Introduction to bulk properties

KEMS409 — Demo #2

Useful links

- **ASE** atomic simulation environment

  https://wiki.fysik.dtu.dk/ase/

- **GPAW** grid-based projected augmented wave

  https://wiki.fysik.dtu.dk/gpaw/

- **AMCSD** American mineralogist crystal structure database

  http://rruff.geo.arizona.edu/AMS/amcsd.php

- **bulk tests** using the GPAW approach

  https://wiki.fysik.dtu.dk/gpaw/setups/bulk_tests.html

Problems

- construct and visualise given metal lattices

  #2.1 – **Na** (bcc)
  #2.2 – **Ag** (fcc)
  #2.3 – **Mg** (hcp)

- check what sampling of the Brillouin zone is sufficient *(remember that it is merely a demo, and we are learning here — no need to achieve research-class accuracy)*

- relax the metal lattice

- apply an equation of state to calculate the bulk modulus

Connect to Electra

- connect to Electra with the display forwarding (otherwise, you won't be able to visualise the results)

```
ssh -Y -l <username> calc.phys.jyu.fi
ssh -Y electra.chem.jyu.fi
```

- connect to one of the nodes exclusively allocated for today's demo session or for homework

```
ssh -Y el33                   ssh -Y el1
ssh -Y el34                   ssh -Y el2
ssh -Y el37
```

Problem #2.1 – the Na bulk

```
from ase.lattice import bulk
from ase.visualize import view

a0 = 4.225    # educational guess for the lattice constant
Me = “Na”     # symbol of the metal

cell = bulk(
        name=Me,
        crystalstructure=“bcc”,
        a=a0,
        cubic=True)

view(cell)    # visualising the unit cell
```

Brillouin zone sampling

- each periodic lattice has a corresponding reciprocal lattice
- a Wigner–Seitz cell on the reciprocal lattice is the first Brillouin zone (or BZ)
- the Brillouin zone is typically sampled by a Monkhorst–Pack type mesh of *k*-points

Testing the BZ sampling

```
from ase.lattice import bulk
from gpaw import GPAW, PW

cell = bulk(...)   # generate an appropriate unit cell here!

for k in [ 1, 2, 4, 8 ]:
    calc = GPAW(
            xc="PBE",                           # XC functional
            mode=PW(400),                       # plane-wave cutoff
            kpts=(k,k,k),                       # MP-mesh of k-points
            eigensolver="rmm-diis")             # special eigensolver
    cell.set_calculator(calc)
    cell.get_potential_energy()
```

Results for the Na bulk

- submit the calculation first (parallel)

```
mpirun -np 4 gpaw-python <name>.py | tee <name>.txt
```

- extract the results

```
grep -E “the Brillouin Zone|Zero Kelvin” <name>.txt

1 k-point ...
Zero Kelvin:     -6.532187
1 k-point ...
Zero Kelvin:     -3.322692
4 k-points ...
Zero Kelvin:     -2.571802
20 k-points ...
Zero Kelvin:     -2.597907
```

...continued

- calculate the relative change in energy upon gradual improve of the BZ sampling

sampling	**IBZ**	**energy (eV)**	**change**
(1x1x1)	1	-6.5322	–
(2x2x2)	1	-3.3227	49.1%
(4x4x4)	4	-2.5718	22.6%
(8x8x8)	20	-2.5979	1.0%

- starting from the (4 × 4 × 4) Monkhorst–Pack mesh, the change in energy gets small, indicating the sufficiency of the selected mesh *(for the demo purposes!)*
  — we will use that for further calculations on the Na bulk

Relaxing the lattice

- the educational guess for the lattice parameter does not necessarily correspond to the minimum on the potential energy surface within the chosen computational approach
- we might use already familiar BFGS algorithm to relax the structure; however it can only optimise the positions of the atoms within the cell
- hence, we ought to employ something that projects forces acting on the lattice onto the forces acting on the atoms — in ASE it is called “strain filter”

Relaxing the Na bulk

```
from ase.optimize import BFGS
from ase.io import Trajectory
from ase.constraints import StrainFilter
from gpaw import GPAW, PW

cell = bulk(...)   # generate an appropriate unit cell here!

calc = GPAW(xc="PBE", mode=PW(400),
        kpts=(4,4,4),              # apply the chosen MP mesh!
        eigensolver="rmm-diis")
cell.set_calculator(calc)

sf = StrainFilter(cell)
opt = BFGS(sf, logfile="<name>.log")
traj = Trajectory("<name>.traj", "w", cell)
opt.attach(traj)

opt.run(fmax=0.025)
```

Results for the Na bulk

- submit the calculation first (parallel)

```
mpirun -np 4 gpaw-python <name>.py | tee <name>.txt
```

- open the log file to check if the optimisation is converged

```
cat <name>.log

BFGS:   0  ........        -2.571802       0.0760
BFGS:   1  ........        -2.571878       0.0646
BFGS:   2  ........        -2.572078       0.0008
```

- visualise the optimisation trajectory

```
ase-gui <name>.traj
```

...continued

- extract the optimised unit cell from the text output

```
grep “Unit Cell:” -A 5 <name>.txt
```

```
Unit Cell:
           Periodic     X              Y              Z        Points  Spacing
  --------------------------------------------------------------------
  1. axis:    yes    4.207378    -0.000000    0.000000    20     0.2104
  2. axis:    yes   -0.000000     4.207378    0.000000    20     0.2104
  3. axis:    yes    0.000000     0.000000    4.207378    20     0.2104
```

- the optimised lattice parameter is a = **4.2074 Å** (to be compared to the experimental guess a_0 = 4.225 Å)

Stabilised jellium equation of state

- we can apply the stabilised jellium equation of state (SJ-EOS) to calculate the bulk modulus of our material

$$E(V) = c_0 + c_1 t + c_2 t^2 + c_3 t^3, \text{ where } t = V^{-1/3}$$

- to ensure a good fitting, we need to specify several energy values (in this demo — seven) corresponding to different unit cell volumes

- the seven points will correspond to the deformation of the lattice by 0%, ±1%, ±2% and ±3% with respect to the optimised lattice

…continued

- ASE-GUI will then do the job finding the minimum t_{min} of the $E(V)$ function and the derivative dE/dt, wherefrom
  - the equilibrium volume $V_0 = t_{\mathrm{min}}^{-3}$
  - the bulk modulus $B = 1/9 \times (t_{\mathrm{min}}^5 \times \mathrm{d}E/\mathrm{d}t(t_{\mathrm{min}}))$

Bulk modulus of Na

```
from ase.io import Trajectory
from gpaw import GPAW, PW

cell = bulk(...)   # generate an appropriate unit cell here!
                   # remember to use the optimised lattice
                   # parameter!
ucell = cell.get_cell()    # save the original unit cell

calc = GPAW(xc="PBE", mode=PW(400), kpts=(4,4,4),
        eigensolver="rmm-diis")
cell.set_calculator(calc)

traj = Trajectory("<name>.traj", "w")
for delta in [ 0.97, 0.98, 0.99, 1.00, 1.01, 1.02, 1.03 ]:
    cell.set_cell(ucell * delta, scale_atoms=True)
    cell.get_potential_energy()
    traj.write(cell)
```

Results for the Na bulk

- submit the calculation first (parallel)

```
mpirun -np 4 gpaw-python <name>.py | tee <name>.txt
```

- open the saved trajectory (i.e. data points with different volumes)

```
ase-gui <name>.traj
```

- select Tools — Bulk Modulus to see the fitting of the stabilised jellium equation of state to your data and to retrieve the equilibrium volume V_0 and the bulk modulus B

...continued

- for the Na bulk
  - V_0 = 74.484 Å^3
  - B = 7.699 GPa
- which compares well to the experimental data
  - V_0 (exp) = 74.088 Å^3
  - B (exp) = 7.6 GPa

Follow up

- now repeat each step for

(2.2) – **Ag** (fcc)

```
cell = bulk(name=“Ag”, crystalstructure=“fcc”,
            a=..., cubic=True)
```

(2.3) – **Mg** (hcp)

```
cell = bulk(name=“Mg”, crystalstructure=“hcp”,
            a=..., c=...)
            # the unit cell is non-cubic!
            # we need to specify two lattice parameters!
```

Solutions

(2.2) – **Ag** (fcc)

a = 4.1814 Å
V_0 = 73.279 Å^3
B = 87.317 GPa

(2.3) – **Mg** (hcp)

a = 3.2228 Å
c = 5.0821 Å
V_0 = 45.608 Å^3
B = 38.143 GPa

Homework

- relax the lattice and calculate the equilibrium volume and the bulk modulus for the following metals (method: PBE functional, PW(400), ($4 \times 4 \times 4$) Monkhorst–Pack mesh of *k*-points, RMM-DIIS eigensolver)

  (2.4) – **Mo** (bcc), (2.5) – **Al** (fcc) and (2.6) – **Be** (hcp)
  (2.7) – **Fe** (bcc), (2.8) – **Ni** (fcc) and (2.9) – **Co** (hcp)
  these structures are ferromagnetic! — you must employ spin-polarised calculations! — make sure you converge the wave-function to a ferromagnetic state!

- return a short report on your results by **April 20, 1 p.m.**

Hints

- to enable the spin polarisation, add to your calculator

```
calc = GPAW(..., spinpol=True)
```

- to drive the wave-function to a ferromagnetic solution, you need to apply starting magnetic moments μ_0 to the atoms in the unit cell prior to attaching the calculator

```
cell = bulk(...)
cell.set_initial_magnetic_moments([...])
# as the argument you must supply an array of N magnetic
# moments, where N is the number of atoms per unit cell
```

- find the experimental values for μ in the literature and use those in your guess