1 Energy-volume curves and EOS

Just like the equation of state for ideal gases is $pV = nRT$, there have also been several attempts to describe bulks with an EOS. Some examples can be found here:
https://en.wikipedia.org/wiki/Equation_of_state

ASE provides a simple interface for fitting such an EOS to calculated energy-volume data points:
https://wiki.fysik.dtu.dk/ase/ase/eos.html

In the following, we will find the equilibrium lattice constant for your system by performing a ground state calculation (= find total energy) for a set of different lattice constants that are slightly smaller or larger compared to the one listed in your CIF file.

1. Calculate the total energies while varying the lattice constant, i. e. while expanding and compressing the cell, within a range of -5% to +5% and 11 total configurations.
   Solution:

```python
import numpy as np
from ase.build import bulk
from ase.io import Trajectory
from gpaw import GPAW, PW

# create a list of lattice parameters to test
a0 = 5.43
percent = 0.05
# syntax: np.linspace(start, end, number of values)
a_list = a0 * (1 + np.linspace(-percent, +percent, 11))
print("Lattice constants:\n", a_list.round(3), "\n\n")

# prepare arrays for total energies and corresponding volumes
etots = []
vols = []
# prepare trajectory object to store all cells in
traj = Trajectory("Si.traj", "w")

# print nice header
print("a       volume   total energy")
print("-----------------------------")
# create the cell, then find, store and print the total energy
for a in a_list:
    si = bulk("Si", "diamond", a)
    si.calc = GPAW(mode=PW(400), xc="PBE", kpts=(8, 8, 8), txt=None)
    # we need volume and energy for E(V) curve;
    # use corresponding getter functions and append values to lists
    vol = si.get_volume()
    vols.append(vol)
    etot = si.get_potential_energy()
    etots.append(etot)
    # print progress; {:1.3f} -> float with 3 digits after dec. point
    print("{:1.3f}   {:2.3f}   {:2.6f}".format(a, vol, etot))
    # write the entire configuration (cell + energy) as trajectory
    traj.write(si)
```

2. Calculate the DFT equilibrium lattice constant from the minimum volume V_0, while keeping in mind that we used the primitive cell for the DFT calculation whereas the lattice constant is always given with respect to the conventional cell! Use the `ase.eos` module to retrieve the minimal volume from the calculated data points.
Solution:

```
from ase.eos import EquationOfState
from ase.io.trajectory import Trajectory
# OPTIONAL: read in the E-V data if not present anymore
traj = Trajectory("Si.traj", "r")
etots = [t.get_potential_energy() for t in traj]
vols = [t.get_volume() for t in traj]
# save volumes and correspoding energies in eos class
eos = EquationOfState(vols, etots)
# fit an EOS and return the minimal volume, energy and bulk modulus
v0, e0, B = eos.fit()
# calculate the DFT lattice constant from the minimal volume
aDFT = (v0 * 4) ** (1 / 3.0)
print("aDFT = {:.3f}".format(aDFT))  # 5.476 (GGA), 5.407 (LDA)
```

3. Verify, that you can obtain the exact same result using the convenience function `ase.eos.calculate_eos(atoms, npoints=5, eps=0.04, trajectory=None, callback=None)`, where the entire code from task 1 is already implemented.
Solution:

```
from ase.build import bulk
from ase.eos import calculate_eos
from gpaw import GPAW, PW
si = bulk("Si", "diamond", 5.43)
si.calc = GPAW(mode=PW(400), kpts=(8, 8, 8), xc='PBE', txt=None)
eos = calculate_eos(si)
v0, e0, B = eos.fit()
aDFT = (v0 * 4) ** (1 / 3.0)
print("aDFT = {:.3f}".format(aDFT))  # 5.476 (GGA), 5.407 (LDA)
```

4. Compare the DFT lattice constant with the one from the CIF file. How large is the difference? Interpret the results.
Solution:
Difference for the Si example is about 1%. However, DFT lattice constants can differ from experimental ones up to 10%!

5. Repeat your script but this time use the LDA approximation instead of the GGA functional (set `xc='LDA'` instead of `xc='PBE'`). What can you tell about the deviation of the lattice constant in this case?
Solution:

   - GGA is underbinding
     smaller binding energies $\longrightarrow$ larger interatomic distances $\longrightarrow$ larger a_0
   - LDA is overbinding
     larger binding energies $\longrightarrow$ smaller interatomic distances $\longrightarrow$ smaller a_0

6. Repeat the analysis graphically: `$ ase gui Si.traj` (Tools $\longrightarrow$ Bulkmodulus)
Solution: $V_0 = 41.047\,\text{Å}$, $B = 88.382\,\text{GPa}$, $E_0 = -10.799\,\text{eV}$ (see Fig. 1)

More information on the `ase.eos` module:
https://wiki.fysik.dtu.dk/ase/tutorials/eos/eos.html
https://wiki.fysik.dtu.dk/ase/ase/eos.html
https://wiki.fysik.dtu.dk/gpaw/tutorialsexercises/structureoptimization/aluminium/aluminium.html
https://wiki.fysik.dtu.dk/gpaw/tutorialsexercises/structureoptimization/lattice_constants/lattice_constants.html

2 Relax calculation via BFGS algorithm

An often performed type of calculation is the so called "relaxation". There, either the atomic positions or the cell parameters or even both simultaneously are optimized with respect to the forces acting on the atoms. For solids/bulks, periodic boundary conditions are usually assumed.

Since the approach of the previous tasks would be rather tedious, there is a simpler and quicker way to search for minima, namely using an optimizer algorithm like BFGS search:
https://en.wikipedia.org/wiki/Broyden-Fletcher-Goldfarb-Shanno_algorithm

1. Perform a BFGS search for the equilibrium lattice constant with GPAW. Set the initial lattice constant to 6.0 Å. Use the `UnitCellFilter` function to ensure simultaneous optimization of cell parameters and atomic positions.

   The algorithm will stop when the force on all individual atoms a is less than a given convergence parameter $f_{\max}$:

   $$\max_a |\boldsymbol{F}_a| < f_{\max}$$

   You can use the following script for the relaxation calculation:

```python
# import necessary modules
from ase.build import bulk
from ase.optimize.bfgs import BFGS
from ase.optimize.bfgslinesearch import BFGSLineSearch
from ase.filters import UnitCellFilter
from gpaw import GPAW
from gpaw import PW

# create cell using non-equilibrium lattice constant
si = bulk("Si", "diamond", 6.0)
si.calc = GPAW(xc="PBE", mode=PW(400), kpts=(8, 8, 8), txt=None)

# make sure atom positions and cell parameters will both be optimized
uf = UnitCellFilter(si)
# also try to replace BFGS by BFGSLineSearch
relax = BFGS(uf, trajectory="Si-BFGS.traj")

# set force convergence parameter and start search
relax.run(fmax=0.05)  # unit of force is eV / Ang

# print cell information
print("\nCell parameters:")
print(si.get_cell().round(5))
print("\nCell volume: ", si.get_volume())
```

2. Compare your result with the ones from fitting the energy-volume-curve. Also try another optimizer like `BFGSLineSearch`. How large is the difference in volume and lattice constant in each case?
   Solution:
   Slight difference in volume, also between relaxations using different optimizers like BFGS vs. BFGSLineSearch, but basically the same lattice constant.

More information on optimization algorithms:
https://wiki.fysik.dtu.dk/ase/ase/optimize.html
https://wiki.fysik.dtu.dk/gpaw/tutorialsexercises/structureoptimization/stress/stress.html

Appendix

Additional material for the EOS task:

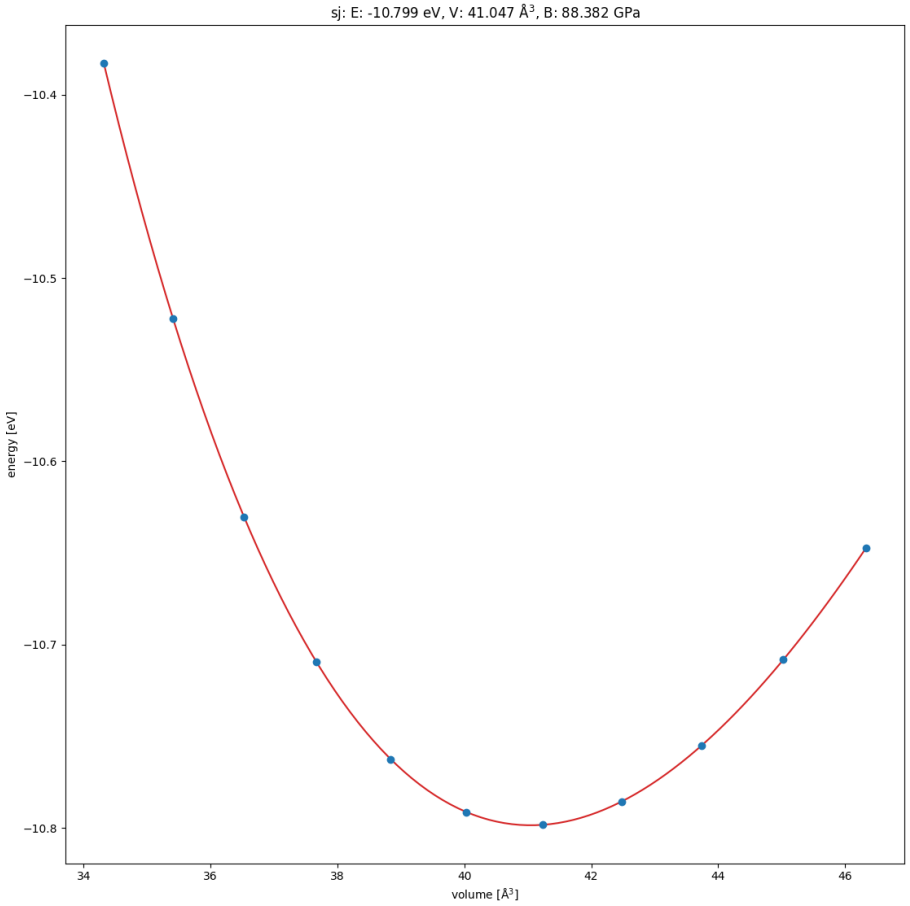

Figure 1: Bulk modulus via ase-gui.

The bulk modulus from thermodynamics is defined by

$$B = -V\frac{\partial p}{\partial V}\,, \quad \text{with} \quad p = -\frac{\partial E(V)}{\partial V}\,. \tag{1}$$

Inserting the (volume/lattice constant)-relation for primitive fcc crystals, $V = a^3/4$, we obtain

$$B = \frac{4}{9a_0}\left.\frac{\partial^2 E(a)}{\partial^2 a}\right|_{a=a_0} \quad \left(\text{using} \quad \frac{\partial V}{\partial a} = \frac{3a^2}{4}\right) \tag{2}$$

for the bulk modulus in terms of the conventional (equilibrium) lattice constant.