Assignment 2 – Computational Materials and Molecular Physics
Second hand-in

Eric Lindgren – CID: Ericlin

February 18, 2020

1 Task 1: Run the genetic algorithm

The genetic algorithm was succesfully run for Na clusters with 6, 7 and 8 atoms respectively.

2 Task 2: What does the scripts do?

2.1 initialization.py

Initializes the system for a given number N of Na atoms. The script is composed of a main function which initializes the system, using several helper functions. First, arguments given to the script are parsed, and a database file (gadb.db) and a Lennard-Jones-potential-based calculator (calc) are created. Then, the cluster is created with N Na-atoms. The function 'closest_distances_generator' is used to find the smallest possible initial bond lengths, which are then passed to the class 'StartGenerator' which takes care of creating starting candidates for our Na cluster. After the slab parameters, which define the volume available to our system, of our StartGenerator are set we use StartGenerator to generate population_size number of starting candidates. These are generated by first adding an atom at (0,0,0), after which the rest of the N atoms are added iteratively by randomly trying different angles and positions next to the already placed atoms. The atoms are placed at the bond-distance calculated using the 'closest_distances_generator' from ASE. Each obtained starting cluster is then relaxed using the Lennard-Jones potential. Finally, all of the these candidates are saved to the database file. The candidates are considered unrelaxed because they haven't been properly relaxed with GPAW yet.

2.2 ga.py

Takes the output from initialization.py. First, all of the unrelaxed candidates fron gadb.db are loaded, and a GPAW-based calculator is defined (calc) which uses the defualt LDA XC-functional. Then, they are each relaxed for 100 steps using calc and saved back to gadb.db. These relaxed candidates are also added to our population-object. Then, the genetic algorithm (GA) starts. A genetic algorithm is an algorithm inspired by the idea of evolution; that individuals exhibiting the

best traits have a greater chance of survival and spreading their traits, as well as the possibility of random mutations to occur. In our specific case, for each of the ncandidates_to_test, two random candidates out of the population are selected and "paired" to create a third candidate. "Pairing" in this sense means that atoms from both parents are "sliced-out" and combined to form a new candidate. With a certain probablilty mutation_probability a random mutation is also added to the new candidate by performing some operation, such as mirroring, rattle or permutation of the atoms. The new candidate is then relaxed, and added to the population after which the process is repeated.

Finally, all relaxed candidates of the population are saved to the file 'all_candidates.traj'.

3 Task 3: Expected outcome

Create three clusters of 6, 7 and 8 Na-atoms respectively, configured according to my expectation. I expect the atoms to be placed as close to each other as possible and in a symmetric manner, in order to minimize the energy of the system. Thus I guess that the 6 Na-atom structure will be in a 3D pyramid-like shape with two points. For the 7 and 8 Na-atom clusters I have no real intuition, and use the random-placement-get-out-of-jail-card and place the atoms randomly.

See appendix A.1 for the script `task3.py`. Note that I used the defualt LDA functional with an LCAO basis and a grid spacing of $h = 0.25$; i.e. the same settings as in `ga.py` from task 1.

4 Task 4: Relaxation

It is insufficient to only relax the structures since we are only testing a limited number of configurations of our system, making it possible to get stuck in a local energy minima in configuration space.

5 Task 5: Extracting the results

Using the script `task5.py`, see appendix A.2, we identify the most stable structures for the different number of Na atoms by iterating through all entries in their respective databases.

The most stable structure for Na_6 with energy $E \approx -4.9339\,\text{eV}$ is given in figure 1. It is 2D and shaped sort of like an arrowhead.

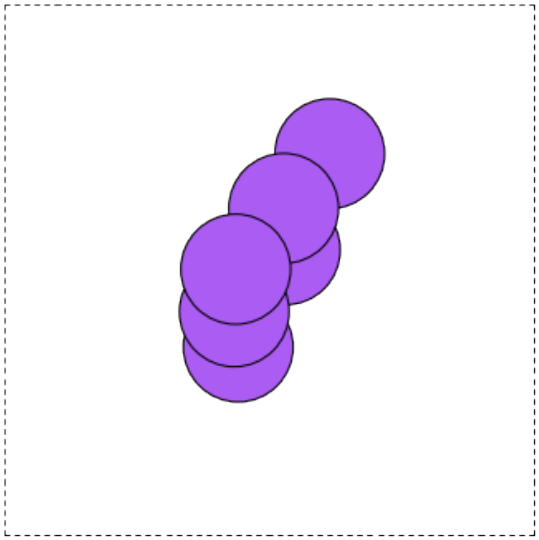


Figure 1: The obatined most stable structure for Na_6. Note the distictly 2D shape.

For Na_7 the most stable structure is decidedly 3D, and shaped somewhat like a double pentagonal pyramid (if that makes sense?). See figure 2. The minimum energy was $E \approx -5.8734\,\text{eV}$.

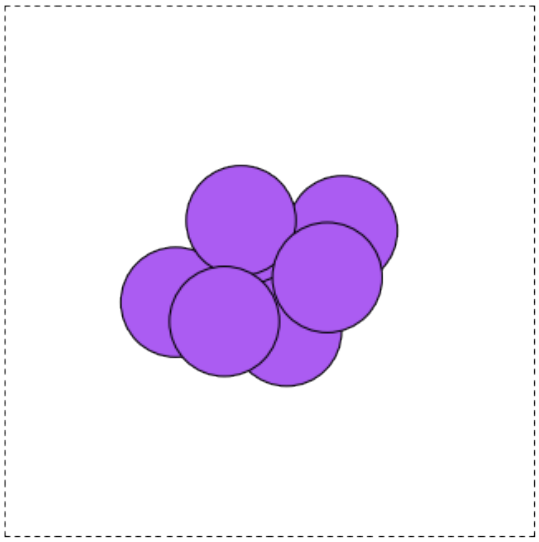


Figure 2: The obatined most stable structure for Na_7. Note the distictly 3D shape.

For Na_8 the most stable structure is also 3D, but less regularly shaped ran Na_7. See figure 3. The minimum energy was $E \approx -7.2812\,\text{eV}$.

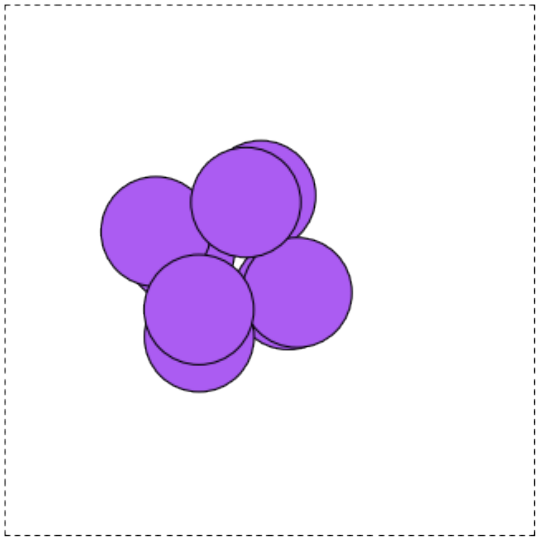


Figure 3: The obatined most stable structure for Na_8. Note the distictly 3D shape.

6 Task 6: Flat versus bulky

Having a flat structure essentially means that we constrain our system to a lower dimensional system (2D instead of 3D), but one that is longer. If we think of the wavefunctions of the atoms in our system as living in a particle of a box, i.e. we think of the cluster as a simplified potential, we would guess that the energy of the system was proportional to $E \propto \frac{1}{L^2}$ as for the particle-in-a-box. Thus, going from 2D to 3D (increasing the length of the system) would lower the energy, but it would simultanously increase the energy due to increased exchange effects. Since the electrons are fermions, they can't occupy the same area of space. Removing one degree of freedom (2D) means that some electrons must occupy higher energy orbitals than in 3D, which increases the energy of the system. This could be one explanation as to why Na_6 is 2D whilst Na_7 and Na_8 are 3D; the added number of electrons gives rise to exchange effects that are larger than the gains of an increased length of the system when going from 3D to 2D.

One possible explanation as to why this doesn't happen for the Lennard-Jones potential could be that it was designed for noble gases, not metals. Thus, the terms in the LJ-potential corresponds to exchange effects (repulsion, r^{-12}) and Van-der-Waals forces (attraction, r^{-6}). Since metal bonding is quite difference to VdW-bonds, it is not so strange that DFT and the LJ-potential yields different results. One argument as to why the DFT result could be more realistic is that often the electrons in a metal can be modelled as free electrons in a periodic potential, in which the energy levels are $E \propto \frac{1}{L^2}$ (see e.g. P. Hoffman Solid State Physics - Chapter 6 [1]). This strengthens the validity of the analogy to the particle in a box as discussed in the previous section. Because of DFT methods based around the Kohn-Sham equations by construction yields the correct electron densitites for the system (with an exact XC-functional), it is possible that the DFT method results in a description of the system that is more realistic and thus akin to the case of electrons in a periodic potential, as compared to the VdW interaction that the LJ-potential models. Hence, we

would expect the DFT-based calculations to yield transitions to 2D structures for smaller clusters.

7 Task 7: First and second most stable structures

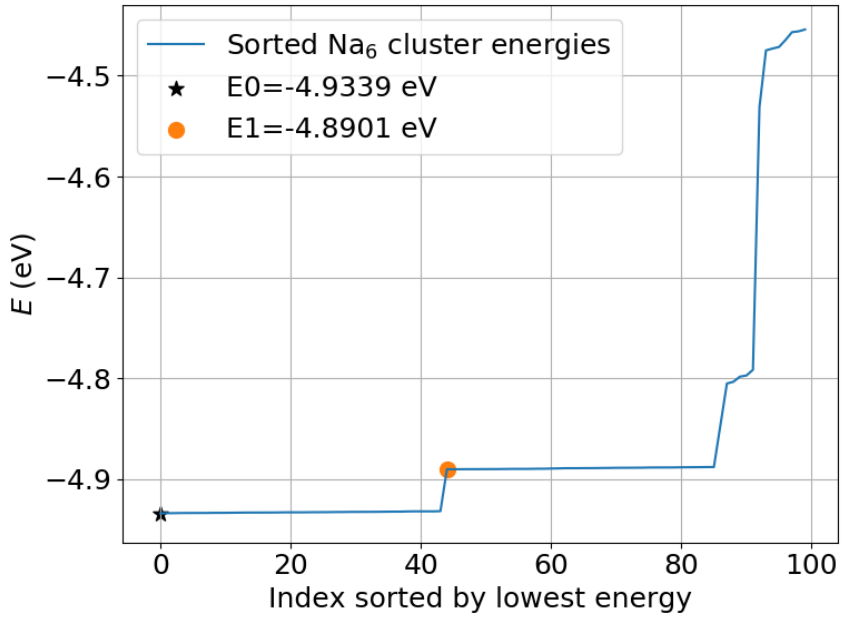


Figure 4: Sorted energy for the various obtained clusters for Na_6. The most stable and second most stable clusters are marked by a black star (E_0) and an orange circle (E_1) respectively. A threshold energy difference between the clusters of $\delta E = 0.04\,\text{eV}$ was used to differentiate between the most stable cluster and the second most stable cluster. I.e. the second most stable cluster is the one that has an energy of $+\delta E$ up from the most stable state.

In a similar manner as in problem 5, the second most stable structures for the clusters of 6, 7 and 8 atoms were identified. I defined a threshold difference in energy as compared to the ground state that another configuration would have to surpass in order for it to be considered another stable state. I chose $\delta E = 0.04\,\text{eV}$. See figure 4 for the energy trajectory plotted versus cluster configuration for Na_6 with the energy of the most stable structure E_0 and of the second most stable structure E_1 plotted in the graph. Using this method, the following differences in energy between the most stable- and second most stable state ΔE where:

- Na_6: $\Delta E = 0.0438\,\text{eV}$
- Na_7: $\Delta E = 0.1446\,\text{eV}$
- Na_8: $\Delta E = 0.0410\,\text{eV}$

Thus we see that the difference in energy is at the $\sim 0.1\,\text{eV}$ scale. In `ga.py` we used a LCAO-dzp basis (more on this in Task 8), which according to lecture 3 [2], slide 18, exhibits an error of $\leq 1\,\text{eV}$ as compared to experimental results. If I've interpreted the slide correctly, this is a systematic accuracy error and not an error of precision; i.e. DFT gives results which are reliable between runs, but that differs from the experimental value by some constant. This makes sense since DFT in itself is not a stochastic method; the only stochastic method we've used here is the genetic algorithm. Thus we would expect DFT to converge to the same values reliably, but since the XC-functional is approximated these values should be systematically off from the experimental value. In other words, DFT is a method which exhibits high precision but lacks in accuracy. Hence we would expect DFT to be able to identify a difference between stable states on the order of $\sim 0.1\,\text{eV}$.

A further indication of DFT being reliable can be seen in the next section, Task 8, specifically in tables 1 and 2; over multiple runs both the energy of the most stable and the second most stable states converge to their respective energies up to a precision of $\sim 10^{-4}\,\text{eV}$ independently of the GPAW settings used.

8 Task 8: A closer look at Na_6

Here I relax the structures for Na_6 in the repository folder, using a script based on the script from problem 3. I will try the following basis sets, functionals and modify these parameters:

- GPAW modes: Plane waves (PD, cutoff energy at 350 eV and at 500 eV), finite differences (FD, $h = 0.25$ Å) and LCAO for Na (dzp basis).
- Functionals: PBE, vdW-DF-xc
- Parameters: nbands=10, 15

This yields a total of 16 table rows, which can be seen in table 1. vdW-DF-xc is a non-local functional developed by among others Per Hyldgaard, which he talked about on his lecture in this course. I thought it would be interesting to try it out on Hebbe, even though the system doesn't really need to model vdW-forces.

We note that the change of GPAW mode seems to have a large impact on the ground state energies for the two clusters. Both when using FD and PW (for both E_{cut}:s tested) mode the cluster which is the most stable state for LCAO is the second most stable state. That we obtain a difference in what configuration is the most stable indicates that what basis the wavefunctions of the electron's in the system are expressed in affects the result of the simulation to a fairly great extens. This will be further discussed in Task 9 (next section).

Changing the functional between PBE and vdW-DF-xc has a very large impact on the result. PBE yields results which are of similar energy scale as what was obtained with LDA in the previous tasks ($\sim -5\,\text{eV}$), whilst vdW-DF-xc gives a ludicrous ground state energy of $\sim +17\,\text{eV}$ for both clusters. This is not even a valid bound state energy which must be negative, which indicates that the vdW-DF-xc functional is not very suited for handling the system of 6 Na-atoms. As mentioned earlier, this could be due to the fact that modelling the system as vdW-interacting

is not very accurate description, and highlights the importance of choosing a functional which properly describes the physics that one expects from the system in question. An interesting aspect is that the vdW-DF-xc functional performs even worse than the LJ-potential used to relax a similar system (lecture 1), even though the LJ-potential is also tailored for vdW-interacting systems. This could be due to the vdW-DF-xc functional being more advanced and thus more niched towards vdW-interacting systems, and thus performs worse for other systems.

The number of electron bands also seems to have little impact. There is a slight difference in energy for both the most stable cluster and the second most stable cluster on the order of $\sim 10^{-6}$ eV when going from 10 to 15 bands when GPAW is in FD or PW mode, but none when in LCAO mode. The reason as to why this has such a small effect could be that 10 bands is enough to handle our system; Na is fairly simple with only one valance electron.

Thus we conclude that the Na cluster configuration which is the most stable is dependent on the GPAW parameters, specifically the mode that the calculator is in.

Table 1: Results from playing around with the modes, functionals and parameters in GPAW. E_0 refers to the obtained energy of the most stable cluster, and E_1 to the energy of the second most stable cluster. Note that the change of GPAW mode seems to have the largest effect, whilst changing the number of bands has a small impact for FD and PW mode. Changing the functional seems to have no impact what so ever with this level of precision (8 decimals). Continuation on the next page in table 2

GPAW configuration	E_0 (eV)	E_1 (eV)
Mode: LCAO XC: PBE nbands = 10	-4.47404674	-4.40113244
Mode: LCAO XC: vdW-DF-cx nbands = 10	17.92457227	18.00156087
Mode: LCAO XC: PBE nbands = 15	-4.47404674	-4.40113244
Mode: LCAO XC: vdW-DF-cx nbands = 15	17.92457227	18.00156087
Mode: FD XC: PBE nbands = 10	-4.84462920	-4.88085744
Mode: FD XC: vdW-DF-cx nbands = 10	17.43594192	17.39378701
Mode: FD XC: PBE nbands = 15	-4.84204826	-4.88085488
Mode: FD XC: vdW-DF-cx nbands = 15	17.43594186	17.39378648
Mode: PW, E_{cut} = 350 eV XC: PBE nbands = 10	-4.85117757	-4.88665521
Mode: PW, E_{cut} = 350 eV XC: vdW-DF-cx nbands = 10	17.40845496	17.38217644
Mode: PW, E_{cut} = 350 eV XC: PBE nbands = 15	-4.85119747	-4.88665966
Mode: PW, E_{cut} = 350 eV XC: vdW-DF-cx nbands = 15	17.40845289	17.38217588

Table 2: Continuation from table 1, containing the results from PW basis with a higher cut-off energy, $E_{cut} = 500\,\text{eV}$.

GPAW configuration	E_0 (eV)	E_1 (eV)
`Mode`: PW, $E_{cut} = 500\,\text{eV}$ `XC`: PBE `nbands` = 10	-4.85120531	-4.88669350
`Mode`: PW, $E_{cut} = 500\,\text{eV}$ `XC`: vdW-DF-cx `nbands` = 10	17.40835874	17.38208374
`Mode`: PW, $E_{cut} = 500\,\text{eV}$ `XC`: PBE `nbands` = 15	-4.85130726	-4.88669096
`Mode`: PW, $E_{cut} = 500\,\text{eV}$ `XC`: vdW-DF-cx `nbands` = 15	17.40835552	17.38208393

9 Task 9: Which calculation is the most accurate?

In the last section, Task 8, it was concluded that changing the basis mode for GPAW between FD, PW and LCAO mode could greatly affect the results of a simulation. Using LCAO mode resulted in a flat 2D structure having the lowest energy, whilst a 3D structure was the second most stable. Using either PW or FD mode resulted in these two changing place, with the 3D structure being more stable than the 2D structure. According to the lecture, using plane waves (PW) or finite differences (FD) as a basis for the wavefunctions generally result in very accurate results, whilst an LCAO basis is typically considered more efficent but less accurate [2]. According to this point of view, the PW or the FD based methods can be considered as ”better” for simulating this system. But as I argued for during Task 6, it seems reasonable that the most stable cluster for Na_6 would be a 2D shape. That would then imply that an LCAO basis is better suited for this system than a FD or PW basis. The reason as to why an LCAO basis would describe the system better could be because of the system being comprised of the same kinds of atoms all bunched together. Inuitively, it would then seem reasonable for the total wavefunction of the system to be well approximated by a linear combination of the atomic orbitals (LCAO) from each Na atom. Hence it is possible that the fairly simple LCAO basis which I used, double zeta polarized (dzp) which includes two basis functions for each atomic orbital, is sufficient for a simulation that is accurate enough to capture the clustering behavior of the system. Since neither the FD or the PW basis are as ”native” to this problem it is possible that one would need to for example decrease the grid spacing h for FD or increase the cutoff energy for PW in order for them to become as accurate as LCAO for this particular system.

In order to obtain an even more accurate simulation one could either use other LCAO basis sets with more basis functions per orbital, such as TZP (triple zeta polarized). But since FD and PW are renowned for being more accurate than LCAO, for a really accurate simulation where

computational efficiency is not an issue I would use one of them with a really small grid spacing h or a very high cut-off energy respectively. An indication that one might obtain the same most stable cluster structure with PW as for LCAO by increasing the cutoff energy is that increasing the E_{cut} from 350 eV to 500 eV seems to lower the energy E_0 by $\sim 1 \cdot 10^{-4}$ eV whilst E_1 is only decreased by $\sim 4 \cdot 10^{-5}$ eV, see tables 1 and 2. Thus it is possible that increasing E_{cut} further might make $E_0 \leq E_1$, i.e. that PW yields the same most stable structure as LCAO.

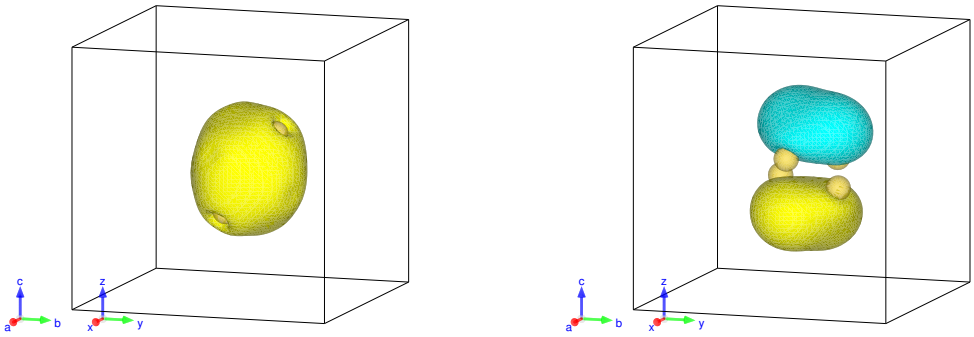
Other methods than DFT which could yield more reliable results could be the use of Hartree- or Hartree-Fock based methods. Especially Hartree-Fock is interesting, since it perfectly models the exchange effects (Pauli repulsion) between the electrons due to the use of a Slater-determinant based basis. The exchange can only be approximated in DFT (until someone finds the perfect XC functional...). That Hartree- and HF based-methods can yield better results than DFT could for example be seen in Home Assignment 1 of this course, where both Hartree and Hartree-Fock yielded results closer to the experiemntal value for the ground state of helium than the DFT-based methods. However, the Hartree- and HF-based methods exhibits poorer scaling with larger systems than DFT, which might make the use of these methods somewhat intractible for our system (especially given how much time we are expected to spend on this assignment ☺).

10 Task 10: Looking at the wave functions

Using the script `task10.py` given in Appendix A.4, the most stable system for each of Na_6, Na_7 and Na_8 where relaxed again and their wavefunctions saved into cube files. For this simulation I used an LCAO dzp-basis with the defualt XC-functional LDA and a grid spacing of $h = 0.25$ Å. Please see `task10.py` for any implementational specific details.

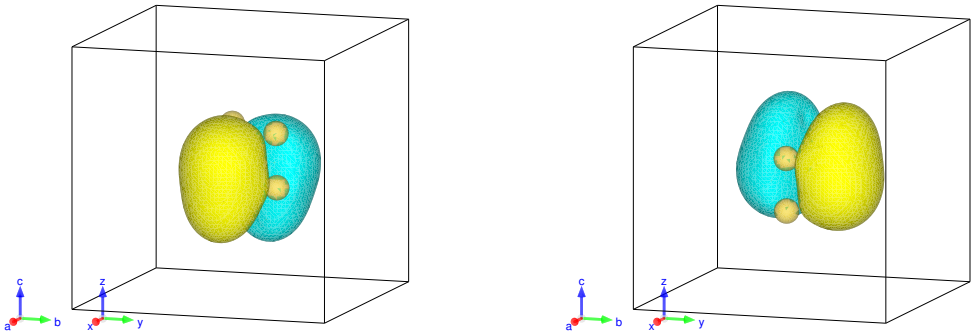
The computed wavefunctions were viewed using VESTA. From the GPAW outfiles, we find that only the three first bands are occupied for Na_6 with occupancy (2,2,2), the first four for Na_7 with occupancy (2,2,2,1) and the first four for Na_8 with occupancy (2,2,2,2). The occupied orbitals for Na_8 where plotted using VESTA and can be seen in figures 5 and 6.

The orbitals looks somewhat like the atomic orbitals; Band 0 looks like 1s, whilst Bands 1–3 looks like the 2p orbitals, but with the bands aligned against different axis. The occupied orbitals for Na_6 and Na_7 where much more complex, and as we saw from task 5 they had higher energies for their most stable configurations than Na_8. The fact the Na_8 is the lowest energy cluster of the three could be attributed to it's atomic-like orbitals; the system looks somewhat "atomic" rather than a molecule. As in an atom, the electrons are distributed in a symmetrical manner around the cluster, which increases the possible configurations of the electrons. Thus these configurations have a high multiplicity and are rather probable, and hence more stable. We also note that all electron bands are occupied for Na_8, which means that the possible number of electrons which have the same spin and thus can exchange their position freely are at a maximum, which lowers the energy from exchange effects as well as increasing the multiplicity of the configuration further. The exchange energy is also further lowered by the fact the densities for bands 1–3 are orthogonal to each other. All of this combined yields a very nice, stable cluster which exhibits a lot of symmetries. Hence, this might be the reason as to why it is called a magic cluster.



(a) Band 0 (b) Band 1

Figure 5: The wave functions for the first and second occupied electron bands of Na_8. Note that the bands are indexed starting with band 0. Notice that the wavefunctions looks like the 1s and 2p orbitals.



(a) Band 2 (b) Band 3

Figure 6: The wave functions for the third and fourth occupied electron bands of Na_8. Note that the bands are indexed starting with band 0. Notice that the wavefunctions looks like the 1s and 2p orbitals.

References

[1] P. Hoffman. *Solid State Physics – An introduction*. Wiley-VCH, 2016.

[2] A. Hellman. *Lecture 3 – Basis sets XC*. Retrieved on 05-02-2020. 2020. URL: %5Curl%7Bhttps://chalmers.instructure.com/files/358124/download?download_frd=1%7D.

A Python scripts

A.1 task3.py

Please run in the folder "task3".

```python
# Built-in packages
import time

# Third-party packages
import numpy as np

from ase import Atoms
from ase.ga.utilities import closest_distances_generator
from ase.visualize import view
from ase.optimize import GPMin

from gpaw import GPAW, FermiDirac

def create_relaxed_Na_cluster(N, view=False):
    print(f'************ Na{N} ************')
    start = time.time()
    #**** Initialize system ****#
    if N==6:
        clust = Atoms('Na'*6, positions=[(1,1,0),(1,-1,0),(-1,-1,0),(-1,1,0),(0,0,1)↩
            ,(0,0,-1)], cell=(d, d, d))
    else:
        clust = Atoms('Na'*N, positions=[np.random.randn(3) for i in range(N)], cell↩
            =(d, d, d))  # random initialization
    clust.center()
    if view:
        view(clust)
    #**** Define the calculator ****#
    calc = GPAW(nbands=10,
                h=0.25,
                txt=f'Na{N}_out.txt',
                occupations=FermiDirac(0.05),
                setups={'Na': '1'},
                mode='lcao',
                basis='dzp')
    #**** Relax the system ****#
    clust.set_calculator(calc)
    dyn = GPMin(clust, trajectory=f'Na{N}_relax_clust.traj', logfile='Na{N}↩
        _relax_clust.log')
    print(f'**** Relaxing system of {N} atoms ****')
    dyn.run(fmax=0.02, steps=100)
    #**** Calculate energy and wavefunction ****#
    e = clust.get_potential_energy()  # Note opposite signa from ga.py
    e_file = open(f'Na{N}_e_cluster.txt', 'w')
    print(f'Na{N} cluster energy: {e} eV', file=e_file)
    calc.write(f'Na{N}_cluster.gpw', mode='all')
    end = time.time()
    print(f'**** Elapsed time: {end-start} s ****')
    print('*****************************\n')
    #*****************************#

Ns = [6,7,8]
for N in Ns:
    create_relaxed_Na_cluster(N)
```

A.2 task5.py

Please run in the folder "task5".

```python
# General imports
import numpy as np
import matplotlib.pyplot as plt

# ASE
from ase.db import connect
from ase.io import write
from ase.visualize import view

# Plot details
plt.rc('font', size=18)          # controls default text sizes
plt.rc('axes', titlesize=18)     # fontsize of the axes title
plt.rc('axes', labelsize=18)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=18)    # fontsize of the tick labels
plt.rc('ytick', labelsize=18)    # fontsize of the tick labels
plt.rc('legend', fontsize=18)    # legend fontsize


def get_most_stable_cluster(db):
    id_min, E_min = 0, 0
    for row in db.select(relaxed=True):
        # Find lowest energy candidate
        if row.energy < E_min:
            E_min = row.energy
            id_min = row.id
    print(f'Minimum energy for Na{N} cluster: e = {E_min:.4f} eV with id={id_min}')
    a = db.get(f'id={id_min}').toatoms()
    return a

def view_results_and_to_xyz(N):
    dirpath_t1 = f'../task1/Na{N}/'
    dirpath_t5 = f'.'
    db = connect(f'{dirpath_t1}gadb.db')
    a = get_most_stable_cluster(db)
    #**** View atoms - save to a png file ****#
    view(a)
    write(filename=f'{dirpath_t5}Na{N}_min.png', images=a)
    #**** Save atoms to XYZ ****#
    write(filename=f'{dirpath_t5}Na{N}_min.xyz', images=a)

Ns = [6,7,8]
for N in Ns:
    view_results_and_to_xyz(N)
```

A.3 task8.py

Please run in the folder "task8".

```python
# Built-in packages
import time
import os

# Third-party packages
```

```
import numpy as np

from ase import Atoms
from ase.ga.utilities import closest_distances_generator
from ase.visualize import view
from ase.optimize import GPMin
from ase.parallel import paropen
from ase.parallel import parprint
from ase.io import read

from gpaw import GPAW, FermiDirac, PW


def test_params_Na6(m, f, nb, idx):
    print(f'************ Na6 - Mode: {m} - Functional: {f} - nbands={nb} ************')
    start = time.time()
    structpath=f'../Na-clusters-GA-search/Na6-structures/'
    #**** Initialize system ****#
    clust0 = read(filename=f'{structpath}christmas-tree.xyz', format='xyz')
    clust1 = read(filename=f'{structpath}half-decahedron.xyz', format='xyz')
    #**** Define the calculator ****#
    if m=='pw350':
        calc = GPAW(nbands=nb,
                    h=0.25,
                    xc=f,
                    txt=f'outfiles/Na6_{m}_{f}_{nb}_out.txt',
                    occupations=FermiDirac(0.05),
                    setups={'Na': '1'},
                    mode=PW(350))
    elif m=='pw500':
        calc = GPAW(nbands=nb,
                    h=0.25,
                    xc=f,
                    txt=f'outfiles/Na6_{m}_{f}_{nb}_out.txt',
                    occupations=FermiDirac(0.05),
                    setups={'Na': '1'},
                    mode=PW(500))
    elif m=='fd':
        calc = GPAW(nbands=nb,
                    h=0.25,
                    xc=f,
                    txt=f'outfiles/out.txt',
                    occupations=FermiDirac(0.05),
                    setups={'Na': '1'},
                    mode='fd')
    elif m=='lcao':
        calc = GPAW(nbands=nb,
                    h=0.25,
                    xc=f,
                    txt=f'outfiles/out.txt',
                    occupations=FermiDirac(0.05),
                    setups={'Na': '1'},
                    mode='lcao',
                    basis='dzp')
    #**** Relax the system ****#

    clust0.set_calculator(calc)
    dyn0 = GPMin(clust0, trajectory=f'trajectories/Na6_0_{m}_{f}_{nb}_relax_clust.traj', logfile=f'logfiles/Na6_0_{m}_{f}_{nb}_relax_clust.log')
    clust1.set_calculator(calc)
    dyn1 = GPMin(clust1, trajectory=f'trajectories/Na6_1_{m}_{f}_{nb}_relax_clust.
```

```
        traj', logfile=f'logfiles/Na6_1_{m}_{f}_{nb}_relax_clust.log')

    print(f'****  ID:{idx} - Relaxing 1st system of atoms    ****')
    dyn0.run(fmax=0.02, steps=100)
    print(f'****  ID:{idx} - Relaxing 2nd system of atoms    ****')
    dyn1.run(fmax=0.02, steps=100)

    #**** Calculate energy and wavefunction ****#
    e0 = clust0.get_potential_energy()  # Note opposite signa from ga.py
    e1 = clust1.get_potential_energy()
    ef = open(f'table/Na6_{idx}.txt', 'w')
    #**** Print to file *****#
    ef.write('-'*78 +'\n')
    ef.write(f'# Mode: {m}' + '|'.rjust(15-len(f'# Mode: {m}')) + '\n')
    subrow1 = f'# XC: {f}' + '|'.rjust(15-len(f'# XC: {f}'))
    subrow2 = f'{e0:.8f} eV \t\t {e1:.8f} eV'.rjust(50-len(subrow1))
    ef.write(subrow1+subrow2 + '\n')
    ef.write(f'# nbands = {nb}' + '|'.rjust(15-len(f'# nbands = {nb}')) + '\n')

    end = time.time()
    print(f'****    Elapsed time: {(end-start):.2f} s    ' + '****'.rjust(19))
    print('*'*77 + '\n')
    #*****************************#

# Parameters to perform grid search over
modes = ['pw350', 'pw500', 'fd', 'lcao']
functionals=['PBE', 'vdW-DF-cx']
nbands = [10, 15]

# Prepare table file with headers
ef = open(f'table/e_table.txt', 'w')
subheader1 = ' Parameters ' + '|'.rjust(15-len(' Parameters '))
subheader2 = '\t\t\t E0 \t\t\t\t  E1'
ef.write(subheader1+subheader2+'\n')
ef.close()

# Iterate over all parameters
print('*'*50 + '\tStarting calculation\t' + '*'*50)
start = time.time()
for i,m in enumerate(modes):
    for j,f in enumerate(functionals):
        for k,nb in enumerate(nbands):
            test_params_Na6(m, f, nb, i*100+j*10+k)
end = time.time()
print('#'*25 + (f'\tTotal time: {(end-start):.2f} s\t' + '#'*25).rjust(20))

# Print results to table file
ef = open(f'table/e_table.txt', 'a')
for filename in os.listdir('table/'):
    if filename.startswith('Na6_'):
        with open(f'table/{filename}') as f:
            for line in f:
                ef.write(line)
ef.write('-'*78)
ef.close()
```

A.4 task10.py

Please run in the folder ”task10”.

```python
# Built-in packages
import time

# Third-party packages
import numpy as np

from ase import Atoms
from ase.ga.utilities import closest_distances_generator
from ase.visualize import view
from ase.optimize import GPMin
from ase.db import connect
from ase.io import write

from gpaw import GPAW, FermiDirac

def get_most_stable_cluster(db):
    '''Fetches the most stable cluster in the db'''
    id_min, E_min = 0, 0
    for row in db.select(relaxed=True):
        # Find lowest energy candidate
        if row.energy < E_min:
            E_min = row.energy
            id_min = row.id
    print(f'Minimum energy for Na{N} cluster: e = {E_min:.4f} eV with id={id_min}')
    a = db.get(f'id={id_min}').toatoms()
    return a

def calculate_stable_wavefunction(N, ef, view=False):
    print(f'-------------- \t Na{N} \t --------------')
    start = time.time()
    #**** Initialize system ****#
    dirpath_t1 = f'../task1/Na{N}/'  # Path to the results from task 1
    db = connect(f'{dirpath_t1}gadb.db')
    stable_clust = get_most_stable_cluster(db)
    #**** Define the calculator ****#
    calc = GPAW(nbands=10,
                h=0.25,
                txt=f'Na{N}_out.txt',
                occupations=FermiDirac(0.05),
                setups={'Na': '1'},
                mode='lcao',
                basis='dzp')
    #**** Relax the system ****#
    stable_clust.set_calculator(calc)
    dyn = GPMin(stable_clust, trajectory=f'Na{N}_relax_clust.traj', logfile=f'Na{N}↩
        _relax_clust.log')
    print(f'****\tRelaxing system of {N} atoms\t****')
    dyn.run(fmax=0.02, steps=100)
    #**** Calculate energy and wavefunction ****#
    e = stable_clust.get_potential_energy()  # Note opposite signa from ga.py
    print(f'Na{N} stable energy: {e} eV', file=ef)

    #**** Write wavefunctions to cube files ****
    basename=f'Na{N}'
    nbands = calc.get_number_of_bands()
    for band in range(nbands):
        wf = calc.get_pseudo_wave_function(band=band)
        fname = 'Na{0}/{1}_{2}.cube'.format(N, basename, band)
        print(f'writing wf {band} to file {fname}----'.rjust(12))
        write(fname, stable_clust, data=wf)
    #**** Finish ****
```

```
    end = time.time()
    print(f'****     Elapsed time: {(end-start):.2f} s     ' + '****'.rjust(12))
    print('-------------------------------------------\n')
    #****************************#

Ns = [6,7,8]
ef = open('energies.txt', 'w+')
for N in Ns:
    calculate_stable_wavefunction(N, ef)
ef.close()
```