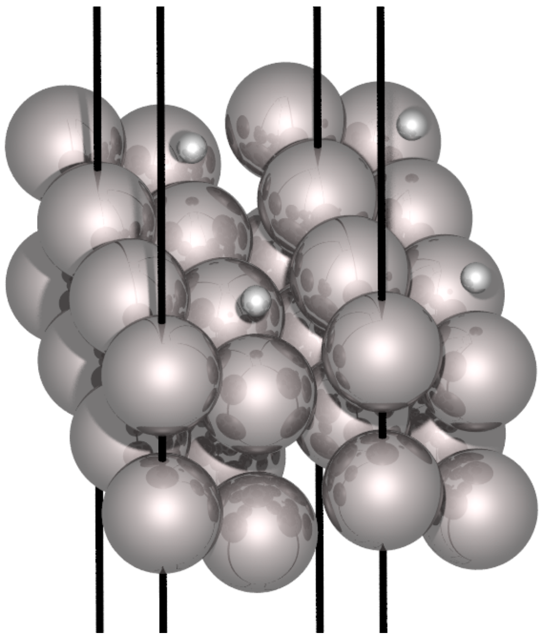
Special Course in Theoretical Physics PHYS-E0546: DFT for Practitioners

Aalto University, September 2019



Tutorial 6: Calculations of barriers and reaction coordinates

Manuscript for Exercise Problems

Prepared by Ondřej Krejčí, Orlando Silveira and Adolfo Fumega
Developed by Patrick Rinke and Xi Chen
Department of Applied Physics
Aalto University, 2022

Introduction

Calculations of the *minimum energy path* and therefore reaction barrier, are important parts of computational chemistry, material science as well as surface science. According to the Transition state theory and Arrhenius equation, the reaction rate k is exponentially depending on the energy barrier E_B:

$$k = A\ Exp\left(-\frac{E_B}{k_b T}\right), \tag{1}$$

where A is an exponential pre-factor, k_b is the Boltzmann constant and T is a temperature. The calculations of the barriers are important not only for estimation the speed of a reaction, but also for calculation stability of various structures as well as determination between different reaction paths.

In this Tutorial we will use DFT to calculate the *minimum energy path* and reaction barrier for an atomic diffusion over crystalline surface. The diffusion on surfaces can be essential in various material and surface science field, from material/molecules clustering, self-assembly to material growth and also catalysis. For example a high diffusion barrier can prevents reactants to meet each other, or meet the active sides of a catalyst and thus lower down the reaction rate considerably.

In part I of this tutorial, we will prepare and optimize the geometries necessary for the barriers calculations. To build the geometries used in this part you will use *Atomic Simulation Environment* (*ASE*) [1], which is a python library already installed in your environment. In case you find the error `No module named ase` when trying to use the library, try to install it by running `pip3 install ase`.

In part II, we will calculate barrier for the diffusion of a hydrogen atom on an Al(110) surface. Three different methods for the barrier estimation will be introduced and their results will be compared and discussed.

For every exercise we also provide solutions and sample input files. As always, they can be found in the `$TUTORIALS/` directory. Please try to generate the input files on your own unless instructed otherwise. In case you get stuck with a particular problem, do not hesitate to ask one of the tutors. An executable of FHI-aims, as well as species files and utility scripts, are provided in the `$COURSE/CODE/` directory.

Additional tools and programs

Utility scripts:
DFT output files are complex, so we use utility scripts to extract important data. In the

`$COURSE/CODE/utilities` directory you will find tools like `aims-SCF_convergence.awk` and `get_relaxation_info.pl` to monitor the simulation, as well as `create_xyz_movie.pl` to track the geometry optimisation. In this tutorial, additional scripts can be found in `Templates` directory for each problems.
`$TUTORIALS/Tutorial_6/utilities`.

Visualization tools:
To visualize structures, vibrational modes, charge density plots, etc., several programs including molden, vmd, and jmol are installed on your workstations. Please use the powerful *ASE GUI* to visualise the geometry as well as to modify it. You will find the tutorial in Appendix I.

Part I: H-atom diffusion on Al(110) surface

In the first part of this Tutorial you will mostly repeat things you already learned in Tutorial 3. The tests and workflow used in this Tutorial are necessary for any calculation with *Periodic Boundary Conditions*. You will learn how to easily build these system and how to add an adsorbate on top of a surface.

Educational Objectives

- Familiarized yourself with *python ASE* environment for creation and manipulation of atomic geometries.
- Learn how to use *ASE* to build bulk materials and surfaces.

Problem I: Finding ideal parameters for Al bulk

In this exercise, you will determine optimal lattice constant and number of k-points (or rather k-points density) for the aluminium fcc crystal lattice.

Tasks

1. Copy `run.py` from the `Templates` directory. Prepare `control.template` file, which will contain basic settings for the calculations: LDA exchange-correlation functional, spin-unpolarized calculations and no relativity, since Al is a light atom:

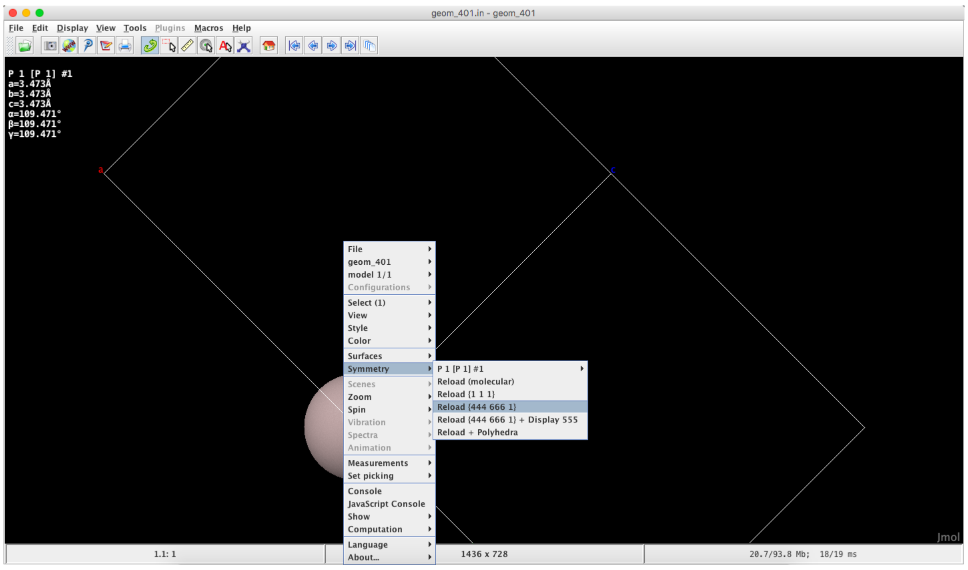
```
# Physical settings
  xc                  pw-lda
  spin                none
  relativistic        none

# SCF settings
  sc_accuracy_rho     1E-4
  sc_accuracy_eev     1E-2
  sc_accuracy_etot    1E-5
  sc_iter_limit       100

# k-points grid
  k_grid  XXX XXX XXX
```

   Adding the k-points, basis functions for aluminium and transforming this file into `control.in` will be done automatically later.

2. Create files for the Al bulk (fcc) geometry using *ASE* with different *elementary* lattice constants – from 3.95 to 4.03 Å. You can use: `python generate_geometries.py` from the `Templates` folder, but ideally you can try to generate them by yourself. From terminal run:

```
python
> from ase import atoms
> from ase.build import bulk
> from ase.io import write
> for i in [395,397,399,401,403]:
...     tmp = bulk( 'Al' , 'fcc' , a=0.01*i )
...     write('geom_'+str(i)+'.in',tmp); del tmp;
...# do not forget about tab or 4 spaces in front of 2 commands above.
> exit()
```

This procedure will create 5 `geom_XXX.in` files containing geometries for the following, calculations. Check one of them using `jmol`. You can click on the geometry with the right mouse button and using `symmetry/reload {444 666 1}` you can visualize also the neighbouring cells. You can see how to do it in Fig. 1.



```
P 1 [P 1] #1
a=3.473Å
b=3.473Å
c=3.473Å
α=109.471°
β=109.471°
γ=109.471°
```

Figure 1: How to visualize neighbouring cells in `jmol`.

3. Run the automatic script `run.py` to loop over the lattice constant and the number of k-points in the following manner: `python3 run.py 8` it will take a couple of minutes. It will automatically create a control.in file for each calculation using the `light` aluminium basis set. In the mean time you can try to read what the script `run.py` does.

4. Check if all the calculations completed without any mistake by looking to at least one of the `nok_XX/outputgeom_XXX.txt` files.

5. To obtain all energies from the output files run: `python3 plot_outputs.py` This script will plot the energies vs. the length of the lattice constant into `Results_energy.pdf`.

   What can you see from the results? What is the optimal lattice constant for the `light` basis set and `Tier 1`? Compare this results with experimental results; for example this one [2]. What is the necessary amount of k-points for getting good results? What is the optimal k-point density using an approximate formula:

$$\rho_k = \frac{2\pi}{a_p * N_k},$$

   where N_k is the number of k-points and a_p is the length of one the *primitive unit cell* vectors?

Optional tasks:

1. Try different xc-functionals (e.g. PBE). is there any difference? If necessary adjust the lattice constant range to clearly distinguish the minimum.

2. Try to change the run.py script so it would use tight or super tight basis sets. What is the difference?

Problem II: Creating an Al(110) surface slab with a hydrogen adsorbate.

In this problem you will create an Al(110) slab geometry with an H atom on top of it and relax it. For the creation of the geometries you will use the optimal lattice constant from Prob_I.

1. Create a new directory.

2. Prepare an Al(110) surface with a H adsorbent using *ASE*. You can use a script `create_surface.py` from a `Templates` directory to create your `geometry.in` file, but you can also try by yourself in the terminal. Do not forget to change the *XXX* with your calculated optimal *elementary* lattice constant:

```
python3
> from ase import atoms
> from ase.build import fcc110, add_adsorbate
> from ase.io import write
> from ase.visualize import view
> from ase.constraints import FixAtoms
> # !!! change XXX with the optimized lattice constant !!! #
> surf = fcc110('Al',(1,2,3),a=XXX,vacuum=15, periodic=True)
> view(surf)
> add_adsorbate(surf, 'H', 0.2, position='longbridge')
> c= FixAtoms(mask=[atom.position[2] < 16.0 for atom in surf])
> surf.set_constraint(c)
> view(surf)
> write('geometry.in',surf)
> exit()
```

   We chose a slab with 3 layers only, in order to speed up the calculations, but the real calculations would require a larger amount of Al layers. We use a 1×2 supercell, so the periodic images of the hydrogen atoms may not interact with each other. We added a hydrogen atom in the so-called *longbridge* position, which is in-between the Al(110) rows, as you can see in Fig. 2. Finally, we fixed only the lowest layer of the slab for the geometry relaxation, due to the low amount of layers. Normally, 2-3 layers of the slabs are fixed, during the optimization of the top-most layers

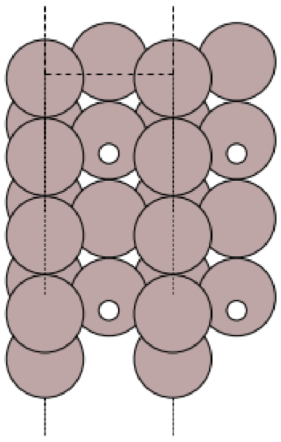


Figure 2: The 2×1 Al(110) slab with H adsorbate crated by *ASE*. To clearly see the periodic boundary conditions 2×2 cells are visualized.

3. Copy the `control.in` from the one of the `nok_XX` directories used in Prob_I. Open the `control.in` in a text editor and adjust the `k_grid` line to `k_grid 8 5 1`. Then add these

lines just bellow it:

```
# relaxation setting

  relax_geometry bfgs 3.e-2
```

Also add the basis set for the hydrogen atom:

```
cat $SPECIES/light/01_H_default >> control.in
```

4. Optimise the geometry with: `mpirun -n 6 aims.x | tee out.out`
5. Check the `geometry.in.next_step` after the run. You can simply open it with `jmol`. How good was our guess for the adsorbant height?

Part II: Barrier calculations

In the previous Part you created a geometry for the barrier calculations that you will use in the following problems.

Educational objectives

- Get an overview of possible barrier calculation methods such as: fixed linear scan, linear scan with atomic constraints and (climbing) NEB.

Problem III: Fixed line-scan

Here we will perform a fixed scan calculation along an atomic row.

Tasks

1. Create a new directory. First, copy `control.in`, `geometry.in.next_step` and `hessian.aims` from your Prob_II directory. Rename `geometry.in.next_step` to `geometry.in` and also copy over the `run.py` script from the `Templates` directory. Finally remove or comment this line: `relax_geometry bfgs 3.e-2` from your `control.in` file.

2. Execute the `run.py` script on 6 cores in the following fashion:

   ```
   python3 run.py 10
   ```

   This will take a couple of minutes. The script will create ten images and calculate their energy as the hydrogen is moved along the y direction on the surface. The hydrogen is always moved by 1/10 of the distance between two `'longbridge'` positions (y lattice vector of 1×1 unit cell). You are encouraged to look at the `run.py` script, while it is running. Try to understand what each line of the script does.

3. The results of the calculations are written in `energies_fixed_line_scan.txt` file. Visualize the results with your favourite plotting utility (e.g. `gnuplot`,`xmgrace`). Here is an example for `gnuplot`:

   ```
   gnuplot
   > set xlabel 'step'; set ylabel 'Energy␣[eV]'
   > plot 'energies_fixed_line_scan.txt' u 1:($2+39435) w lp
   ```

   What is the estimated barrier? Did we start from a proper minimum? You can also visualize the geometries writen in `fixed_path.xyz` e.g. with `jmol`. To visualize all the geometries in one image you can use the button shown in Fig. 3

Problem IV: Relaxed line-scan

In this exercise we will make our barrier calculations more precise – we will fix the y position of the hydrogen atom, but the atom can move in the x and z directions. All the other atoms can relax freely, except for the fixed bottom-most layer.

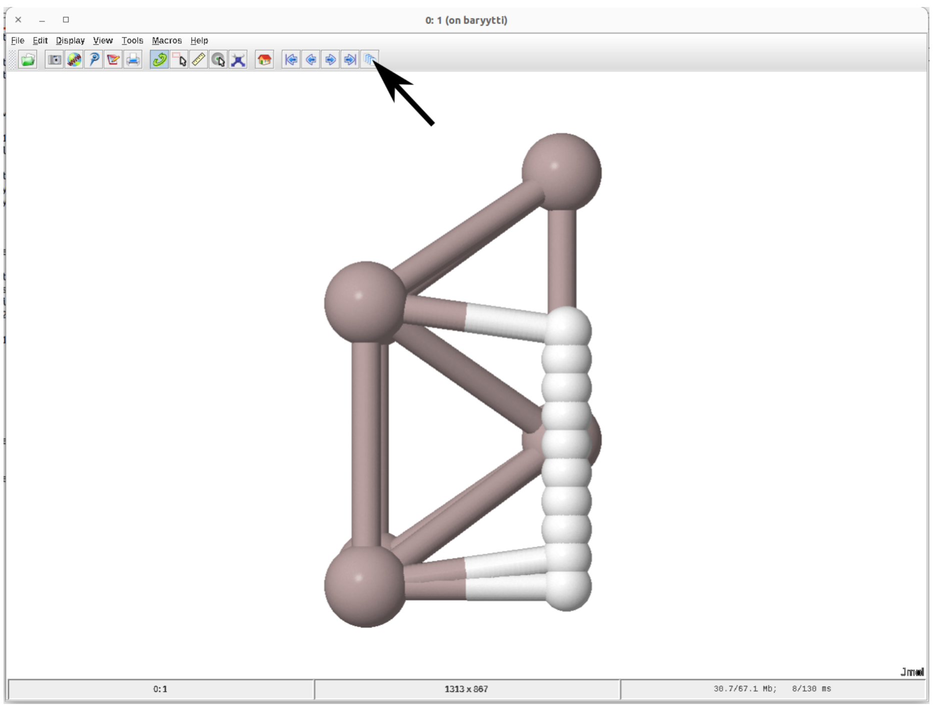


Figure 3: Clicking on this button in `jmol`, you can visualize all the gometries in any `xyz` file in one image.

Tasks

1. Create a new directory for this exercise. Once more copy `control.in`, `geometry.in.next_step` and `hessian.aims` from your Prob_II calculations and rename `geometry.in.next_step` to `geometry.in`. Copy the run script `run.py` from the `Templates` directory. To constrain the relaxation of the H atom to the y direction add this line into the geometry.in, right after the hydrogen atom:

   ```
   constrain_relaxation y
   ```

2. Now run the relaxed scan calculations with: `python3 run.py 6`. This will take longer time. During this time, please take a look at the `run.py` script. What are the differences in comparison with the previous problem? Later start to read and prepare files for the last problem Problem V: Nudged Elastic Band Calculations

3. After the calculations, you can clearly see the difference between the fixed and the the relaxed scan. First you can visualise the geometries in `relaxed_path.xyz` . Compare what you saw in `fixed_path.xyz` in the previous excercise. The difference becomes even more apparent when plotting both energy profiles together. You can use your favourite plotting option, here is a gnuplot example:

   ```
   gnuplot
   > set xlabel 'step'; set ylabel 'Energy␣[eV]'
   > plot 'PATH_TO_FIXED/energies_fixed_line_scan.txt' u 1:($2+39435)
   w lp t 'fixed␣scan', 'PATH_TO_RELAXED/energies_relaxed_line_scan.txt'
   u 1:($2+39435) w lp t 'relaxed␣scan'
   ```

   The last three lines should be written together.

   How big is the difference between the calculated barriers?

Problem V: Nudged Elastic Band Calculations

In the last problem, we will use the most complex method for finding the reaction coordinate and transition state: (Climbing Image -) Nudged Elastic Band. [3]

Tasks

1. For these calculations some initial settings are necessary: Therefore use:

   ```
   export PATH=$PATH:$COURSE/CODE/src/aimsChain/tools
   export PYTHONPATH=$PYTHONPATH:$COURSE/CODE/src/aimsChain
   ```

   To test, if all the settings are correct run

   ```
   python:
   > import aimsChain
   > import scipy
   > import numpy
   > exit()
   ```

   If there are no errors, then your system should be set properly.

2. Create a new directory. For the last time copy the `control.in` the from your Prob_II directory. Also copy `geometry.in.next_step` as `ini.in` . Remove last two lines from `ini.in` . In your new directory copy `ini.in` to `fin.in` and in this file change the hydrogen y -coordinate from `0.0000` to `2.821356056934325`. To check, that all the files are properly prepared, visualize them in `jmol`.

3. Now create the `chain.in` file, which controls the NEB calculations:

   ```
   #A sample chain.in    # next line: how aims will be runned
   run_aims mpirun -n 6 $COURSE/CODE/bin/aims.x
   initial_file ini.in  # what is the initial geometry file
   final_file fin.in    # what is the final geometry file
   n_images 3           # the amount of NEB images
   force_thres 0.2      # threshold for stopping optimisation
   use_climb true       # usage of climbing NEB method ?
   climb_thres 0.05     # threshold for climbing image optimisation
   ```

4. Before lauching the NEB calculations, copy `runchain.py` from the `Templates` directory. Then execute:

   ```
   python3 runchain.py 10
   ```

   It will take about five minutes.

5. To get the energies from the optimised path you can use the following python script:

   ```
   python3 plot_NEB_output.py
   ```

   Take a look at `Results_energy.pdf` or look at the text file: `NEB_energies.txt`.

   Compare the NEB calculated barrier with previous computations. You can also visualise the pathway. There is an `xyz` file: `Optimised/path.xyz` .

Optional tasks:

1. You can try to use NEB calculation to estimate the barriers for diffusion in between the atomic row or diagonally. Do not forget, that you will need to create new super-cells for

these calculations (2×1 or 2×2) and thus also change the amount of k-points in `control.in`. What is the difference between the diffusion barriers in different directions?

2. You can try to change the amount of the images in the (climbing) NEB calculations. How the estimated barrier is depending on the number of NEB images? How is the optimisation behaves?

Appendix I: How to use ASE GUI

ASE-GUI (Grafic User Interface) is an atomic geometry visualization software build in the `python-ASE` package. For more information about the GUI
`https://wiki.fysik.dtu.dk/ase/ase/gui/gui.html?highlight=gui#module-ase.gui`.

Open *ASE-GUI* by typing:

```
ase gui
```

which will open a clean GUI in which you can open almost any geometry file through `file/open`, or immediately read almost any geometry file, via:

```
ase gui YOUR_GEOMETRY_FILE
```

If any of the previous ways is not working, you can always go through python:

```
python
> from ase.io import read
> from ase.visualize import view
> temp = read('YOUR_GEOMETRY_FILE')
> view(temp)
```

Once the GUI reads a geometry, it visualize the geometrys as balls only. You can change the mode to the balls-and-sticks through `View/Show bonds` or by *CTRL+B*. Example of both modes can be seen in Fig. 4.

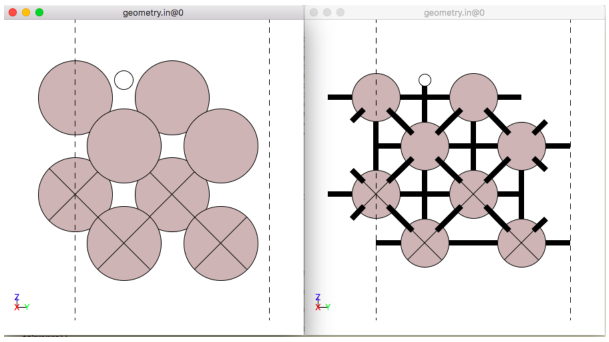


Figure 4: ASE-GUI Standard mode of geometry visualization (left) and balls-and-sticks mode (right).

You can rotate the visualized structure through holding the right mouse button and dragging, or you can used some of the preset view-points: top-view can be obtained by pressing z, front-view by x and rotate side-view by y. You can see the example in Fig 5.

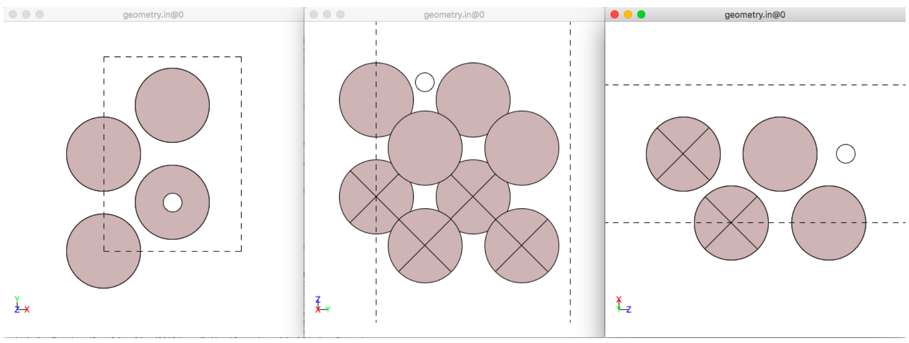
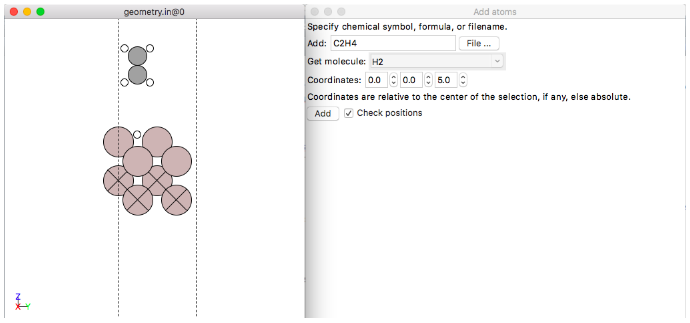


Figure 5: ASE-GUI top, front and side view of the same structure

By holding the left-mouse button and dragging you can select several atoms of the visualized geometry. You can move (*CTRL+M* and use arrows), rotate (*CTRL+R* and use arrows) or you can apply constraints (`Tools/Constraints/Constrain Selected Atom`) to the selected atoms. You can also remove selected atoms, with *Backspace*.

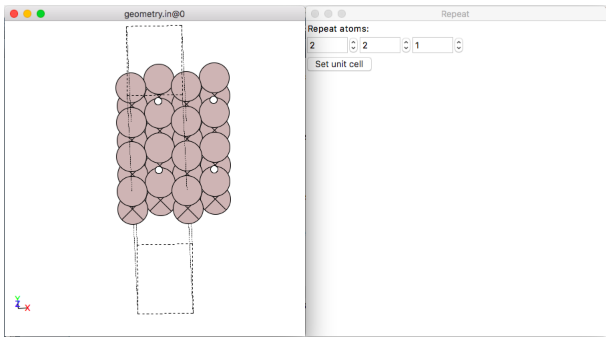
You can also add new atoms or molecules into the geometry through `Edit/Add atoms` or *CTRL+A*. The example for adding an ethylene molecule is shown in Fig. 6



Specify chemical symbol, formula, or filename.
Add: C2H4 File ...
Get molecule: H2
Coordinates: 0.0 0.0 5.0
Coordinates are relative to the center of the selection, if any, else absolute.
Add [✓ Check positions]

Figure 6: ASE-GUI adding an ethylene molecule 5Å above selected atom.

One of the most important features – cell multiplier – for checking a structure with *Periodic Boundary Conditions* can be obtained by pressing *r*. You can choose how many unit cells are repeated in which direction (Fig. 7). If you want to make a new supercell from the visualized structure, just push `Set unit cell` .



Repeat atoms:
2 2 1
Set unit cell

Figure 7: ASE-GUI repeating the unit cell in any-direction.

You can save the prepared structure to an `xyz`, `geometry.in`, other formats or even to an `png` image just by pushing *CTRL+S*

References

[1] https://wiki.fysik.dtu.dk/ase/ .

[2] https://www.webelements.com/aluminium/crystal_structure.html .

[3] Graeme Henkelman, Blas P. Uberuaga, and Hannes Jónsson. "A climbing image nudged elastic band method for finding saddle points and minimum energy paths". In: *The Journal of Chemical Physics* 113.22 (2000), pp. 9901–9904. DOI: 10.1063/1.1329672. eprint: http://dx.doi.org/10.1063/1.1329672. URL: http://dx.doi.org/10.1063/1.1329672.