_\nu f = \partial_\nu f + (\partial_\nu \mathbf{y})\cdot\nabla f$). Accordingly, the derivatives of $\mathcal{A}$ are simply given by

$$\partial_\nu\mathcal{A} = \left\langle \mathrm{D}_\nu\mathcal{U}'\right\rangle \qquad \text{and} \qquad \partial_{\mu\nu}\mathcal{A} = \left\langle \mathrm{D}_{\mu\nu}\mathcal{U}'\right\rangle - \mathrm{Cov}\left(\mathrm{D}_\nu\mathcal{U}'\ ,\ \mathrm{D}_\mu\mathcal{U}'\right)$$

where $\mathrm{Cov}\left(X,Y\right) \equiv \left\langle XY\right\rangle - \left\langle X\right\rangle\left\langle Y\right\rangle$ is the covariance between the stochastic variables X and Y. We are left with evaluating derivatives of $\mathcal{U}'$, which are related to U derivatives via $\mathrm{D}_\nu\mathcal{U}' = \mathrm{D}_\nu\mathcal{U} - \mathrm{D}_\nu J$ and $\mathrm{D}_{\mu\nu}\mathcal{U}' = \mathrm{D}_{\mu\nu}\mathcal{U} - \mathrm{D}_{\mu\nu} J$.

1. *Evaluation of energy derivatives*

First, $\mathcal{U}$ derivatives can be directly evaluated using the relation between the total (Lagrangian) and partial (Eulerian) derivatives:

$$\mathrm{D}_\nu\mathcal{U} = \partial_\nu\mathcal{U} - \mathcal{F}\cdot\dot{\mathbf{x}}^\nu$$

where $\mathcal{F} \equiv -\nabla\mathcal{U} = -\beta\nabla U$ is the force vector on all atoms and $\dot{\mathbf{x}}^\nu \equiv \partial_\nu\mathbf{y}$ represents the mapping "velocity" of the external perturbation ν. Applying this operator twice, we get the second derivative

$$\mathrm{D}_{\mu\nu}\mathcal{U} = \partial_{\mu\nu}\mathcal{U} - \left(\ddot{\mathbf{x}}^{\mu\nu} + \dot{\mathbf{x}}^\mu\cdot\nabla\dot{\mathbf{x}}^\nu\right)\cdot\mathcal{F} + \dot{\mathbf{x}}^\nu\cdot\Phi\cdot\dot{\mathbf{x}}^\mu - \left(\dot{\mathbf{x}}^\nu\cdot\partial_\mu\mathcal{F} + \dot{\mathbf{x}}^\mu\cdot\partial_\nu\mathcal{F}\right)$$

where $\Phi \equiv \nabla\nabla\mathcal{U} = \beta\nabla\nabla\mathcal{U}$ is the force constant matrix and $\ddot{\mathbf{x}}^{\mu\nu} \equiv \partial_\mu\dot{\mathbf{x}}^\nu$ is the $\mu\nu$ "acceleration", or the rate of change of $\dot{\mathbf{x}}^\nu$ w.r.t. μ (note that $\ddot{\mathbf{x}}^{\mu\nu} \neq \ddot{\mathbf{x}}^{\nu\mu}$).

2. *Evaluation of Jacobian derivatives*

Now, in order to get $\mathrm{D}_\nu J$, we need to do two steps. First, perform differentiation of the phase space volume, using (again) the change of variables technique

$$\partial_\nu\Omega(\lambda) = \partial_\nu\int_{\Omega(\lambda)} 1\ \mathrm{d}\mathbf{y} = \partial_\nu\int_V \mathrm{D}_\nu J\ \mathrm{d}\mathbf{x}$$

Second,use the Reynolds transport theorem along with the divergence theorem in our multidimensional space

$$\partial_\nu \int_{\Omega(\lambda)} f\left(\mathbf{y},\lambda\right) \mathrm{d}\mathbf{y} = \int_{\Omega(\lambda)} \left[\partial_\nu f + \nabla \cdot \left(\dot{\mathbf{x}}^\nu f\right)\right] \mathrm{d}\mathbf{y}$$

Applying this theorem to our case of interest (i.e., $f = 1$; hence, $\partial_\nu f = 0$), we get

$$\partial_\nu \int_{\Omega(\lambda)} 1 \ \mathrm{d}\mathbf{y} = \int_{\Omega(\lambda)} \nabla \cdot \left(\dot{\mathbf{x}}^\nu f\right) \mathrm{d}\mathbf{y} = \int_{V} \nabla \cdot \left(\dot{\mathbf{x}}^\nu f\right) J \mathrm{d}\mathbf{x}$$

where we used the change of variables in the last term on the right-hand side. Now, equating both derivatives we directly get an expression for $\mathrm{D}_\nu J$

$$\mathrm{D}_\nu J = J \nabla \cdot \dot{\mathbf{x}}^\nu$$

Repeating the same process with another derivative w.r.t. μ, we directly get

$$\mathrm{D}_{\mu\nu} J = J \left[\nabla \cdot \left(\partial_\mu \dot{\mathbf{x}}^\nu\right) + \dot{\mathbf{x}}^\mu \cdot \nabla \left(\nabla \cdot \dot{\mathbf{x}}^\nu\right)\right]$$

Since we are interested in evaluating the derivatives at $\mathbf{y} = \mathbf{x}$, then $J = 1$; hence $\mathrm{D}_\nu J = \nabla \cdot \dot{\mathbf{x}}^\nu$ and $\mathrm{D}_{\mu\nu} J = \nabla \cdot \left(\partial_\mu \dot{\mathbf{x}}^\nu\right) + \dot{\mathbf{x}}^\mu \cdot \nabla \left(\nabla \cdot \dot{\mathbf{x}}^\nu\right)$.

1.3 Mapping velocity

Since Q is a function only of λ, **average** free energy derivatives do not depend on how $\mathbf{x}$ get mapped into the $\mathbf{y}$ coordinates; or, in other words, they do not depend on the mapping velocity $\dot{\mathbf{x}}^\nu$. However, the **fluctuations** (or uncertainty) in these averages do depend on the mapping. Therefore, for the purposes of molecular simulation measurements we need to choose $\dot{\mathbf{x}}^\nu$ that reduces the stochastic uncertainty as much as possible.

To develop such a mapping we need to recognize that free energy derivatives are given as ensemble averages over $\mathrm{D}_\nu \mathcal{U}'$ (and its derivative, $\mathrm{D}_{\mu\nu}\mathcal{U}'$). Therefore, a perfect mapping is such that $\mathrm{D}_\nu \mathcal{U}'$ is independent on coordinates $\mathbf{x}$; hence

$$\partial_\nu \mathcal{A} = \ \langle \mathrm{D}_\nu \mathcal{U}' \rangle = \mathrm{D}_\nu \mathcal{U}'$$

Using the above energy and Jacobian derivatives, we get

$$\partial_\nu \mathcal{A} = \partial_\nu \mathcal{U} - \nabla \cdot \dot{\mathbf{x}}^\nu - \mathcal{F} \cdot \dot{\mathbf{x}}^\nu$$

Solving this equation yields the unique mapping that yields no fluctuations; however, there are two problems. First of all, $\partial_\nu \mathcal{A}$ is the very quantity that we need to measure. Second, since $\dot{\mathbf{x}}^\nu$ is a multidimensional vector ($3N$ for the case of atomic systems) we have under-determined system as we only have one equation to solve.

The first problem is solved using the fact that $\dot{\mathbf{x}}^\nu$ does not affect average estimates; hence, it can be derived from another (known) system that approximates $\mathcal{A}$, which we will call reference.

$$\partial_\nu \mathcal{A}^{\mathrm{ref}} = \partial_\nu \mathcal{U}^{\mathrm{ref}} - \nabla \cdot \dot{\mathbf{x}}^\nu - \mathcal{F}^{\mathrm{ref}} \cdot \dot{\mathbf{x}}^\nu$$

where $\partial_\nu \mathcal{A}^{\mathrm{ref}}$ is a reference-dependent constant (function only of λ), named c. Because the reference only approximates $\mathcal{A}$, the $\dot{\mathbf{x}}^\nu$ obtained from this formula will not yield a zero-fluctuation average; however, if the reference is a good approximation, we can expect substantially smaller fluctuations in the average.

To address the second problem, we will assert that each degree of freedom (dof) is mapped with the same amount (scaling); so

$$\partial_\nu \sqcap^{\mathrm{ref}} - \partial_x \dot{x}^\nu - \{^{\mathrm{ref}} \dot{x}^\nu = \partial_\nu \dashv^{\mathrm{ref}} \equiv c(\lambda)$$

where small symbols represent an intensive quantities (e.g., $u \equiv U/\mathrm{dof}$), and x is one of the coordinates of $\mathbf{x}$. For a given λ, this is a standard first-order differential equation, with the unknown being the velocity of mapping $\dot{x}(x, \lambda)$. For simplicity, we will drop the λ dependency from all terms, hence

$$\partial_x \dot{x}^{\nu}\left(x\right) + \{\left(x\right)^{\mathrm{ref}} \dot{x}^{\nu}\left(x\right) = \partial_{\nu} \sqcap \left(x\right)^{\mathrm{ref}} - c \equiv g\left(x\right)$$

where $g(x)$ is a known function once a reference system is chosen. The solution of this equation is given by

$$\dot{x}^{\nu} = e^{-I(x)} \left(\int g \; e^{I(x)} \mathrm{d}x + \mathrm{constant} \right)$$

where $I(x) \equiv \int f(x)^{\mathrm{ref}} \mathrm{d}x$. The integration constant can be evaluated by requiring the mapping to have some value at some coordinate x.

CHAPTER 2

Application to Crystals

For crystalline systems, a reference for mapping can be chosen to be non-interacting harmonic system, or Einstein crystal (EC). Using the offset from the lattice sites (i.e., $r - r_{\mathrm{lattice}}$) to represent our coordinate x, the potential energy for each dof is then given by

$$u^{\mathrm{ref}} = \alpha(\lambda)x^2$$

hence, the free energy is given by

$$a^{\mathrm{ref}} = \frac{1}{2}\ln\left(\alpha(\lambda)/\pi\right)$$

Accordingly, $f^{\mathrm{ref}} = -2\alpha(\lambda)x$, $I = -\alpha(\lambda)x^2$, and $g = (\partial_\nu\alpha)\left[x^2 - 1/2\alpha\right]$. Requiring the mapping velocity to vanish when atoms are at their lattice sites; i.e., $\dot{x}^\nu = 0$ at $x = 0$, the solution of the velocity mapping equation is reduced to

$$\dot{x}^\nu =$$

$$e^{-I(x)}\int_0^x g\ e^{I(x)}\mathrm{d}x$$

$$=$$

$$e^{\alpha(\lambda)x^2}\int_0^x g\ e^{-\alpha(\lambda)x^2}\mathrm{d}x$$

$$=$$

$$-\frac{\partial_\nu\alpha}{2\alpha}\ x$$

We will now consider two cases: temperature and volume free energy derivative; or energy and pressure, consequently. Note that all energy quantities (and its derivatives, like forces) are multiplied by β; hence, the force constant $\alpha \to \beta\alpha$, where α is now temperature independent.

- **case 1:** $\nu = \beta$

$$\partial_\beta(\beta\alpha) = \alpha$$

hence,

$$\dot{x}^\beta = -\frac{1}{2\beta}\ x$$

- **case 2:** $\nu = V$

Here, we need to use the L-scaled coordinates (i.e., $x \to x/L$); hence $\alpha \to L^2\beta\alpha = V^{2/3}\beta\alpha(V)$

$$\dot{x}^V = -\frac{1}{2}\frac{\partial_\nu(V^{2/3}\alpha(V))}{V^{2/3}\alpha(V)}\, x = \left(-\frac{1}{2}\frac{\partial_V\alpha(V)}{\alpha(V)} - \frac{1}{3V}\right) x = \left(\beta\, p^{\text{harm}} - \frac{1}{3V}\right) x$$

Where p^{harm} is the harmonic pressure per each dof. For more details you can refer to our PRE and JCTC work.

2.1 Anharmonic energy

Conventional (no mapping):

$$U^{\text{ah}} = \langle U \rangle - \frac{d(N-1)}{2} k_{\text{B}}T - U^{\text{lat}}$$

Mapped averaging (Einstein crystal reference):

$$U^{\text{ah}} = \left\langle U + \frac{1}{2}\mathbf{F}\cdot\Delta\mathbf{r} \right\rangle - U^{\text{lat}}$$

2.2 Anharmonic pressure

Conventional (uniform scaling):

$$P^{\text{ah}} = \left\langle P^{\text{vir}} \right\rangle + \rho k_{\text{B}}T - P^{\text{qh}} - P^{\text{lat}}$$

Mapped averaging (Einstein crystal reference):

$$P^{\text{ah}} = \left\langle P^{\text{vir}} + c\,\mathbf{F}\cdot\Delta\mathbf{r} \right\rangle - P^{\text{lat}}$$

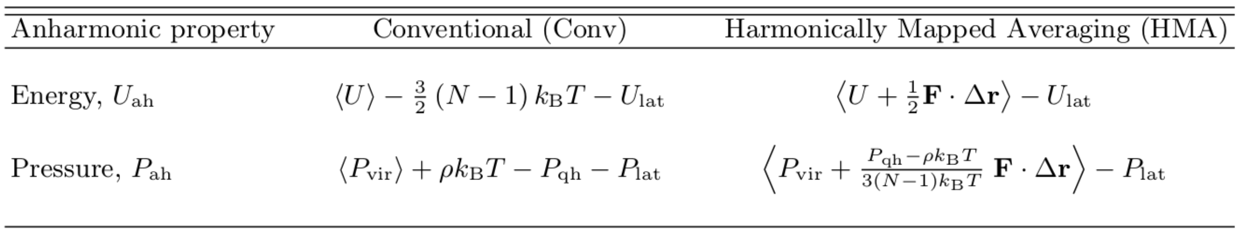
where c is a constant and given by, $c = \frac{\beta P^{\text{qh}} - \rho}{d(N-1)}$

The formulas for both anharmonic energy and pressure are summarized in Figure 2.1.

2.3 Equivalence of Conv and HMA

The equivalence between both conventional and mapped-averaging expressions can be easily seen by recognizing this equality for crystalline systems:

$$\langle \mathbf{F}\cdot\Delta\mathbf{r} \rangle = -d\,(N-1)\,k_{\text{B}}T$$



Anharmonic property	Conventional (Conv)	Harmonically Mapped Averaging (HMA)
Energy, U_{ah}	$\langle U\rangle - \frac{3}{2}\left(N-1\right)k_{\mathrm{B}}T - U_{\mathrm{lat}}$	$\left\langle U + \frac{1}{2}\mathbf{F}\cdot\Delta\mathbf{r}\right\rangle - U_{\mathrm{lat}}$
Pressure, P_{ah}	$\langle P_{\mathrm{vir}}\rangle + \rho k_{\mathrm{B}}T - P_{\mathrm{qh}} - P_{\mathrm{lat}}$	$\left\langle P_{\mathrm{vir}} + \frac{P_{\mathrm{qh}}-\rho k_{\mathrm{B}}T}{3(N-1)k_{\mathrm{B}}T}\,\mathbf{F}\cdot\Delta\mathbf{r}\right\rangle - P_{\mathrm{lat}}$

Fig. 2.1: Conventional and HMA formulas for anharmonic energy and pressure.

Plugging this expression into the HMA expressions yields the conventional average expression.

Proof:

The general expression for the configurational partition function is given by:

$$Q = \int e^{-\beta U}\mathrm{d}\mathbf{x}$$

For crystalline systems, we use $\Delta\mathbf{x} \equiv \mathbf{x} - \mathbf{x}^{\mathrm{lat}}$

$$Q = \int_{\mathrm{WS}} e^{-\beta U}\,\mathrm{d}^{dN}\Delta x$$

Where the integration is carried out withing the Wigner-Seitz (WS) volume of each atom. This can be written as

$$Q = \int_{\mathrm{WS}} \mathrm{d}^{dN-1}\Delta x \int_{\mathrm{WS}} e^{-\beta U}\,\mathrm{d}\Delta x_1$$

Using integration by parts:

$$Q = \int_{\mathrm{WS}} \left[\Delta x_1 e^{-\beta U}\right]_{\Delta x_1^{\mathrm{min}}}^{\Delta x_1^{\mathrm{max}}} \mathrm{d}^{dN-1}\Delta x \;\; - \beta \int_{\mathrm{WS}} F_1 \Delta x_1\, e^{-\beta U}\mathrm{d}^{dN}\Delta x$$

The surface (first) term on the right-hand side vanishes due to large values of U at the surface of the WS volume. Dividing by Q, we finally get:

$$\langle F_1 \Delta x_1\rangle = -k_{\mathrm{B}}T$$

For $d(N-1)$ degrees-of-freedom, we get: $\langle \mathbf{F}\cdot\Delta\mathbf{r}\rangle = -d\left(N-1\right)k_{\mathrm{B}}T$

CHAPTER 3

pyHMA

`pyHMA` is a VASP post-processor (written in Python 3) for precise measurment of crystalline *anharmonic* properties using Harmonically Mapped Averaging (HMA) method. It is based on post-processing `vasprun.xml` output file(s) obtained from NVT Born-Oppenheimer *ab initio* molecular dynamics (AIMD) simulation. See Table 2.1 as an example for HMA expressions for anharmonic energy and pressure, along with direct/conventional (Conv) counterpart.

`pyHMA` is free software: you can modify and/or redistribute it under the terms of the Mozilla Public License (MPL 2.0).

Please cite this paper when using pyHMA package in your research:

> Sabry G. Moustafa, Apoorva Purohit, Andrew J. Schultz, and David A. Kofke, pyHMA: A VASP Post-processor for Precise Measurement of Crystalline Anharmonic Properties using Harmonically Mapped Averaging, Comput. Phys. Commun., 2020.

Note:

- The term *anharmonicity* is commonly used in literature to qualitatively describe a system with no equilibrium configuration at 0 K (i.e., imaginary frequencies); in other words, it refers to a "non-harmonic" potential energy surface.
- Here, however, we define *anharmonic contribution* of some property X as the residual in excess of the harmonic approximation; i.e., $X_{\mathrm{ah}} \equiv X - (X_{\mathrm{lat}} + X_{\mathrm{qh}})$. Therefore, this specific definition is meaningless if the system does not have equilibrium lattice configuration at $T = 0$ K. For this reason, `pyHMA` checks forces on the first configuration to make sure the system has an equilibrium configuration (i.e., zero forces).

3.1 Installation

Attention: pyHMA is written in Python 3 syntax; hence, only compatible `pip` installer can be used.

From PyPI (Recommended)

pyHMA can be directtly installed from Python Package Index (PyPI) using pip command:

```
$ pip install pyhma
```

From Github (development version)

First, clone the source code from the Github repository:

```
$ git clone https://github.com/etomica/mapped-averaging.git
```

and then, go to the mapped-averaging/pyhma directory to install pyHMA locally using pip command:

```
$ pip install --user -e .
```

3.2 Usage

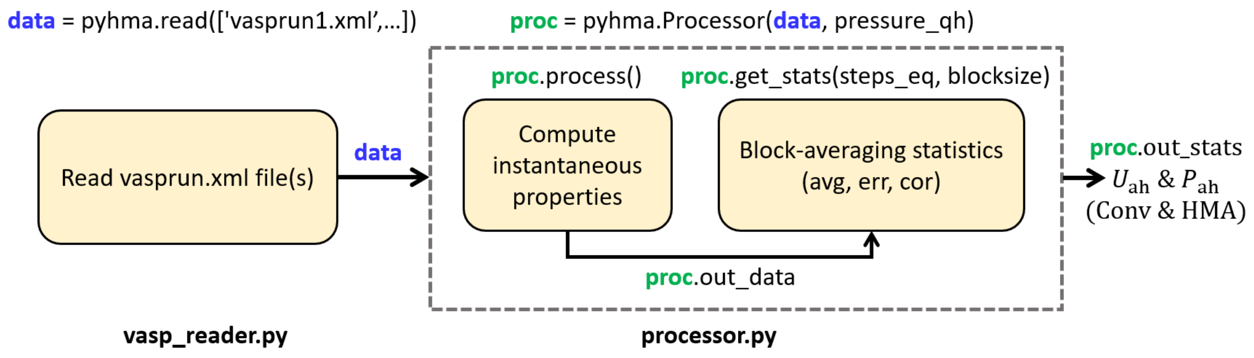

data = pyhma.read(['vasprun1.xml',...])
proc = pyhma.Processor(data, pressure_qh)
proc.process()
proc.get_stats(steps_eq, blocksize)
Read vasprun.xml file(s)
data
Compute instantaneous properties
Block-averaging statistics (avg, err, cor)
proc.out_stats
U_{ah} & P_{ah} (Conv & HMA)
proc.out_data
vasp_reader.py
processor.py

Fig. 3.1: Overall structure of pyHMA package.

As shown in the diagram, pyHMA post-processes VASP AIMD data in two stages: first, reading vasprun.xml output file(s) (pyhma.vasp_reader module), and then process the data to compute anharmonic properties, using conventional (Conv) and harmonically-mapped averaging (HMA) approaches (pyhma.processor module). Below is a detailed description of each stage along with an example of fcc aluminum at high pressure ($P_{\rm lat} = 114.4$ GPa; corresponds to $V = 10A^3$/atom) and temperature (1000 K).

Attention:

1. It is worth emphasizing here that $U_{\rm lat}$, $P_{\rm lat}$, and $P_{\rm qh}$ inputs needed for the anharmonic calculations (see Table 2.1) must be obtained using the same setting with with AIMD (e.g., system size, DFT parameters, etc.) in order to ensure having the same potential-energy surface.
2. However, lattice and quasiharmonic contributions needed for computing the full property (lat+qh+ah) should be obtained from separate calculations as those components can converge at a different rates from the anharmonic one. The full property X can be then decomposed to $X = X^*_{\rm lat} + X^*_{\rm qh} + X_{\rm ah}$, where the asterisk refers to using different DFT setting.
3. To summarize points 1 and 2 above: $X_{\rm lat}$ and $X_{\rm qh}$ used as inputs to pyHMA should not be used as the lat and qh contributions of the full property. On the other hand, $X^*_{\rm lat}$ and $X^*_{\rm qh}$ used for computing the full property should not be used as the lat and qh input parameters for pyHMA. Mixing between these two uses will result in inconsistent and inaccurate results.

3.2.1 1. Interactive usage

Reading

In this stage, `pyhma.vasp_reader.read()` function parses `vasprun.xml` AIMD simulation file and returns a `data` dictionary with simulation information to be processed in the second stage. In the below example, the AIMD simulation consists of two consecutive runs of the same simulation, where the initial configuration of the first run is the fcc lattice positions.

```
>>> import pyhma
>>> data = pyhma.read(['vasprun-1.xml', 'vasprun-2.xml'])
```

The `data` dictionary contains the following keys:

- **box_row_vecs** (Å): box edge row vectors
- **num_atoms**: total number of atoms
- **volume_atom** (Å^3/atom): average volume per atom
- **basis**: list of atomic fractional positions of first configuration
- **position**: instantaneous atomic fractional positions
- **force** (eV/Å): instantaneous atomic forces
- **energy** (eV/atom): instantaneous potential energy (E0)
- **pressure** (GPa): instantaneous pressure
- **pressure_ig** (GPa): ideal gas pressure
- **timestep** (fs): MD timestep
- **temperature** (K): NVT set temperature

This function also takes optional arguments: *raw_files*, *force_tol*, and *verbose*. By setting `raw_files=True` (default is `False`), the following `.dat` files will be generated, which contain raw data from `vasprun.xml` file(s). These files are only for diagnostics purposes, and not used in the process stage.

- **poscar_eq.dat**: initial (must be the equilibrium) POSCAR file (in fractional coordinates)
- **energy.dat**: instantaneous potential energy, E0 (in eV/atom)
- **pressure.dat**: instantaneous pressure (in GPa)
- **posfor.dat**: instantaneous atomic positions and forces (in Å and eV/Å)

The second argument (`force_tol`) is the maximum force allowed on any atom in the initial configuration (default is 0.001 eV/Å). This is used to make sure the initial configuration is the minimized (or, equilibrium). If this condition is not matched, the function will be interrupted and prints a warning with a list of atoms having large initial forces.

To print the lattice vectors, and the initial atomic positions and forces, you can add `verbose=True` (default is `False`) to the arguments.

Below is the output resulted from using all optional arguments:

```
>>> data = pyhma.read(['vasprun-1.xml', 'vasprun-2.xml'], raw_files=True, force_tol=0.
↪002, verbose=True)

Reading vasprun-1.xml vasprun-2.xml
==============================================
 first configuration data from vasprun-1.xml
 ----------------------------------------------
```

(continues on next page)

(continued from previous page)

```
 32 atoms (total)
 Box edge (row) vectors
  6.83990379   0.00000000   0.00000000
  0.00000000   6.83990379   0.00000000
  0.00000000   0.00000000   6.83990379

 atom       xyz (direct) coordinates (A)                  xyz forces (eV/A)
    1    0.00000000   0.00000000   0.00000000      0.00000098   0.00000047  -0.00000207
    2    0.50000000   0.00000000   0.00000000     -0.00000114  -0.00000068  -0.00000010
    3    0.00000000   0.50000000   0.00000000      0.00000092  -0.00000001   0.00000009
    4    0.50000000   0.50000000   0.00000000     -0.00000075   0.00000133   0.00000002
    5    0.00000000   0.00000000   0.50000000      0.00000156  -0.00000124   0.00000148
    6    0.50000000   0.00000000   0.50000000     -0.00000228  -0.00000113   0.00000060
    7    0.00000000   0.50000000   0.50000000      0.00000058   0.00000088  -0.00000025
    8    0.50000000   0.50000000   0.50000000     -0.00000022   0.00000147   0.00000020
    9    0.25000000   0.25000000   0.00000000      0.00000626   0.00000445   0.00000093
   10    0.75000000   0.25000000   0.00000000     -0.00000698   0.00000368  -0.00000093
   11    0.25000000   0.75000000   0.00000000      0.00000876  -0.00000401  -0.00000012
   12    0.75000000   0.75000000   0.00000000     -0.00000828  -0.00000414  -0.00000230
   13    0.25000000   0.25000000   0.50000000      0.00000749   0.00000298  -0.00000160
   14    0.75000000   0.25000000   0.50000000     -0.00000890   0.00000465   0.00000053
   15    0.25000000   0.75000000   0.50000000      0.00000879  -0.00000420   0.00000048
   16    0.75000000   0.75000000   0.50000000     -0.00000810  -0.00000479   0.00000260
   17    0.00000000   0.25000000   0.25000000      0.00000300   0.00000494   0.00000561
   18    0.50000000   0.25000000   0.25000000     -0.00000351   0.00000432   0.00000600
   19    0.00000000   0.75000000   0.25000000      0.00000149  -0.00000550   0.00000703
   20    0.50000000   0.75000000   0.25000000     -0.00000171  -0.00000398   0.00000669
   21    0.00000000   0.25000000   0.75000000      0.00000072   0.00000397  -0.00000567
   22    0.50000000   0.25000000   0.75000000     -0.00000041   0.00000300  -0.00000674
   23    0.00000000   0.75000000   0.75000000     -0.00000135  -0.00000323  -0.00000732
   24    0.50000000   0.75000000   0.75000000      0.00000067  -0.00000412  -0.00000650
   25    0.25000000   0.00000000   0.25000000      0.00000892   0.00000095   0.00000758
   26    0.75000000   0.00000000   0.25000000     -0.00000866  -0.00000109   0.00000571
   27    0.25000000   0.50000000   0.25000000      0.00000780  -0.00000037   0.00000582
   28    0.75000000   0.50000000   0.25000000     -0.00000778   0.00000087   0.00000677
   29    0.25000000   0.00000000   0.75000000      0.00000777  -0.00000093  -0.00000665
   30    0.75000000   0.00000000   0.75000000     -0.00000764  -0.00000253  -0.00000547
   31    0.25000000   0.50000000   0.75000000      0.00000795   0.00000070  -0.00000629
   32    0.75000000   0.50000000   0.75000000     -0.00000705   0.00000248  -0.00000653

  Reading vasprun-1.xml  ( 1 out of 2 )
  Reading vasprun-2.xml  ( 2 out of 2 )
>>>
```

Note:

- The read() function can handle incomplete vasprun.xml file(s) generated from interrupted AIMD runs (by the user, or due to some time constraint). The was possible with using the recovery option of LXML parser.
- If your MD simulation starts from a thermalized/equilibrated (not lattice) configuration, you can just run a single-point energy calculation on the lattice configuration (using the same DFT parameters used with AIMD) and use the output as your `vasprun-1.xml` input to pyHMA, followed by your thermalized `vasprun.xml` files.

Processing

In this stage, the `data` from previous step are processed to compute anharmonic properties. This is done, first, by creating a processor instance (`proc`) of the `pyhma.processor.Processor()` class, using the `data` dictionary and the quasiharmonic pressure (GPa) at the given V and T. The class takes one optional argument, `meV`, to specify whether to report the energy results in meV (`meV=True`) or eV (`meV=False`, default). At this point, the `proc` object carries the same information exist in the `data` dictionary.

```
>>> proc = pyhma.Processor(data, pressure_qh=4.94154, meV=True)
```

Then, the instantaneous properties are obtained by calling the `pyhma.processor.Processor.process()` method, which takes two optional arguments: `steps_tot` and `verbose`. The `steps_tot` is the total number of MD steps to be used for ensemble averages (default is all steps found in `vasprun.xml`) and the `verbose` (default is False) directs `pyHMA` to print information while running. The output is saved to a 2D array (`proc.out_data` attribute) of length equal to all MD steps (or, `steps_tot` if set) and contains four columns: Conv and HMA anharmonic energies and pressures.

```
>>> proc.process(steps_tot=10000, verbose=True)

   Simulation data
   ===============
    Set temperature       (K): 1000.00000
    Volume          (A^3/atom):   10.00000
    MD timestep          (fs):    2.00000
    Lattice energy  (eV/atom):   -2.21324
    Harmonic energy (eV/atom):    0.12522
    Lattice pressure     (GPa):  114.44281
    Harmonic pressure    (GPa):    4.94525

    Found 11036  total MD steps
    Using 10000  user-set MD steps

    Computing instantaneous properties ...
```

The method also generates `energy_ah.out` and `pressure_ah.out` output files for the instantaneous anharmonic energy (eV/atom; or meV/atom if meV=True) and pressure (GPa), respectively. Each file contains three columns; time (in fs), Conv, and HMA estimates of the property. This data is plotted below.

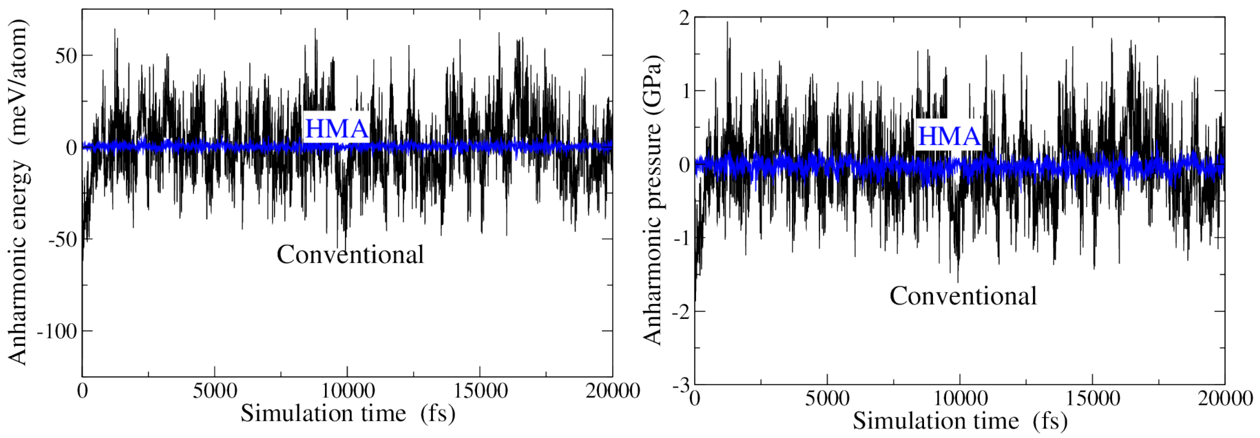


Fig. 3.2: Time vartaion of the anharmonic energy (`energy_ah.out`) and pressure (`pressure_ah.out`).

Lastly, ensemble statistics (average, uncertainty, and block correlation) are obtained using block averaging technique. This is done by invoking the `pyhma.processor.Processor.get_stats()` method, which takes two required arguments (`steps_eq` and `blocksize`) and one optional argument (`verbose`). The `steps_eq` is the number of MD steps used for equilibaration and `blocksize` is the number of MD steps in each block used for block averaging;

so, `steps_tot/blocksize` is the number of blocks to be used. If True, the verbose flag will direct `pyHMA` to print samples information.

The method returns the statistics output in a form of a dictionary (`stats`) of four entries: Conv and HMA anharmonic energies (`e_ah_conv` and `e_ah_hma`) and pressures (`p_ah_conv` and `p_ah_hma`), each with three elements of average (avg), uncertainty (err), and adjacent blocks correlation (cor). The output can be presented in a more user-friendly format by using `pyhma.processor.Processor.print_stats()` method, which yields the output shown below.

```
>>> stats = proc.get_stats(steps_eq=1000, blocksize=90, verbose=True)

Block averaging statistics
==========================
 9000 production steps (after 1000 equilibration steps)
 100 blocks (blocksize = 90  steps)

 Computing statistics ...

>>> proc.print_stats(stats)

 e_ah_conv  (meV/atom):    2.10911 +/- 1.1e+00    cor: 0.35
 e_ah_hma   (meV/atom):    0.42650 +/- 4.3e-02    cor: 0.11
 p_ah_conv       (GPa):    0.01371 +/- 3.1e-02    cor: 0.36
 p_ah_hma        (GPa):   -0.03419 +/- 4.1e-03    cor: 0.26
```

Note:

- The correlation should be as small as possible (less than $\lessapprox 0.2$) to ensure accurate estimate of uncertainty. Although increasing the `blocksize` reduces the correlations, the number of blocks should be large enough ($\gtrapprox 50$) to yield meaningful statistics.
- The Conv and HMA should be statistically consistent, as long as the results are converged with respect to timestep. However, the above example has inconsistent results due to using relatively large timestep ($\Delta t = 2$ fs), though the HMA estimate is still accurate as it converges faster than Conv (see our JCP2018 work for details).

3.2.2 2. pyhma script

Anharmonic properties can be computed in one step from the command-line using `pyhma` script, which uses the same arguments as those used above, except for the use of `r` and `v` short forms of `raw_files` and `verbose` options, respectively. The usage of `pyhma` is given here, where the square brackets represent optional keys:

```
$ # Usage:
$ # pyhma --pressure_qh=qh pressure (GPa) --steps_eq=equilib. steps --blocksize=block␣
↪size
$ #       [--steps_tot=used steps] [--force_tol=force tolerance] [--raw_files|-r] [--
↪meV]
$ #       [--verbose|-v] vasprun-1.xml vasprun-2.xml ...
```

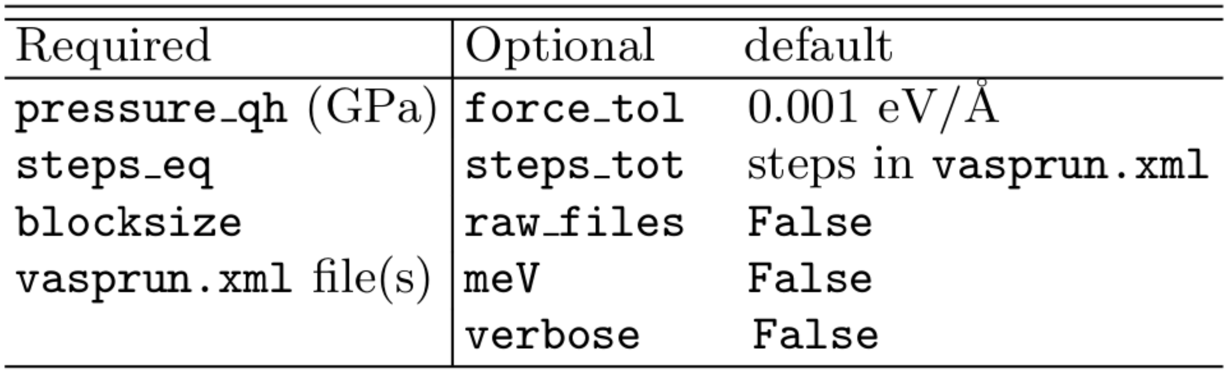
Using the `pyhma` script (with default option) to compute anharmonic energy and pressure of the above fcc aluminum example yields:

```
$ pyhma --pressure_qh=4.94525 --steps_eq=1000 --steps_tot=10000 --blocksize=90  -r --
↪meV
       vasprun-1.xml  vasprun-2.xml

  e_ah_conv (meV/atom):    2.10911 +/- 1.1e+00    cor: 0.35
  e_ah_hma  (meV/atom):    0.42650 +/- 4.3e-02    cor: 0.11
  p_ah_conv       (GPa):    0.01371 +/- 3.1e-02    cor: 0.36
  p_ah_hma        (GPa):   -0.03419 +/- 4.1e-03    cor: 0.26
```

3.2.3 3. Parameters table

The Table below gives a summary of both required and optional arguments used by `pyHMA`.



Required	Optional	default
pressure_qh (GPa)	force_tol	0.001 eV/Å
steps_eq	steps_tot	steps in vasprun.xml
blocksize	raw_files	False
vasprun.xml file(s)	meV	False
	verbose	False

3.3 Modules

3.3.1 pyhma.vasp_reader

3.3.2 pyhma.processor

CHAPTER 4

Indices and tables

- genindex
- modindex
- search