1 Ground state calculation using GPAW

1. Prepare two cell objects, one in conventional and the other in primitive configuration.
2. Set GPAW as calculator for both cells with some reasonable settings like:
   https://wiki.fysik.dtu.dk/gpaw/tutorialsexercises/electronic/bandstructures/bandstructures.html
   - Plane wave (PW) mode with an energy cut-off of 400 eV (~30 Ry)
   - Monkhorst-Pack k-point grid of $8 \times 8 \times 8$ points
   - Exchange-correlation functional PBE (= version of GGA) or LDA
   - File where text output is written to

   Solution:

```
# import necessary (sub-)modules from the gpaw package
from gpaw import GPAW, PW
# set details for the DFT calculation, try: txt='cell.out'
cell.set_calculator(GPAW(mode=PW(400), xc='PBE', kpts=(8, 8, 8)))
```

3. Find help text, then ask GPAW to calculate the ground state of your system
   Solution:

```
# print description of a function in interactive python shell
cell.get_potential_energy?
# tell the program to calculate the total energy
epot = cell.get_potential_energy()
```

4. Repeat the calculation and time it
   Solution:

```
# time calculation
import time
start = time.time()
# reset calculator b/c potential energy is cached
cell.set_calculator(GPAW(mode=PW(400), xc='PBE', kpts=(8, 8, 8)))
cell.get_potential_energy()
```

5. Compare the total energy of both setups. Can you find a reason for the huge difference in energy? How much is the difference exactly? What would be a reasonable quantity to be used for comparing both calculations?
   Solution:
   The huge energy difference is a consequence of the different crystal configurations. The conventional cell is 4 times larger than the primitive one and contains 8 instead of 2 atoms. Thus, total energies should differ exactly by a factor of 4. A good way to compare calculations with different setups of the same system is to calculate the energy density in units of total energy per atom. Note, that in general, the computational cost does not depend on the number of atoms but more precisely scales with the total number of (valence-) electrons N in the system setup. DFT, for instance, scales worst-case with $\mathcal{O}(N^3)$, whereas most quantum chemical methods are more expensive and scale at least with $\mathcal{O}(N^4)$.

2 Calculate and plot ground state density

DFT is based on the Hohenberg-Kohn theorem, which states that all information about the quantum mechanical system is contained within one essential quantity: the ground state charge density $n(\boldsymbol{x})$. In the following, we will calculate this quantity and plot it using the already familiar VESTA tool.
https://wiki.fysik.dtu.dk/gpaw/tutorialsexercises/wavefunctions/plotting/plot_wave_functions.html
https://wiki.fysik.dtu.dk/gpaw/tutorialsexercises/wavefunctions/all-electron/all_electron_density.html

1. Write wave functions to file for later reference (CUBE format)
   Solution:

```
# repeat the calculation for the (conventional) CIF-cell and ...
# A) write out all wave functions to the same file
cell.calc.write('Si.gpw', mode='all')
```

   Optional: Write all WFs to different files
   Solution:

```
# B) write each wave function to its own file
nbands = cell.calc.get_number_of_bands()
for band in range(nbands):
    wf = cell.calc.get_pseudo_wave_function(band=band)
    fname = f'wf_{band}.cube'
    print('writing wf', band, 'to file', fname)
    write(fname, cell, data=wf) # write is a function from ase.io
```

2. Get pseudo and all-electron density
   Solution:

```
# make your life easier and introduce a shortcut
calc = cell.calc
# save the pseudo and all-electron density in new variables n and np
np = calc.get_pseudo_density()
# refine grid for all electron density
# -> possible values are 1, 2 [=default] and 4
gridref = 4
n = calc.get_all_electron_density(gridrefinement=gridref)
```

3. Integrate both densities numerically and interpret the output
   Solution:

```
# store volume of cell in a variable
vol = cell.get_volume()
# WFs are not stored on the "k-grid", but on a separate one in real
# space with a different number of grid points!
numpts = calc.get_number_of_grid_points().prod()
# calculate volume elements
dv = vol / numpts
# "discrete integration" equals summing up; volume elements can be
# pulled out of the sum since they are constant
Ip = np.sum() * dv
I = n.sum() * dv / gridref**3
```

Interpretation (taken from the manual):

> As the all-electron density has more structure than the pseudo-density, it is necessary to refine the density grid used to represent the pseudo-density. This can be done using the `gridrefinement` keyword of the `get_all_electron_density` method.
>
> The all-electron density will always integrate to the total number of electrons of the considered system (independent of the grid resolution), while the pseudo density will integrate to some more or less arbitrary number.

4. Write densities to file and plot them using VESTA:
   a) Compute volumetric data (e. g. wave function or density) for plotting (task 1 and 2).
   b) In VESTA, load CIF/Structure first.
   c) Import plot data via:
   `Edit` → `Edit Data` → `Volumetric Data` → `Isosurfaces` → `Import`
   d) Set iso value to a number that results in a nice plot:
   `Style Tab` → `Properties…` → `Isosurfaces` → `Isosurface Level` → `+0.5`

Solution:

```
# load the write module from ASE library
from ase.io import write
# write both densities in CUBE file format -> can be read by VESTA
write('Si-n.cube', cell, data=n)
write('Si-np.cube', cell, data=np)
```

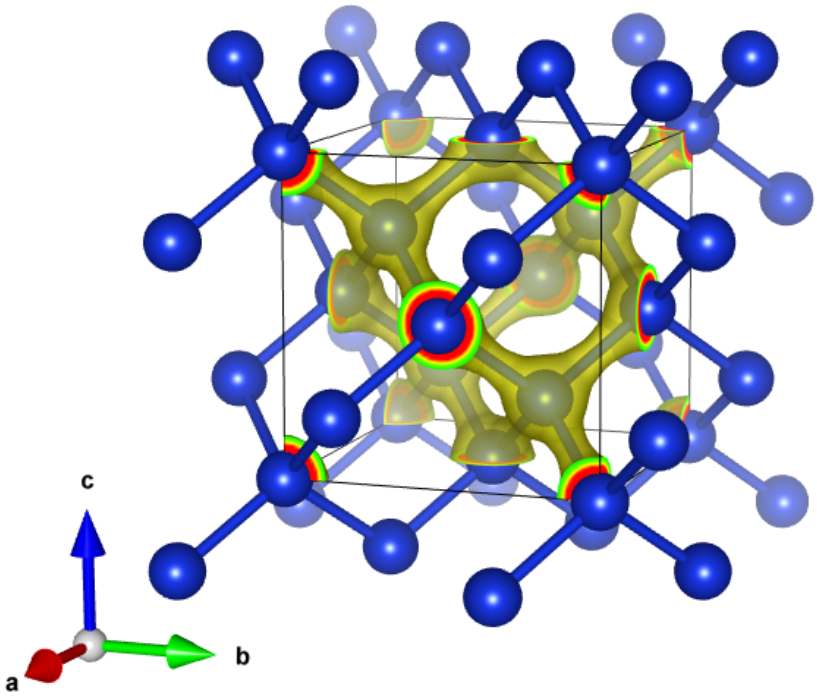

Figure 1: All-electron density of Si in diamond structure computed with GPAW and plotted as iso-surface on top of CIF structure (includes symmetry). As expected, the highest density is centered around the core positions.