Quantum ESPRESSO tutorial: Self-Consistent Calculations, Supercells, Structural Optimization

What can I learn in this tutorial?

1. How to run PWscf (pw.x) in self-consistent mode for Silicon

2. How to deal with metals (Aluminum)

3. How to deal with Ultrasoft pseudopotentials and with spin polarization (Iron)

4. How to set up a supercell and optimize a structure (Graphene)

More info can be found in

- on-line manuals at `www.quantum-espresso.org/resources/users-manual`

- Doc/ subdirectories in the QUANTUM ESPRESSO distribution

0. Getting example files

Download example file handson_pwscf.tgz and unpack it:

```
$ tar -xzvf handson_pwscf.tgz
```

($ is the prompt of your computer). This will create a sub-directory named Day1 containing several files.

Move to handson_pwscf/ and check its content:

```
$ cd handson_pwscf/
$ ls
Aluminum  handson_pwscf.pdf doc Graphene Iron  pseudo Silicon
```

The doc directory contains the input description files for the codes used in this tutorial. The pseudo directory contains pseudopotential files used in this tutorial. The handson_pwscf.pdf file contains these slides.

1. Getting started: Silicon

Self-consistent calculation for Silicon in the diamond structure:

- Move to the Silicon/ sub-directory

- Look at the input file si.scf.in. It is composed of three "namelists" &control (notice that calculation='scf' is the default value), &system, &electrons, followed by three "cards" ATOMIC_SPECIES, ATOMIC_POSITIONS, K_POINTS

- Write the appropriate values for the two variables

  - outdir: temporary directory for large files. Must be writable, will be created if not existent. You may set environment variable ESPRESSO_TMPDIR instead. outdir='../tmp' should be fine.
  - pseudo_dir: directory where pseudopotential (PP) files are kept. It must exist, be readable, and contain the required PP file (in this example, Si.pz-vbc.UPF for Silicon). You may set environment variable ESPRESSO_PSEUDO instead. pseudo_dir='../pseudo' should be fine

Providing atomic structure in input

How is the crystal structure defined? This is a very simple case: Diamond lattice is a fcc (face-centered cubic) lattice with two atoms per unit cell. You need to specify:

Diamond Structure   ZincBlend Structure
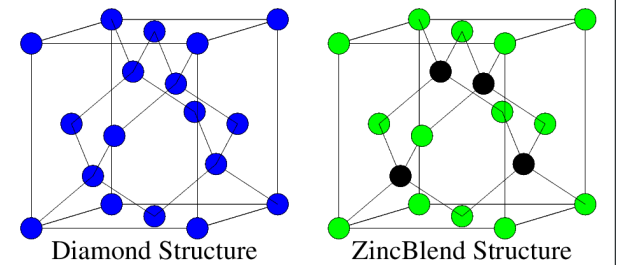
– What is the Bravais lattice
  `ibrav=2`, meaning fcc lattice
– How many and which parameters are needed to completely define Bravais lattice geometry
  just one: `celldm(1)=10.2`, lattice parameter a in a.u.
– How many atoms there are in the unit cell
  `nat=2`: two atoms
– How many different atomic species are present
  `ntyp=1`: one species
– Which ones, described by which pseudopotential
  See card `ATOMIC_SPECIES`
– Where the atoms are located in the unit cell
  See card `ATOMIC_POSITIONS`: here, in Cartesian axis, in units of a ("alat")

Notice that there are several alternative methods to specify an atomic structure!

Brillouin zone (BZ) sampling

BZ sampling is performed using the "2 Chadi-Cohen special points" (D.J. Chadi and M.L. Cohen, *Phys. Rev. B* **8**, 5747 (1973)) for the fcc lattice.

k-points are described in card `K_POINTS`. One has to choose

- Whether to provide a list of k-points, or a uniform grid
  In this case: a list in Cartesian axis in units of $2\pi/a$ ("tpiba")

- If a list is chosen: list of k-points *in the Irreducible BZ* and corresponding symmetry weights; the latter do not need to add up to 1, they are normalized by the code
  Frequently Asked Question: where do I find special k-points and their weights?
  Answer: 1) in papers, 2) use auxiliary code `kpoints.x`, 3) use uniform grids

- If a uniform grid is chosen: Monkhorst-Pack parameters (H.J. Monkhorst and J.D. Pack, *Phys. Rev. B* **13**, 5188 (1976)), and offsets along the three directions

Running the pw.x code

For serial (single processors) execution you can use

`$ pw.x -in si.scf.in > si.scf.out`

(note: input redirection `pw.x < si.scf.in` works but it is not recommended on parallel machines)

Look at the scratch directory `outdir` and its content:

```
$ ls my_outdir_directory
silicon.save   silicon.wfc1
```

The scratch directory contains temporary files used during the calculation as well as the final data directory for further processing. The name of the files is determined by the value of the `prefix` variable, by their content, number of processors, options, ...

Do not run two instances of pw.x that access the same `outdir` with the same `prefix`! Unpredictable behavior may follow. In case of trouble, clean the scratch directory.

Running the pw.x code II

Examine output file `si.scf.out`, look how self-consistency proceeds:

```
$ grep -e 'total energy' -e estimated si.scf.out
     total energy              =       -15.79103344 Ry
     estimated scf accuracy    <         0.06376674 Ry
     total energy              =       -15.79409289 Ry
     estimated scf accuracy    <         0.00230109 Ry
     total energy              =       -15.79447822 Ry
     estimated scf accuracy    <         0.00006291 Ry
     total energy              =       -15.79449510 Ry
     estimated scf accuracy    <         0.00000448 Ry
!    total energy              =       -15.79449593 Ry
     estimated scf accuracy    <         0.00000005 Ry
     The total energy is the sum of the following terms:
```

Notice that there are 8 electrons in the cell: 2 (pseudo-)atoms/cell with 4 electrons. The system is a non-magnetic insulator, so just the lowest 4=8/2 valence bands (Kohn-Sham states) are computed.

Convergence w.r.t. kinetic energy cutoff

The kinetic energy cutoff `ecutwfc` (in Ry) determines the size of the Plane-Wave (PW) basis set used to expand wavefunctions (i.e. Kohn-Sham orbitals)

(for Norm-Conserving PP, the PW set for charge density `ecutrho=4*ecutwfc`, do not specify it)

1. Change value of `ecutwfc` in si.scf.in input file to 16, 20, 24, 28, 32 Ry

2. Run `pw.x` again and again, noting the final energy;

3. Complete the data in file `si.etot_vs_ecut`;

4. Plot file `si.etot_vs_ecut` using your preferred plotting program, for instance:

   ```
   $ gnuplot
   gnuplot > plot 'si.etot_vs_ecut' using 1:2 with lines
   ```

   You should get a plot like the one in next page. Notice the monotonic convergence, as a consequence of the variational principle

Convergence w.r.t. kinetic energy cutoff II
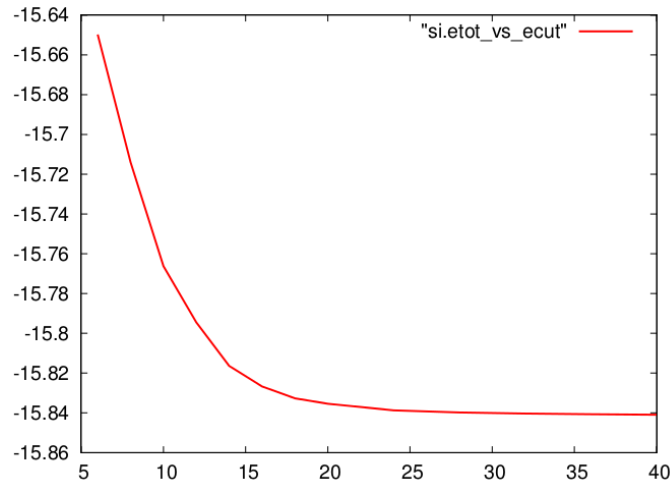
Reminder:

- Convergence w.r.t to cutoff is a property of the *pseudopotential(s) used*.

- Convergence of *absolute energy* is typically slower than convergence on *interesting physical properties*, e.g. structure.

- Absolute values of total energy do not have any physical meaning (and depend upon the specific PP): only energy *differences* do

Convergence w.r.t. k-points

A sufficiently dense grid of k-points is needed in order to account for *periodicity*

1. Edit si.scf.in (set `ecutwfc` back to 12 Ry), modifying the `K_POINTS` card to use *automatic* Monkhorst-Pack grids:

   ```
   K_POINTS automatic
   nk1 nk2 nk3 k1 k2 k3
   ```

   with `nk1=nk2=nk3=2,4,6` (increasing number of k-points), `k1=k2=k3=1`

2. Run `pw.x`, complete entries in file `si.etot_vs_nks` (nks is the actual number of k-points in the Irreducible BZ used in the calculation, reprinted on output)

3. Plot column 3 vs column 1, for instance using the following syntax:

   ```
   $ gnuplot
   gnuplot > plot 'si.etot_vs_nks' u 1:3 with lines
   ```

   You should get a plot like the one in next page.

Convergence w.r.t. k-points II
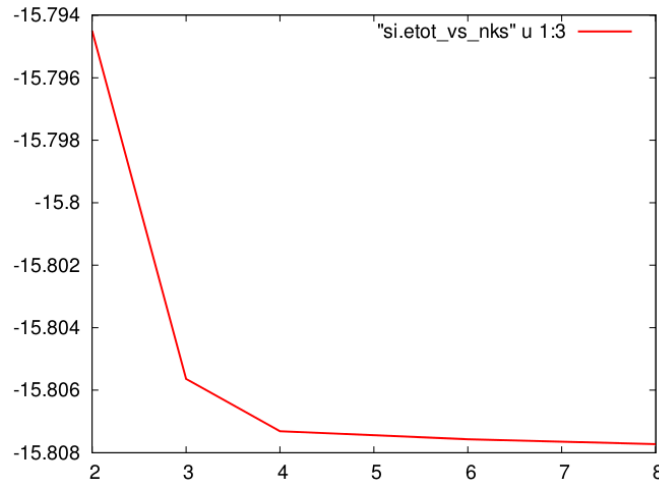
Reminder:

- The first three "nk1 nk2 nk3" numbers mean "there are nk1,nk2,nk3 grid points along crystal axis 1,2,3"; the second three "k1 k2 k3" numbers, either 0 or 1, mean "grid starts from 0" or "displaced by half a step" along crystal axis 1,2,3

  Also note that:

  - Convergence is not necessarily monotonic: there is no variational principle w.r.t. k-point number
  - The "2 2 2 1 1 1" Monkhorst-Pack grid is the same as the "two Chadi-Cohen points"

Equation of State of Silicon

Equilibrium in Si is determined by the minimum-energy lattice parameter alone: there are no forces on atoms, by symmetry (please verify this by setting `tprnfor=.true.` in namelist `&control` and looking for forces reprinted at the end)

- Choose suitable values for `ecutwfc` and the k-point grid (e.g.: 20 Ry, 4 4 4 1 1 1)

- Run the code for values of `celldm(1)` ranging from 9.8 a.u to 10.7 a.u in steps of 0.1 a.u.; to extract the final energy, use command "`grep !` *output-file*"

- Collect the results into a file `si.etot_vs_alat` as a sequence of rows a_i, $E(a_i)$

- Plot content of `si.etot_vs_alat` as in previous examples

Lazy (or maybe efficient?) people may edit script `run_si_eos`, defining variables `espresso_dir`, `pseudo_dir` and `outdir`, and run it as

`$ sh run_si_eos`

File `si.etot_vs_alat` is created at the end of the script file

Equation of State of Silicon II

The experimental lattice parameter for Si is 5.47 Å, or 10.26 a.u.: this is a case where plain simple LDA yields remarkable results
You may experiment changing cutoff, k-points, pseudopotential, ... You should find that
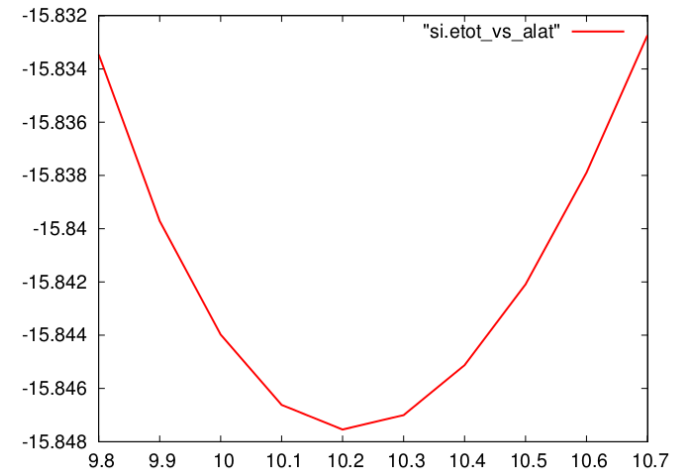
- The energy vs lattice parameter $E(a)$ curves are shifted down rather uniformly with increasing cutoff and are not strongly dependent on k-points.

- Structural properties and energy differences converge faster that total energies.

Use code `ev.x` to fit your results to a phenomenological EOS (e.g. Murnaghan) and to get accurate values for the lattice parameter and for the bulk modulus. The code prompts for some data, reads a file like `si.etot_vs_alat`: for cubic systems, rows
$a_i \qquad E(a_i)$

Band Structure of Silicon

Re-run the self-consistent calculation at equilibrium lattice parameter, then run a *non-self-consistent* (fixed-potential) calculation, with the same input as for scf, but

- variable `calculation` is set to 'bands' ;

- the number `nbnd` of Kohn-Sham states must be explicitly set;

- k-points are chosen along suitable high-symmetry lines. See sample file `si.bands.in`, containing the $L - \Gamma - X - K - \Gamma$ path.
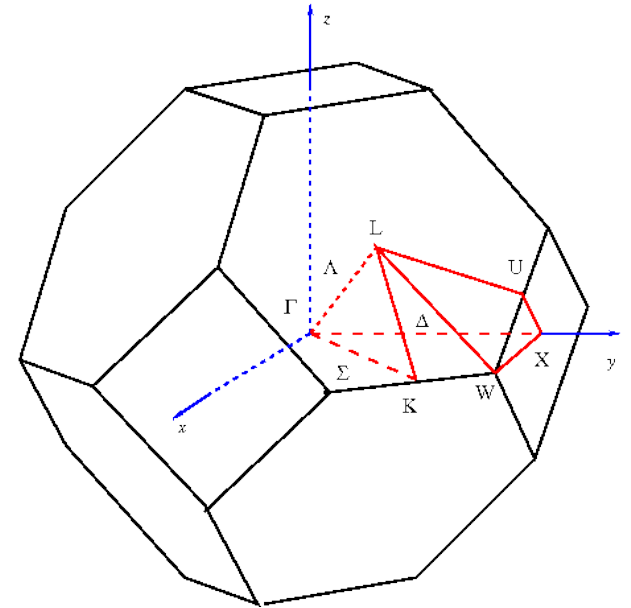
Important: `outdir` and `prefix` must be the same in bands and in scf calculations.
Important: the k-point path must be continuous in k-space

$ `pw.x -in si.bands.in > si.bands.out`

The list of k-points and of Kohn-Sham energies can be found after the line

      `End of band structure calculation`

Plotting Band Structure

There are various ways to plot the band structure. The simplest: use command `bands.x` with the following input (`outdir` and `prefix` as in previous steps):

```
&bands
  prefix='...', outdir='...', filband = 'sibands.dat', lsym=.true.
/
```

Two files are produced: `sibands.dat.gnu`, directly plottable with gnuplot, and `sibands.dat`, for further processing by auxiliary command `plotband.x`.

If option `lsym=.true`, bands.x performs a symmetry analysis. An additional file `sibands.dat.rep` is generated, containing information on symmetry labels of the various bands.

If option `lsym=.true`, bands.x does not perform the symmetry analysis.

Plotting Band Structure II

`plotband.x` prompts for terminal input:

```
$ plotband.x
Input file > sibands.dat
Reading     8 bands at     36 k-points
Range:    -6.3010    13.7920eV  Emin, Emax > -6.30 10.30
high-symmetry point:  0.5000 0.5000 0.5000    x coordinate    0.0000
high-symmetry point:  0.0000 0.0000 0.0000    x coordinate    0.8660
high-symmetry point:  0.0000 0.0000 1.0000    x coordinate    1.8660
high-symmetry point:  0.0000 0.5000 1.0000    x coordinate    2.3660
high-symmetry point:  0.0000 1.0000 1.0000    x coordinate    2.8660
high-symmetry point:  0.0000 0.0000 0.0000    x coordinate    4.2802
output file (gnuplot/xmgr) > sibands.plot
bands in gnuplot/xmgr format written to file sibands.plot
output file (ps) > (press Return)
```

If symmetry analysis was performed in the previous step, the output is written to several plottable files `sibands.plot.`N`.`M, where N labels the high-symmetry lines, M labels irreducible representations.

2. A metallic example: Aluminum

Aluminum is even simpler than Silicon: one atom per unit cell in a fcc lattice. BUT: it is a metal, valence bands only and a few k-points will not suffice.

- Move to the `Aluminum` sub-directory

- Edit file `al.scf.in`, define suitable values for `outdir` and `pseudo_dir`;

- Notice the presence of new variables: `occupations`, `smearing`, `degauss`;

- Run `pw.x` in the following way:

  ```
  $ pw.x -in al.scf.in > al.scf.out
  ```

- Notice in output file that
  - the number of bands (Kohn-Sham states) is automatically set to a value larger than the number of electrons divided by 2
  - the Fermi energy is computed.

Convergence with respect to degauss/smearing

1. Now edit file `al.scf.in`, varying

   - variable `smearing`, possible values:
     'gauss' (g), 'marzari-vanderbilt' (m-v), 'methfessel-paxton' (m-p)
   - variable `degauss`, suggested values: 0.06, 0.07, 0.08, 0.09, 0.10 (in Ry)

   so that you can complete the entries in files `al.g.6`, `al.m-v.6`, `al.m-p.6`

2. Do the same with denser k-point grid, e.g. replacing 6 6 6 in card `K_POINTS` with 8 8 8, 12 12 12, 16 16 16 (this step has already been done for you: the relative results are in files `al.g.8`, `al.g.12`, `al.g.16`, and similarly for m-p and m-v)

Plot the content of files `al.g.6`, `al.g.8`, `al.g.12`, `al.g.16`, and similarly for m-v and m-p. Notice how much slower the convergence is for metals than for insulators!

Both m-v and m-p are much less dependent upon degauss and allow for faster and safer convergence than simple gaussian broadening. For Al and m-v or m-p smearing, good convergence is achieved for a 12 12 12 grid and degauss $\sim 0.01 \div 0.05$ Ry.

Convergence with respect to degauss/smearing II
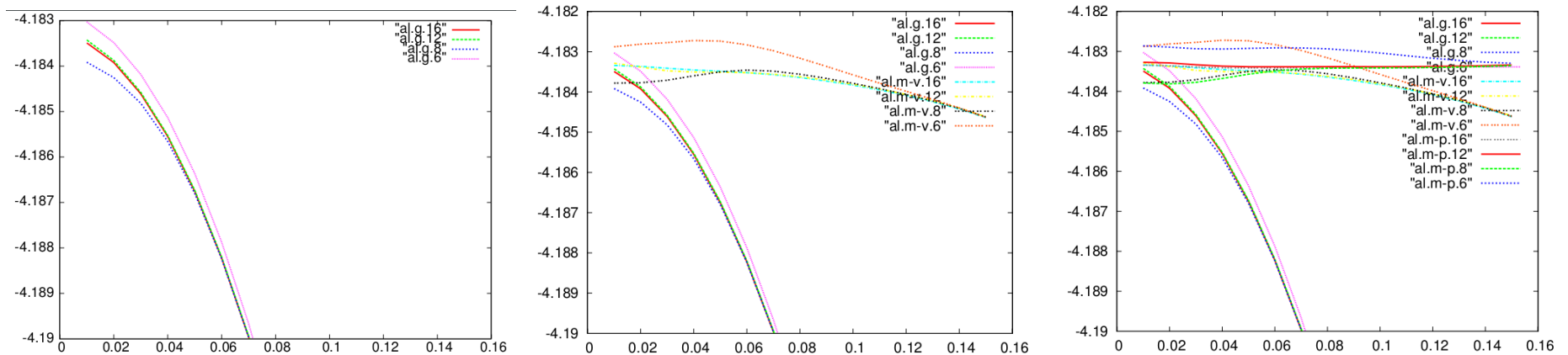
Notice that quantities like the force on an atom may also be used instead of, and be better suited than, the total energy in this kind of testing.

Also notice that you cannot reduce the broadening too much: the energy levels must have some overlap, or else the advantage of broadening is lost!

You may take inspiration from script `run_al_test` for further testing

3. A magnetic example: Iron

Iron has two remarkable features: it is magnetic, and it requires Ultrasoft PP (USPP) since its localized 3d atomic states are very hard.

- Move to the `Iron` sub-directory, edit file `fe_fm.scf.in`, defining suitable values for `outdir` and `pseudo_dir`;

- The structure is bcc (`ibrav=3`) with one atom per unit cell

- The number of Kohn-Sham states to be computed is explicitly indicated: `nbnd=8`

- Notice the presence of variables `nspin` and of `starting_magnetization`, indicating LSDA (`nspin=2`) with unconstrained total magnetization and initial symmetry broken; plus, variables for *metallic* calculations

- Notice that this calculation uses GGA (PBE): it is specified inside the PP file (can be guessed from the PP file name), reprinted on output as "Exchange-correlation"

- Also notice that with USPP, it is typically needed to set $\mathtt{ecutrho} > 4\times\mathtt{ecutwfc}$ (it should be at least 8 to 12 times larger)

Magnetic structures

Run `pw.x` in the usual way. In the output, notice:

- the number of k-points is doubled w.r.t the non-magnetic case: the first set of k-points contains spin-up states, the second set spin-down states (use `verbosity='high'` in namelist `&control` if there are more than 100 k-points)

- the lines

```
total magnetization      =      2.41 Bohr mag/cell
absolute magnetization   =      2.60 Bohr mag/cell
```

Since there is a single (magnetic) atom per unit cell, the only possible magnetic structure is ferromagnetic. In order to reach antiferromagnetic states, you need to

– Introduce a supercell with two sublattices of different species of atoms (even if they are the same, it is important that they are labeled as different)
– Start with opposite initial magnetization for the two sublattices

Can you write input data for an AFM structure? (hint: split bcc into two simple cubic sublattices, `ibrav=1`, with two atoms at (0,0,0) and (1/2,1,2,1/2))

Convergence check for USPP

For computational efficiency, it is convenient to keep `ecutwfc` as low as possible, while `ecutrho` is less critical (look at the CPU time report at the end of an output: there are very many `fftw`, depending upon `ecutwfc`, while a much smaller number of `fft` depends upon `ecutrho`)

Fix the `ecutrho`/`ecutwfc` ratio ("dual") to 4, 6, 8, 12 and compute the energy vs `ecutwfc` curve. For dual=4 it will look funny: energy *increases* with increasing cutoff, but for higher dual (i.e. better description of augmentation charge) the normal behavior is observed.
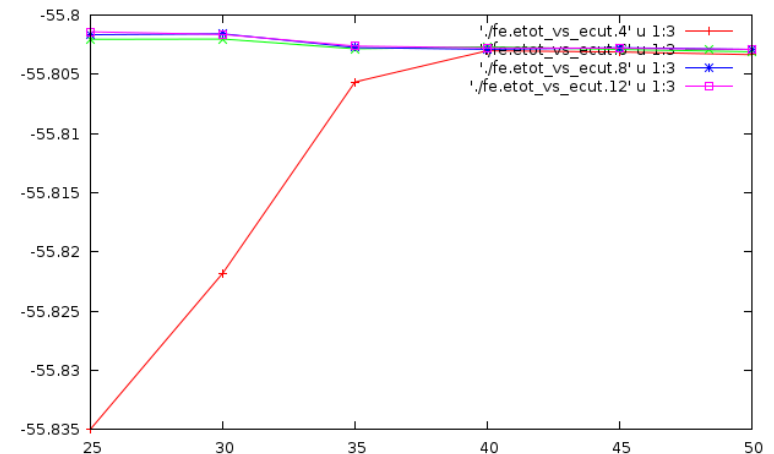
Once convergence is reached for both cutoffs and k-points, you may compare the stability of Iron in the bcc, fcc, hcp phases (the latter being a slightly more complicated structure)

Density of States (DOS)

1. Run a non self-consistent calculation (`calculation='nscf'` in namelist `&control`), using a previously computed scf potential. Input file `fe_fm.nscf.in`:

   - has the same `prefix` and `outdir` of the previous scf step
   - uses a denser k-point mesh than in the previous scf step
   - uses the *linear tetrahedron method* (variable `occupations='tetrahedra'`)

2. Run code `dos.x`, using input file `fe.dos.in`: $ dos.x -in fe.dos.in, where

   - `prefix` and `outdir` are the same of the previous non-scf step
   - output is written to file `fildos`

3. Plot spin-up (column 2) and down (column 3) DOS as a function of E (column 1)

   (step 1 is not strictly needed, but if you want a nice DOS you need tetrahedra and a dense k-point grid)
   Reminder: Absolute values of Kohn-Sham eigenvalues have no direct physical meaning
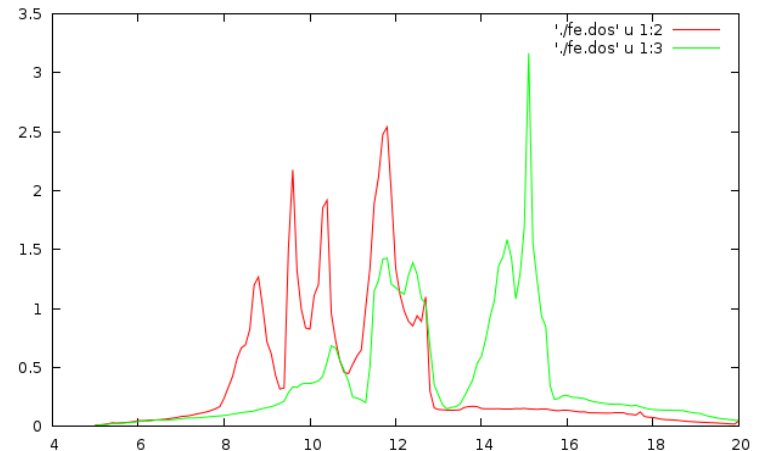
Projected Density of States (PDOS)

1. Run a non self-consistent calculation as for the DOS case

2. Run code `projwfc.x`, using input file `fe.pdos.in`:
   `$ projwfc.x -in fe.pdos.in`, where

   - gaussian broadening degauss is used (tetrahedra not yet implemented)
   - `prefix` and `outdir` are the same of the previous non-scf step
   - the prefix to all output files in in variable `filpdos`

3. Plot spin-up (column 2) and down (column 3) PDOS for s and d states: `fe.pdos.pdos_atm#1(Fe)_wfc#1(s)`, `fe.pdos.pdos_atm#1(Fe)_wfc#2(d)`, as a function of E (column 1), compared to the sum of all PDOS (columns 4 and 5 of `fe.pdos.pdos_tot`, total DOS in column 2 and 3: they aren't exactly the same, why?)
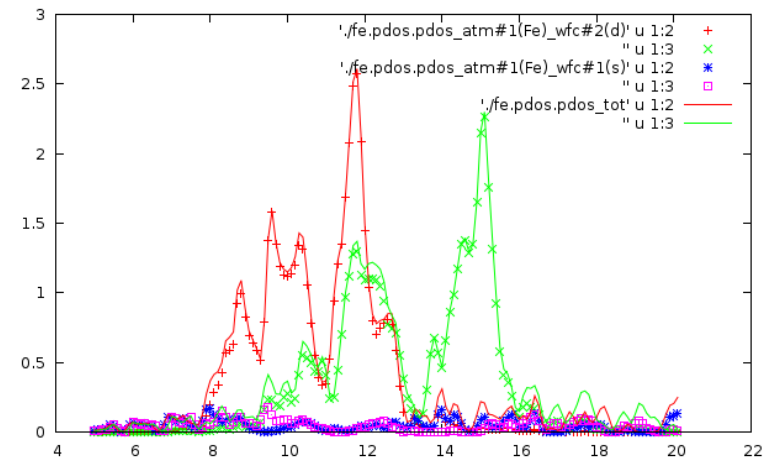
A slightly more complex structure: hcp Iron

Iron in the hcp structure has a hexagonal lattice with two parameters, a and c:

$$\mathbf{a_1} = (a, 0, 0), \quad \mathbf{a_2} = (-\frac{a}{2}, \frac{a\sqrt{3}}{2}, 0), \quad \mathbf{a_3} = (0, 0, c)$$

and two atoms in the unit cell, at positions

$$\mathbf{d_1} = (0, 0, 0), \qquad \mathbf{d_2} = (0, \frac{a\sqrt{3}}{3}, \frac{c}{2}).$$

The noncubic lattice can be described in many equivalent ways:
- `ibrav=4`, `celldm(1)=` a, `celldm(3)=` c/a, as in sample file `fe_hcp.scf.in`
- `ibrav=4`, `a=` a, `c=` c, *both in Å, not a.u.*
- `ibrav=0`, `celldm(1)=` a, lattice vectors in card

```
CELL_PARAMETERS (alat)
1.000000 0.000000 0.000000
-0.500000 0.866025 0.000000
0.000000 0.000000 1.633000
```
- Using `space_group=194`, and Wyckoff positions $2c$ (see `fe_hcp.wy.in`)

....

Useful hint: use `xcrysden --pwi fe_hcp.scf.in` to verify the structure.

Equation of state for noncubic structures

For a few selected values of the unit cell volume,

1. Perform a series of calculations with different c/a, locate minimum E; or

2. Perform a *variable-cell calculation* (sample file `fe_hcp.vcr.in`:
   - set `calculation='vc-relax'`
   - add two namelists, `&ions ... /` and `&cell ... /`, after `&electrons`
   - in `&cell ... /`, set `cell_dofree='shape'` (keeps volume fixed)

Inspect output of $ `pw.x -in fe_hcp.vcr.in > fe_hcp.vcr.out`:

- several scf steps are performed, forces (zero by symmetry) and stresses computed

- the energy decrease and the stress becomes more and more isotropic 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

4. A more complex example: Graphene-based materials

• Graphene has a 2-atom hexagonal unit cell in the xy plane: ibrav=4, celldm(1)=4.6542890, celldm(3)=some suitably large value, e.g. 3.0) (remember: celldm(1) in Bohr radii, celldm(3)=c/a; alternatively: A=2.463, C=7.389, in Å)
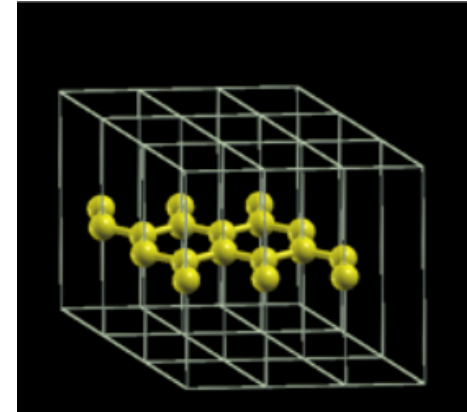
• Atomic positions:

ATOMIC_POSITIONS (alat)

C 0.000000 0.000000 0.000000

C 0.000000 0.5773503 0.000000

or, equivalently:

ATOMIC_POSITIONS (crystal)

C 0.000000 0.000000 0.000000

C 0.333333 0.6666666 0.000000

• K-points: use a dense grid in the xy plane only, e.g.

K_POINTS (automatic)

9 9 1 0 0 0
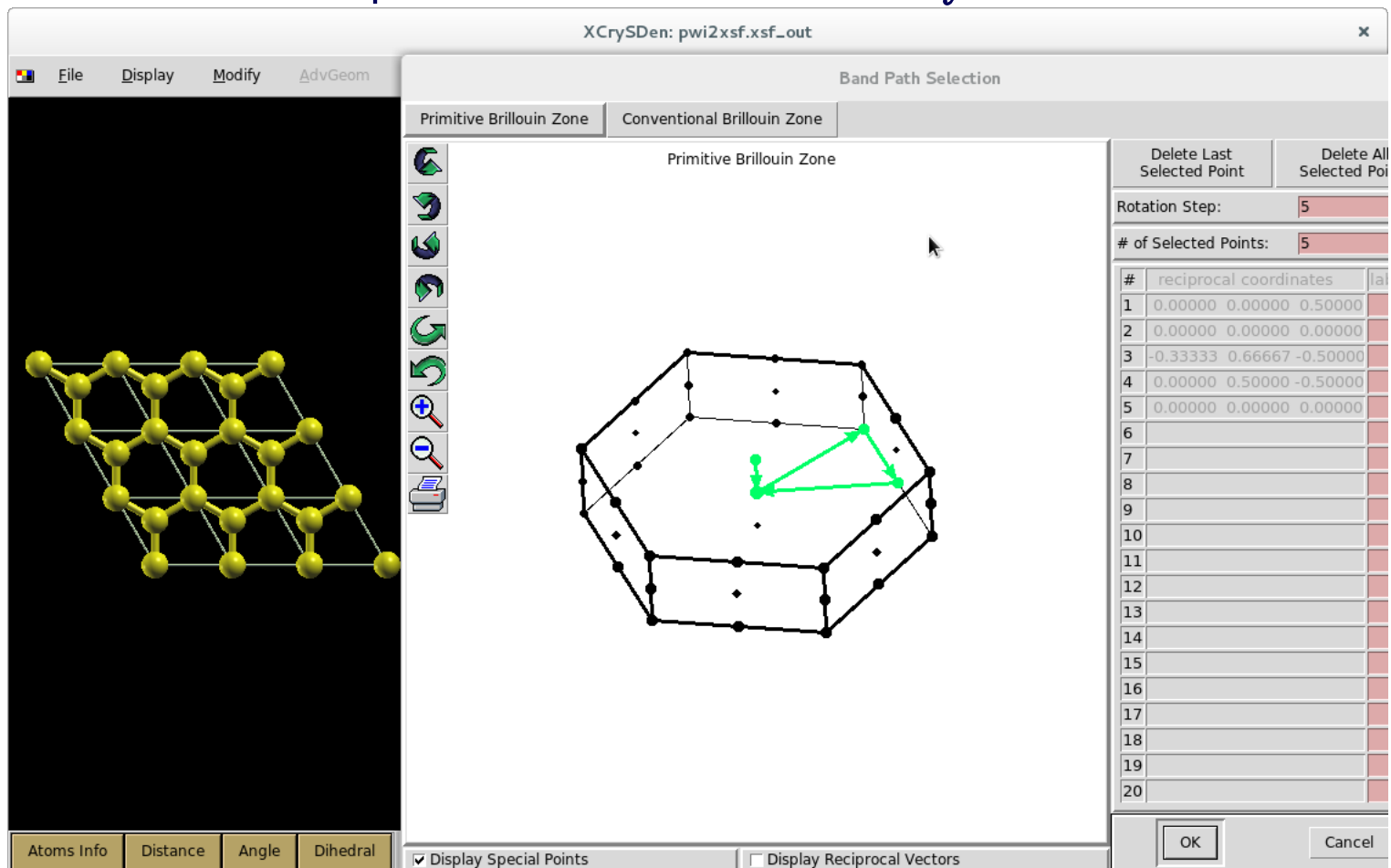
(a uniform 9×9×1 grid, centered on $\mathbf{k} = (0,0,0)$)

• Run the scf calculation for graphene, file graphene.in (or graphene-alt.in), analyze the output.

Bands of graphene

Can you reproduce the "cones" of the band structure at the Fermi surface? The most difficult (?) part is to figure out a suitable path of k-points.

You may either use the "k-path selection" tool of `xcrysden` ...

.... or the SeeK-path web site at http://materialscloud.org/tools/seekpath:
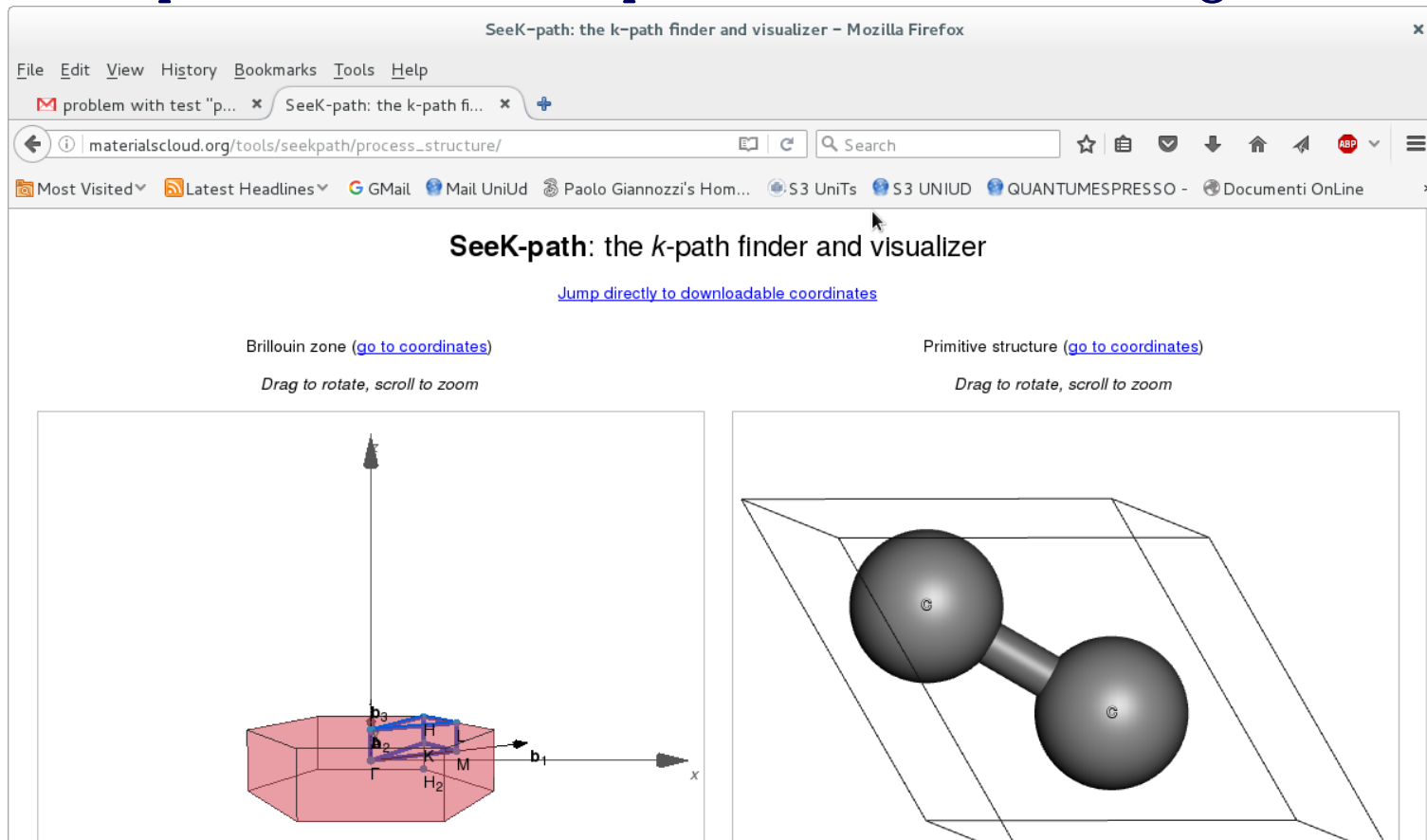
then proceed as for Si: scf calculation followed by a band calculation and `bands.x`, `plotband.x` postprocessing.

For DOS: scf calculation followed by a dense-grid calculation with tetrahedra, `dos.x` postprocessing.

Graphene (3×3 supercell)

• The previous calculation is periodic in the xy plane, has a slab geometry in the z direction, with a large vacuum space separating layers and a fictitious periodicity (no crystalline states). We want now to build a supercell in the xy plane.
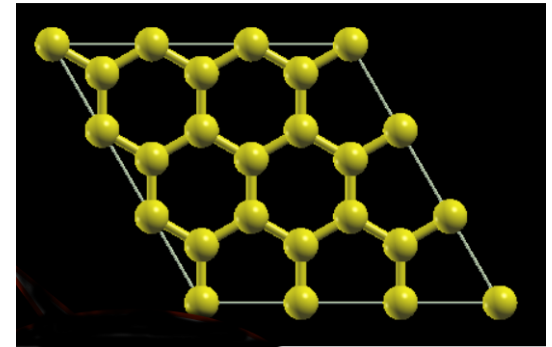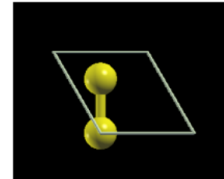
• Let us build a 3×3 supercell in the xy, file `graphene3x3.in`:

  – Primitive lattice vectors a and b are multiplied by 3: `celldm(1)=13.9628670`
  – Primitive lattice vector c remains the same, so: `celldm(3)=1.0`. If you want the same k-point grid, just use `K_POINTS (automatic)` with 3 3 1 0 0 0 grid
  – Reciprocal lattice vectors in the xy plane are divided by 3 (look at the output)
  – There are 9 times the atoms of the original unit cell (that is, 18 atoms)
  – There are ~ 9 times more Kohn-Sham energies per k-point

• Provided that the two k-point grids are equivalent (please check!):

  – The energy of the supercell $E^{SC} = 9E^{UC}$ almost exactly (UC=unit cell)
  – all $\epsilon^{SC}(\mathbf{k}_i)$ are (almost) equal to some $\epsilon^{UC}(\mathbf{k}_j)$ if $\mathbf{k}_j$ refolds into $\mathbf{k}_i$

Graphane (structural optimization)

• Graphane is like graphene, with a H atom bound to each C atom, on opposite sides. You need to optimize atomic positions, i.e. find the minimum-energy structure (zero forces).
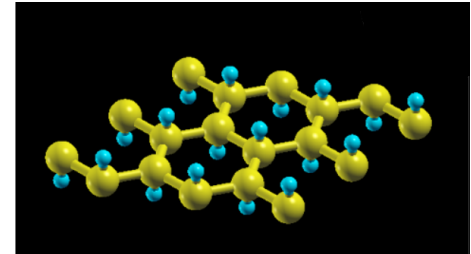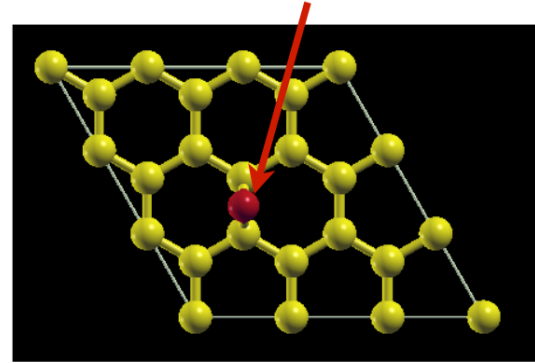

• File `graphane.in` is a modified version of `graphene.in`, with

   – `calculation='relax'` for structural optimization, new namelist `IONS`
   – `ntyp=2` (2 types of atoms), `nat=4` (4 atoms in the cell)
   – `ATOMIC_POSITION` card with 4 initial positions (C-H distance ~ 1 Å)
   – `ATOMIC_SPECIES` card with 2 species of atoms and pseudopotentials

• Run the structural optimization, analyze the output: it consists of several scf steps, followed by calculation of forces and generation of new positions

In the picture, the final structure of graphane, exhibiting "buckling"
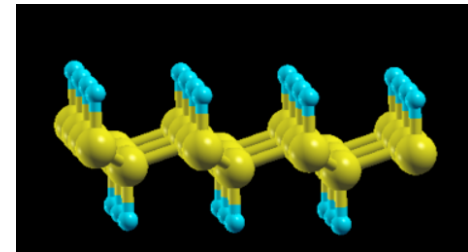
Graphene Oxide

• 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

• File `graphene3x3-O.in` is a modified version of `graphene3x3.in`, with

  – `calculation='relax'` for structural optimization, new namelist `IONS`
  – `ntyp=2` (2 types of atoms), `nat=19` (19 atoms in the cell)
  – `ATOMIC_POSITION` card with 19 initial positions (C-O distance ~ 1.5 Å)
  – `ATOMIC_SPECIES` card with 2 species of atoms and pseudopotentials

• Run the structural optimization, analyze the output