QUANTUM ESPRESSO FOUNDATION

QUANTUMESPRESSO

MaX School on Advanced Materials and Molecular Modelling
with QUANTUM ESPRESSO

QE-2021: Hands-on session – Day-3

(Structural relaxations + NEB; PWTK)

MaX

ICTP The Abdus Salam International Centre for Theoretical Physics

CINECA

上海大学 SHANGHAI UNIVERSITY

Jožef Stefan Institute Ljubljana, Slovenia

cecam

SISSA

QE-2021: Hands-on session – Day-3

Topics of Day-3 hands-on session:

- Structural optimizations:
  - relaxations – atomic positions only (`example1.relax`)
  - variable-cell relaxations (`example2.vc-relax`)
- NEB method: saddle points of elementary chemical reactions (`example3.neb/`)
- Automating the workflow with PWTK (`example4.pwtk`)

To get the latest version of the exercises, move to `Day-3/` directory and execute:

```
$ git pull
```

How to run calculations remotely on the "hpc" HPC cluster

Several utility commands have been implemented specially for the QE-2021 school to aid at submitting jobs to HPC cluster(s). These are:

- **remote_mpirun** – this is like `mpirun`, but it automatically submits the calculation to a queuing system on the "hpc" HPC system. For example, a `pw.x` calculation can be submitted as:

  ```
  $ remote_mpirun pw.x -inp pw.file.in
  ```

  where `pw.file.in` is the name of the `pw.x` input file. **BEWARE:** stdin/stdout redirection does not work for `remote_mpirun`, hence you must use `-inp` option (i.e., do note use "<" redirection operator). You do not need to specify the number of processors, because the default is set to `-np 8`.

- **remote_pwtk** – this automatically submits the PWTK script to queuing system on the "hpc" HPC system. Example:

  ```
  $ remote_pwtk script.pwtk
  ```

  where `script.pwtk` is the name of the PWTK script.

- **remote_sbatch** – automatically submits the Unix-shell script to queuing system on the "hpc" HPC system. Example:

  ```
  $ remote_sbatch script.sh
  ```

  where `script.sh` is the name of the Unix-shell script.

- **hpc** – this makes `ssh` to "hpc" HPC login node, such that the user will be located in the same directory as used locally
- **rsync_to_hpc** – copies specified files to the "hpc" cluster to the same directory as is currently used locally. Example:

  ```
  $ rsync_to_hpc *.in
  ```

  This will copy all `*.in` files from local directory to the same directory on the "hpc" cluster.
- **rsync_from_hpc** – download the specified file from the "hpc" cluster from the same directory as is currently used locally. Example:

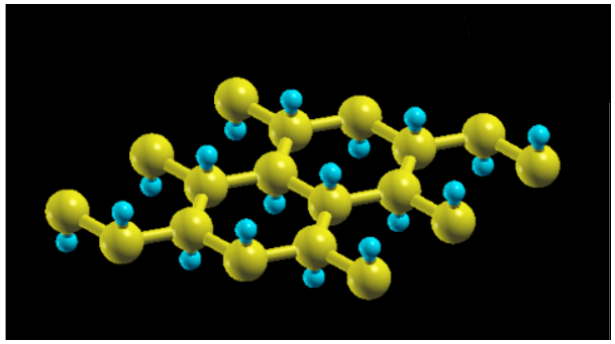
  ```
  $ rsync_from_hpc *.out
  ```

  This will copy all `*.out` files from the "hpc" cluster.

1. How to perform structural optimization: graphane

- Move to `Day-3/example1.relax/` directory.

  Graphane is like graphene, with an H atom bound to each C atom in *trans* configuration. You need to optimize atomic positions, i.e., find the minimum-energy structure (zero forces).



- File `pw.graphane.relax.in` is a modified version of `pw.graphene1x1.scf.in` with:
  - `calculation='relax'` for structural optimization and a new namelist `&IONS` with variable `upscale=100.0`
  - `ntyp=2` (2 types of atoms), `nat=4` (4 atoms in the cell)
  - `ATOMIC_SPECIES` card with 2 species of atoms and pseudopotentials
  - `ATOMIC_POSITION` card with 4 initial positions (C–H distance ~ 1 Å)

1. How to perform structural optimization: graphane (II)

- Run the structural optimization, i.e.:

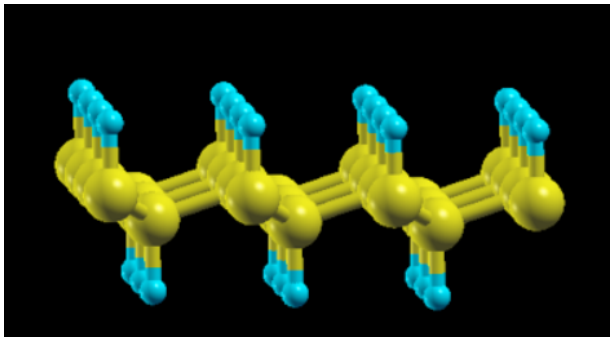
```
$ pw.x < pw.graphane.relax.in > pw.graphane.relax.out &
```

- When calculation finishes, analyze the output: it consists of several SCF steps, followed by calculation of forces and generation of new atomic positions.

- To visualize the evolution of the structure during structural optimization, execute:

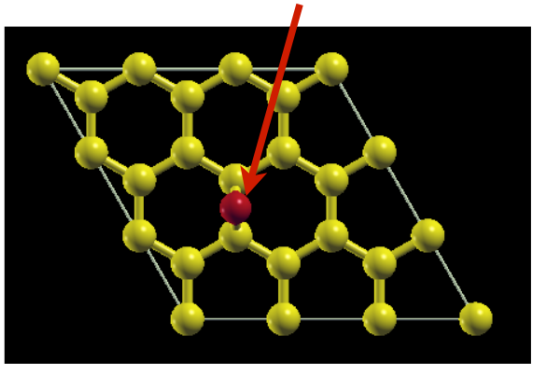
```
$ xcrysden --pwo pw.graphane.relax.out
```

Relaxed structure of graphane exhibits "buckling".



1. Supercell and structural optimization: graphene-oxide

- The first stage of graphene oxidation is the formation of an epoxy bridge. Let us add an O atom on a (3×3) supercell of graphene



This example consists of two tasks: (i) build a (3×3) supercell of graphene; (ii) add an O atom onto graphene–(3×3) supercell structure and run a relaxation calculation.

Step-1:

- The (3×3) supercell structure of graphene is provided in file `pw.graphene3x3.scf.in`, which is a modified version of `pw.graphene1x1.scf.in`. Please notice that:
  - Lattice parameters a and b are multiplied by 3: `celldm(1)=13.962`
  - Lattice parameter c remains the same, hence `celldm(3)`, which equals c/a, is divided by 3, hence: `celldm(3)=1.0`.
  - There are 9 times the atoms of the original unit cell, i.e. `nat=18`.

- Reciprocal lattice vectors in the xy plane are divided by 3 (look at the output), hence if you want the same k-point grid, just use `K_POINTS (automatic)` with `3 3 1 0 0 0` grid, which is equivalent to `9 9 1 0 0 0` k-point grid of (1×1) unit-cell
- Provided that the two k-point grids are equivalent:
  * The energy of the supercell $E^{\mathrm{SC}} = 9E^{\mathrm{UC}}$ almost exactly (UC = unit cell)
  * all $\epsilon^{\mathrm{SC}}(\mathbf{k}_i)$ are (almost) equal to some $\epsilon^{\mathrm{UC}}(\mathbf{k}_j)$ if $\mathbf{k}_j$ refolds into $\mathbf{k}_i$

Step-2:

- File `pw.graphene3x3-0.relax.in` is a modified version of `pw.graphene3x3.scf.in` with:
  - `calculation='relax'` for structural optimization and a new namelist `&IONS`
  - `ntyp=2` (2 types of atoms), `nat=19` (19 atoms in the cell)
  - `ATOMIC_SPECIES` card with 2 species of atoms and pseudopotentials
  - `ATOMIC_POSITION` card with 19 initial positions (C–O distance ~ 1.5 Å)

- Run the structural optimization and analyze the output

2. How to perform variable-cell relaxation: hcp-Zinc

Zinc displays a hcp (hexagonal-closed-packed) crystal structure, hence it has two lattice parameters a and c. The unit-cell lattice vectors are:

$$\mathbf{a}_1 = (a, 0, 0), \quad \mathbf{a_2} = (-\frac{a}{2}, \frac{a\sqrt{3}}{2}, 0), \quad \mathbf{a}_3 = (0, 0, c)$$

This lattice can be described as:

- `ibrav=4`, `A=`a, `C=`c, *both in Å, not a.u.*, as in file `pw.Zn.scf.in`
- or `ibrav=4`, `celldm(1)=`a, `celldm(3)=`c/a, as in the file `pw.Zn.vc-relax.in`

For hexagonal lattices one needs to optimize two lattice parameters. This can be done either *manually* or by using the *variable-cell relaxation*. (See also `README.md`).

1. **Manual way:** to optimize the a and c lattice parameters, one need to perform a 2D scan over the two parameters. With PWTK this can be achieved with the following snippet (full script is available in `Zn-scan.pwtk`):

```
foreach A [seq 2.4 0.1 2.8] {
   foreach C [seq 4.8 0.2 5.6] {
      SYSTEM " A = $A , C = $C"
      runPW pw.Zn.scf.$A.$C.in
   }
}
```

2. **Variable-cell relaxation:** This is a more convenient option. An example of how to perform variable-cell relaxation is provided by the input file `pw.Zn.vc-relax.in`. Notice:

   - `calculation = 'vc-relax'`
   - `&IONS` and `&CELL` namelists after the `&ELECTRONS`

To run the calculation, execute:

```
$ pw.x -in pw.Zn.vc-relax.in > pw.Zn.vc-relax.out
```

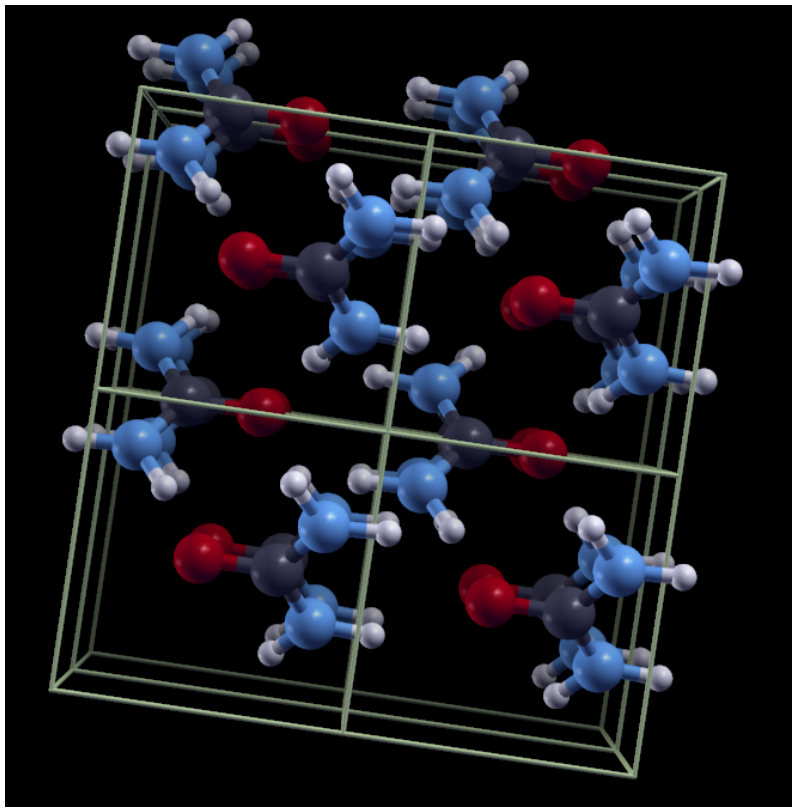
Inspect the output (file: `pw.Zn.vc-relax.out`) and notice that:

- several scf steps are performed, forces (zero by symmetry) and stresses computed
- the energy and the stress decrease as the minimum is approached
- a final scf step is performed with plane waves computed for the final cell
- the final cell is printed after the last `CELL_PARAMETERS` card

Compare optimized parameters estimated from the *manual* 2D-scan to those obtained from the *variable-cell relaxation*.

2. Variable-cell relaxation (II): molecular crystal of Urea

While in the preceding example, the forces were zero by symmetry, in this example (`pw.urea.vc-relax.in`) both unit-cell and atomic positions are optimized by utilizing the *variable-cell relaxation* (`calculation = 'vc-relax'`). Beware that it is computationally heavier than the `pw.Zn.vc-relax.in` example.



See the instructions in `README.md` for how to run this calculation remotely.

3. NEB method: saddle points of elementary chemical reactions

Saddle points on the *Potential Energy Surface (PES)*, which correspond to *Transition States (TS)* of chemical reactions, can be found by means of the *Nudged Elastic Band (NEB)* method.

In the first example we will analyze a simple H transfer reaction:

$$\mathrm{H_2} + \mathrm{H} \rightarrow \mathrm{H} + \mathrm{H_2}$$

Example 1a: No intermediate image

In the NEB method one needs to supply a minimum of two images, the so-called **first-** and **last-image**, corresponding to reactants and products.

In `neb.H2+H.in`, you will see that near the end of the file `FIRST_IMAGE` and `LAST_IMAGE` are specified. Seven images are requested (`num_of_images = 7`) along the reaction and the `neb.x` code discretizes the path by means of a linear interpolation.

You can visualize this path with `xcrysden`:

```
$ xcrysden --pwi neb.H2+H.in
```

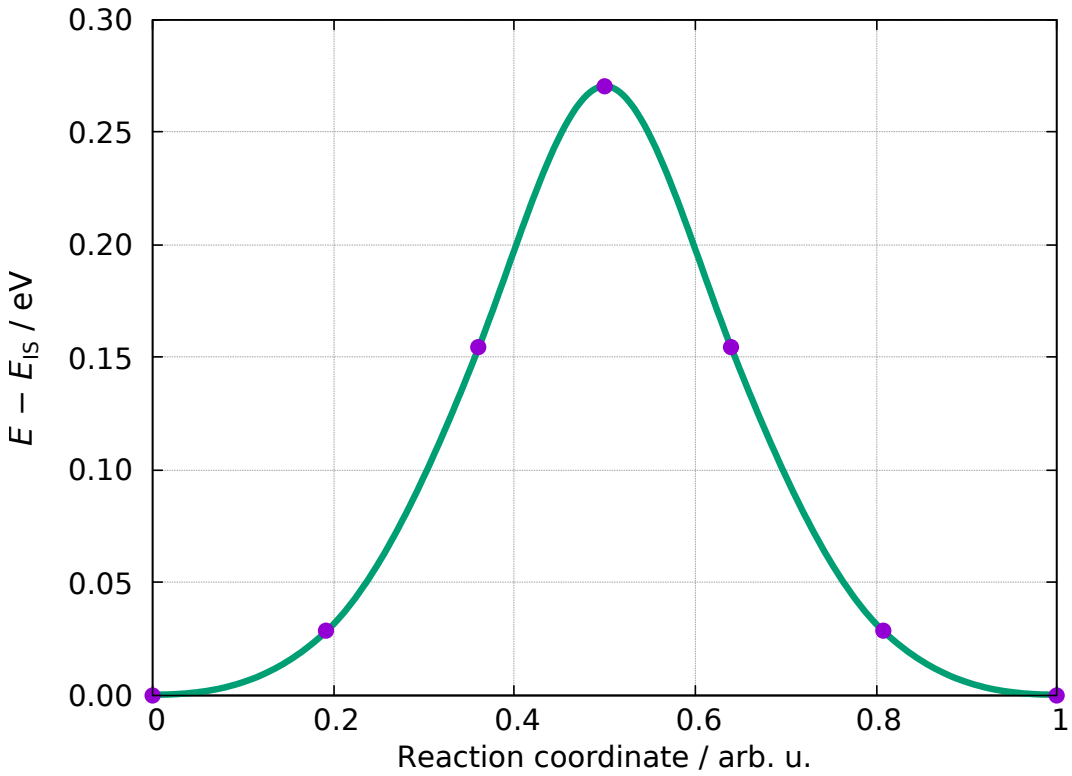
To run the example execute:

```
$ neb.x -inp neb.H2+H.in > neb.H2+H.out &
```

When the calculation finishes analyze the output (`neb.H2+H.out`). Pay attention to the *number of steps* required to reach convergence and the reaction energy barrier.

```
$ gnuplot H2+H.gp
```

The resulting plot should look something like this.



You can visualize individual points on the reaction path by using xcrysden.

Either by typing:

```
$ xcrysden --xyz H2+H.xyz
```

or

```
$ xcrysden --axsf H2+H.axsf
```

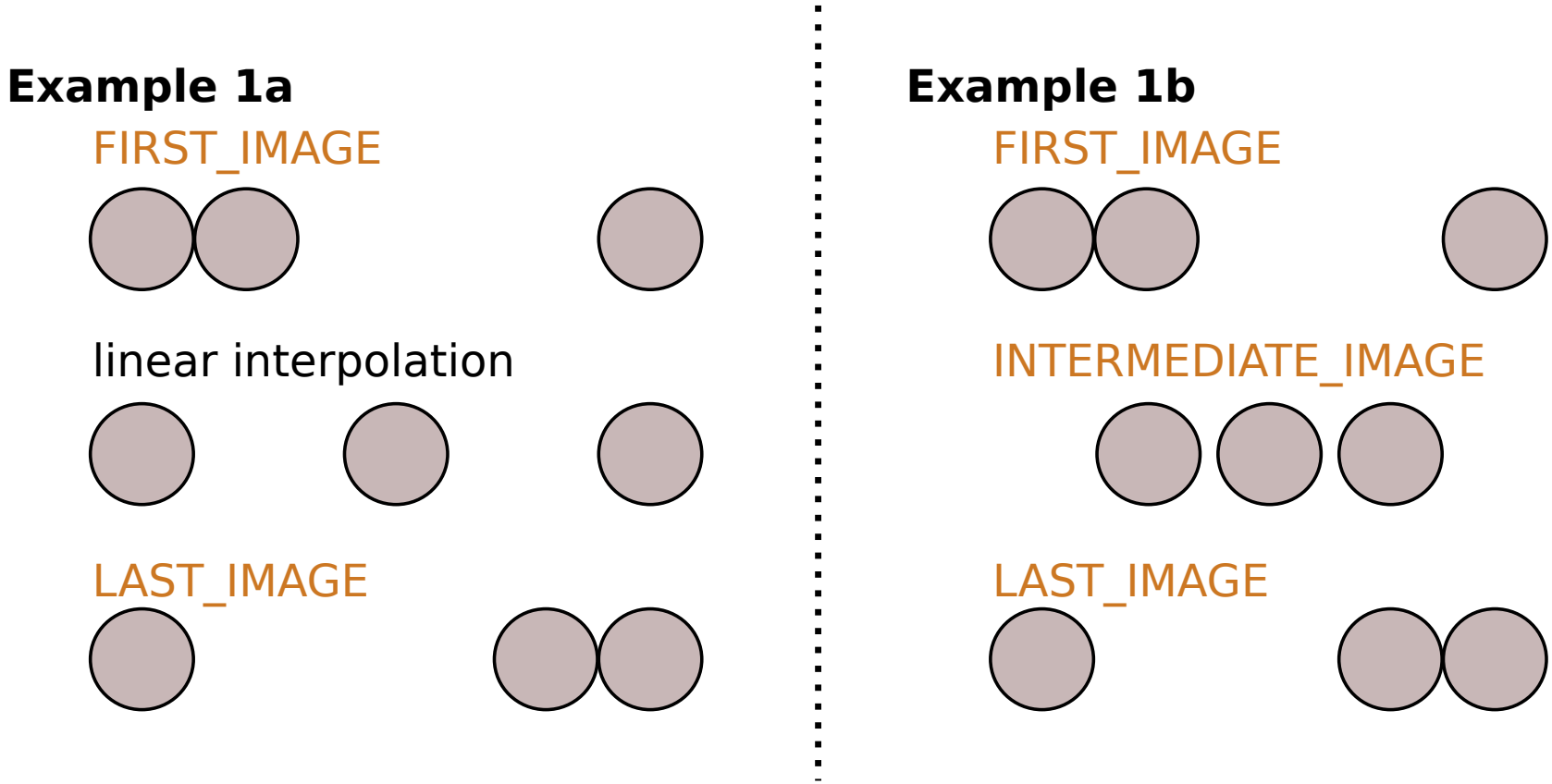
Example 1b: NEB with intermediate image

This example is similar as the previous example, however now chemical intuition is used to construct a better initial reaction path. This is achieved by using the INTERMEDIATE_IMAGE card, in which a rough guess for the transition state is provided. You can visualize the path by using xcrysden:

```
$ xcrysden --pwi neb.H2+H.w-inter-image.in
```

Try to spot the difference between the two examples. Notice that without the intermediate image, the three atoms are evenly spaced in the middle of the initial interpolated reaction path, which is not a good guess of the transition-state. If we use INTERMEDIATE_IMAGE then the three H atoms can be placed closer together, which

is a lot closer to the actual transition state and the NEB calculation does not require as many steps to converge. This is shown schematically below:



To run this example type:

```
$ neb.x -inp neb.H2+H.w-inter-image.in > neb.H2+H.w-inter-image.out &
```

The final reaction barrier should be the same as in the previous calculation, however convergence should require fewer steps.

Example 2: A more complex NEB calculation

In the second example we will study H_2 dissociation on the Al(100) surface. We will be using `INTERMEDIATE_IMAGE` and the calculation itself is also divided into two parts, in order to achieve convergence more quickly.

We will be using climbing-image NEB (CI-NEB), however initially we do not know which image is the *climbing-image* (the one with the highest energy), because its ID can change from iteration to iteration, depending on how far from convergence we are.

Thus we will first perform a plain NEB (`CI_scheme = 'no-CI'`) in order to stabilize the pathway and only once it is stabilized we will switch on the CI-NEB with `CI_scheme = 'auto'`. We will use a PWTK script to manage these two calculations.

Steps:

- Read `neb.H2-diss.Al100-2x1-2L.in` and try to understand it. This file will serve as the main input for the `pwtk` script. Visualize the initial guess by :

  ```
  $ xcrysden --pwi neb.H2-diss.Al100-2x1-2L.in
  ```

- Read the `neb.pwtk` script and try to understand it.

- Beware that this example takes about 20–30 min on a laptop. Hence, run the calculation remotely, i.e., submit it to the HPC cluster, i.e.:

  ```
  $ remote_pwtk neb.pwtk
  ```

- To download the calculated output files, use:

  ```
  $ rsync_from_hpc *.out
  ```

  Give the remote computer some time to make the calculation. To download other data files that were produced by `neb.x`, use:

  ```
  $ rsync_from_hpc .
  ```

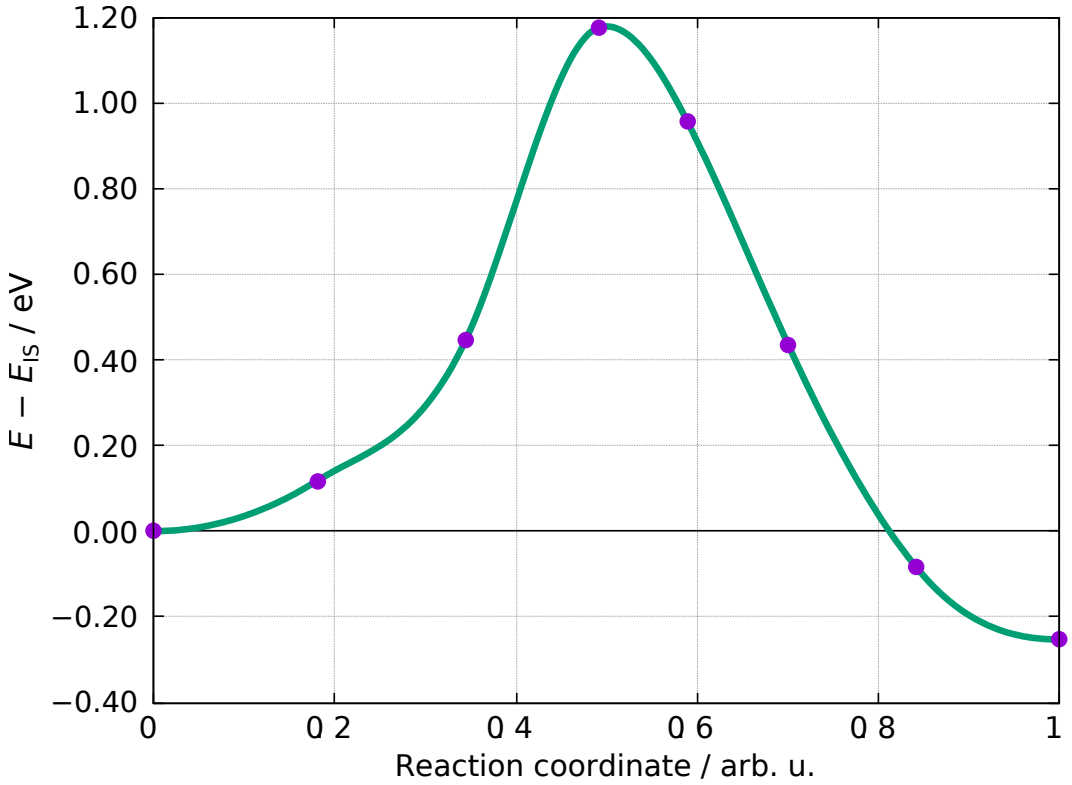
- Once the calculation converges analyze `neb.noCI.out` and `neb.auto.out` and check the number of steps and the reaction energy barrier. The reaction energy graph can be plotted by:

  ```
  $ gnuplot H2-diss.Al100-2x1-2L.gp
  ```

  and the corresponding path visulized by:

  ```
  $ xcrysden --axsf H2-diss.Al100-2x1-2L.axsf
  ```

The resulting PES along the reaction coordinate should look like this.



4. PWTK: a short tutorial

PWTK is a Tcl-based scripting interface for Quantum ESPRESSO. It aims at providing a flexible and productive framework.

- PWTK web-site is:
  `http://pwtk.quantum-espresso.org/` or `http://pwtk.ijs.si/`

- PWTK documentation is available at:
  `http://pwtk.ijs.si/toc_index.html`
  (see also `http://pwtk.ijs.si/pwtk-slides.pdf`)

- Move to `Day-3/example4.pwtk` directory. Therein are three examples:
  - `ex1.eos/` – how to use EOS (equation-of-state) utility of PWTK
  - `ex2.O@Al111/` – how to run many calculations with a simple PWTK script
  - `ex3.CO@Rh100/` – a more elaborate PWTK example that shows how to glue together various calculations. In particular, it analyzes the bonding of CO molecule on Rh(100).

5. PWTK: hierarchical configuration

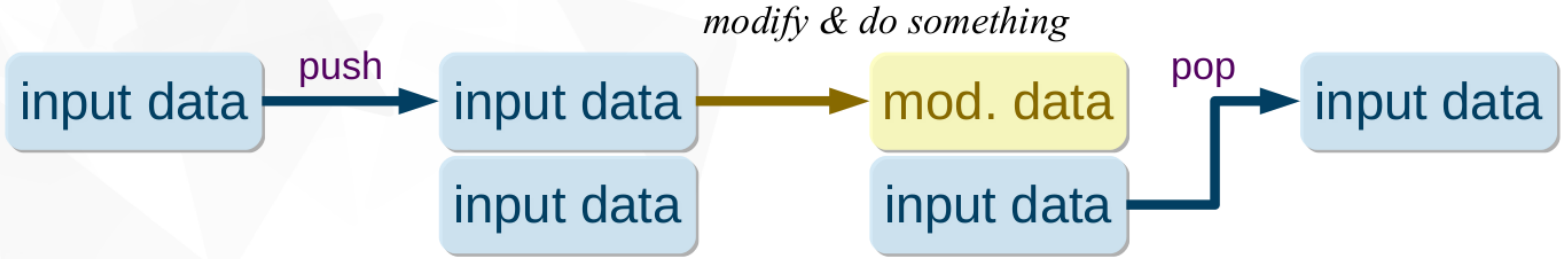
- Main user configuration file: `$HOME/.pwtk/pwtk.tcl`
  - executables and how to run them (`bin_dir`, `prefix`, `postfix`)
  - special directories (`pseudo_dir`, `outdir`, `wfcdir`)

- Project-based configuration (via `import`), e.g,:
  - `Day-3/example4.pwtk/common.pwtk`

    Specification of default input data that are common to all examples in `example4.pwtk/`, e.g., cutoff energies, list of pseudopotentials, etc.
    - **Beware:** for each specific calculation PWTK filters atomic species and uses only those that are actually used. Due to that the index of species can change from case to case, hence for *ntyp*-type variables do not use numeric indices, but use atomic labels instead, e.g.:
      `starting_magnetization(1) = 1.0` (*not recommended*)
      `starting_magnetization(Fe) = 1.0` (*recommended*)
  - in a given example the project-based `common.pwtk` is then imported via, e.g., `import ../common.pwtk`

5. PWTK: concept of input data stacking

Input data stacking is a useful concept when performing multiple calculations. In such a case, one typically defines a set of default parameters for all calculations. However, each individual calculation may require some modification of the input data, yet it is inconvenient if such a change would affect the input data for other subsequent calculations. This inconvenience can be prevented with `input_push` and `input_pop` commands (actually it is more convenient to use `input_pushpop {...}` instead). The concept is illustrated below:




```
... define default input data (#0)

# 1st calculation

input_push
   # here input data is the same as in (#0)

   ... modify input data for this calculation (#1)
   ... perform 1st calculation
input_pop

# input data is again the same as in (#0)
```

5.1 PWTK: EOS utility

Move to `Day-3/example4.pwtk/ex1.eos` directory.

In this example the *equation-of-state (EOS)* will be calculated using `pwtk`. Run this calculation by typing:

```
$ pwtk eos.Rh-bulk.pwtk > eos.Rh-bulk.log &
```

This procedure performs 15 SCF calculations with different lattice parameters (using the initial guess as input) and then uses the results as input for `ev.x`, to calculate EOS in accordance with the four equations of state. The results are then summarized in `eos.Rh-bulk.RESULTS` (the `*.dat` data-files with collected total energies are contained in `eos.Rh-bulk.d/*.dat`).

```
OPTIMISED LATTICE PARAMETER, a0 (in Bohr units)::

------------+------------+------------+------------+-----------
 # points   | MURNAGHAN  | BIRCH 0(1) | BIRCH 0(2) |    KEANE
------------+------------+------------+------------+-----------
 15:  1--15 |     7.294  |     7.293  |     7.293  |     7.294
 14:  1--14 |     7.294  |     7.293  |     7.293  |     7.294
 13:  2--14 |     7.293  |     7.293  |     7.293  |     7.294
 12:  2--13 |     7.294  |     7.293  |     7.293  |     7.294
 11:  3--13 |     7.293  |     7.293  |     7.293  |     7.293
 10:  3--12 |     7.293  |     7.293  |     7.293  |     7.293
  9:  4--12 |     7.293  |     7.293  |     7.293  |     7.297
  8:  4--11 |     7.293  |     7.293  |     7.293  |     7.296
  7:  5--11 |     7.293  |     7.293  |     7.293  |     7.296
------------+------------+------------+------------+-----------
```

Notice that the bulk-modulus is more sensitive to the sampling of data points, can you tell why?

OPTIMISED Total Energy, E0 (in Ryd units)::

# points	MURNAGHAN	BIRCH 0(1)	BIRCH 0(2)	KEANE
15: 1--15	-52.0363	-52.0364	-52.0364	-52.0363
14: 1--14	-52.0363	-52.0364	-52.0364	-52.0363
13: 2--14	-52.0364	-52.0364	-52.0364	-52.0363
12: 2--13	-52.0364	-52.0364	-52.0364	-52.0364
11: 3--13	-52.0364	-52.0364	-52.0364	-52.0364
10: 3--12	-52.0364	-52.0364	-52.0364	-52.0364
9: 4--12	-52.0364	-52.0364	-52.0364	-52.0364
8: 4--11	-52.0364	-52.0364	-52.0364	-52.0364
7: 5--11	-52.0364	-52.0364	-52.0364	-52.0364

OPTIMISED Bulk Modules, B0 (in kbar units)::

# points	MURNAGHAN	BIRCH 0(1)	BIRCH 0(2)	KEANE
15: 1--15	2550	2570	2591	1097
14: 1--14	2557	2571	2597	1178
13: 2--14	2557	2572	2609	1207
12: 2--13	2568	2578	2616	1176
11: 3--13	2569	2581	2604	1244
10: 3--12	2587	2594	2593	1170
9: 4--12	2587	2594	2606	829
8: 4--11	2602	2606	2602	703
7: 5--11	2603	2606	2620	638

5.2 PWTK: gluing together many calculations

The subject of this example is the calculation of a 2D potential energy scan of lateral positions of O @ Al(111). It introduces the PWTK's concept of input-data stacking (i.e. `input_pushpop {...}`).

The corresponding PWTK snippet is:

```
foreach ia [seq 0.0 0.2 1.0] {
   foreach ib [seq 0.0 0.2 1.0] {

      set x [expr $ia - 0.5*$ib]
      set y [expr sqrt(3)/2*$ib]

      input_pushpop {
         insertAtoms begin "O  $x $y 1.35   0 0 1"
         runPW pw.O-Al111.$ia.$ib.in
      }
   }
}
```

Beware that without the use of input-data stacking, each new iteration would add a new O atom to the structure. But with `input_pushpop {...}`, we add an O atom, make a calculation, and pop the O atom away.

The full script is provided by `scan.pwtk`; read it and and try to understand it.

Beware: this example takes quite long! Hence, it is better to submit it to HPC cluster. This can be achieved by:

```
$ remote_pwtk scan.pwtk
```

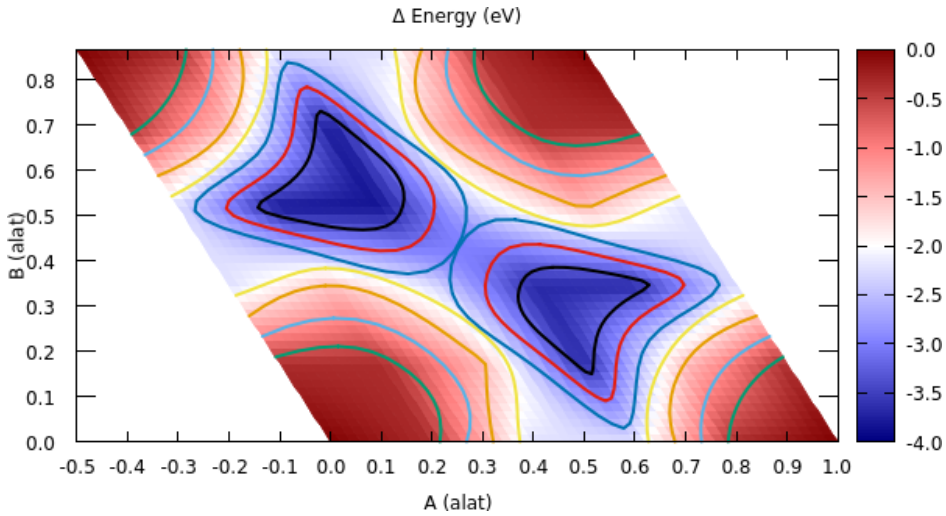
After calculation finishes, download the output and data files as:

```
$ rsync_from_hpc .
```

The resulting potential-energy-surface can be visualized as:

```
$ gnuplot plot2D.gp
```

Then plot the result with gnuplot. You should obtain something like this.



5.3 PWTK: analysis of electronic structure of CO@Rh(100)

This is a bit more elaborate PWTK example that shows how to glue together various calculations. In particular, it analyzes the bonding of CO molecule on Rh(100) by means of (i) charge-density difference, (ii) PDOS to atomic orbitals, (iii) MOPDOS to molecular orbitals, and (iv) ILDOS (integrated-local density of states).

The analysis reveals the charge-donation from the CO σ HOMO orbital into metal states and back-donation of charge from metal states into the CO π^* LUMO orbital.

The master PWTK script that performs the whole analysis is `run-all.pwtk`. This file imports several script, each performing specific task(s), in particular:

- `relax.pwtk` – script for relaxing the CO@Rh(100) structure
- `difden.pwtk` – script for calculating charge-density difference
- `ildos.pwtk` – script for calculation of ILDOSes
- `pdos.pwtk` – script for calculating PDOS to atomic orbitals, MOPDOS to molecular-orbitals of CO, and plots of molecular-orbitals (ψ^2) of CO (BEWARE: this is a more elaborate script)

5.3.1 DIFDEN utility

PWTK has a *DIFDEN* utility to aid at calculating, e.g., charge-density difference:

$$\Delta\rho_{\mathrm{AB}}(\boldsymbol{r}) = \rho_{\mathrm{AB}}(\boldsymbol{r}) - \rho_{\mathrm{A}}(\boldsymbol{r}) - \rho_{\mathrm{B}}(\boldsymbol{r})$$

(*remark:* with *DIFDEN* you can calculate any difference, does not need to be density)

The above difference can be calculated using the following PWTK snippet (the CO@Rh(100) example):

```
DIFDEN {
    segment(1) = "all"
    weight(1)  = 1.0
    name(1)    = "all"

    segment(2) = "1 2"
    weight(2)  = -1.0
    name(2)    = "CO"

    segment(3) = "3-"
    weight(3)  = -1.0
    name(3)    = "Rh100"
}
difden_run difden.CO-Rh(100)
```

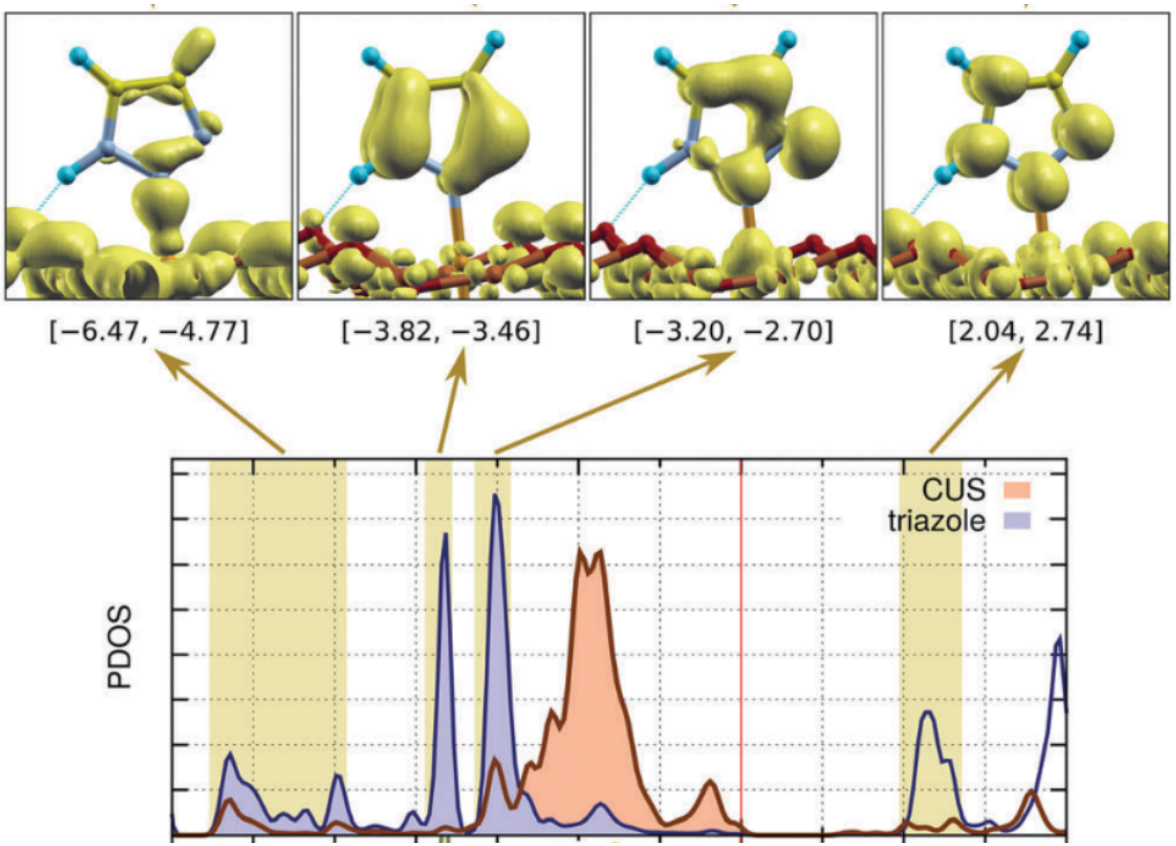
Note that the calculation of $\Delta\rho(\boldsymbol{r})$ requires 6 or 7 calculations, i.e., `pw.x` and `pp.x` calculations for each of AB, A, and B. The last (7^{th}) `pp.x` calculation is to subtract the calculated densities. PWTK manages all these 7 calculations automatically.

The full PWTK script for calculating the $\Delta\rho(\boldsymbol{r})$ of CO@Rh(100) is provided by file `difden.pwtk`.

5.3.2 ILDOS

ILDOS = integrated local density of states:

$$N(E_{\min}, E_{\max}, \boldsymbol{r}) = \int_{E_{\min}}^{E_{\max}} n(\epsilon, \boldsymbol{r}) d\epsilon = \sum_{v} \sum_{\boldsymbol{k}} \int_{E_{\min}}^{E_{\max}} |\psi_{v,\boldsymbol{k}}(\boldsymbol{r})|^2 \times \delta(\epsilon - \epsilon_{v,\boldsymbol{k}}) d\epsilon.$$



5.3.2 ILDOS

With aid of PWTK, a series of ILDOSes can be calculated using the following snippet:

```
INPUTPP " plot_num = 10 "

foreach {Emin Emax} {
    -6.00 -5.50
    -3.40 -3.30
    -3.05 -2.95
     ...   ...
} {
    INPUTPP " emin = $Emin, emax = $Emax "
    PLOT " fileout = 'ildos.${Emin}.${Emax}.xsf' "

    runPP pp.ildos.${Emin}.${Emax}.in
}
```

The full PWTK script for calculating a series of ILDOSes of CO@Rh(100) is provided by file `ildos.pwtk`.

5.3.3 PDOS and MOPDOS

The PWTK script `pdos.pwtk` is a bit more lengthy/elaborate. It calculates PDOS (projected-density-of-states) to atomic orbitals, MOPDOS (molecular-orbital projected-density-of-states) to molecular-orbitals of CO, and it plots molecular-orbitals $(\mathrm{sign}(\psi(\boldsymbol{r}))\psi^2(\boldsymbol{r}))$ of CO.

For further details, see `README.md` and the `pdos.pwtk` script itself. The purpose of this script is to demonstrate a bit more advanced usage of PWTK to those who may be interested.