Workshop on High Performance Computing for Materials Characterization, Design and Discovery - SMR3966

Quantum ESPRESSO Hands-on session

(Basic SCF + post-processing + convergence test calculations)

Rafael González, Carlos Pinilla, Camilo Espejo

(Based on MaX School tutorial - 2021 by Pietro Delugas, et. al.)

ICTP-SMR3966: QE Hands-on session

Topics of Quantum ESPRESSO hands-on session:

1. How to run basic PWscf (`pw.x`) calculations
2. How to run post-processing calculations to plot molecular-orbitals and charge-density (`pp.x`), DOS (`dos.x`), and band-structure (`bands.x`)
3. How to calculate low-dimensional systems (`example1.benzene/` and `example2.graphene/`)
4. How to make basic convergence tests (`example3.Si/`)
5. How to deal with metals (`example4.Al/`)

About Quantum ESPRESSO

More info about Quantum ESPRESSO can be found in:

- https://www.quantum-espresso.org/
- Quantum ESPRESSO (QE) documentation:
  - on-line manuals at www.quantum-espresso.org/resources/users-manual
  - Doc/ sub-directories in the QUANTUM ESPRESSO distribution
  - input data description: most programs contained in QE have their own input file description in the form of hyperlinked **INPUT_***.html** files (where *** stands for the name of the program)

Hands-on material

QE Hands-on examples material is contained within its own directory:

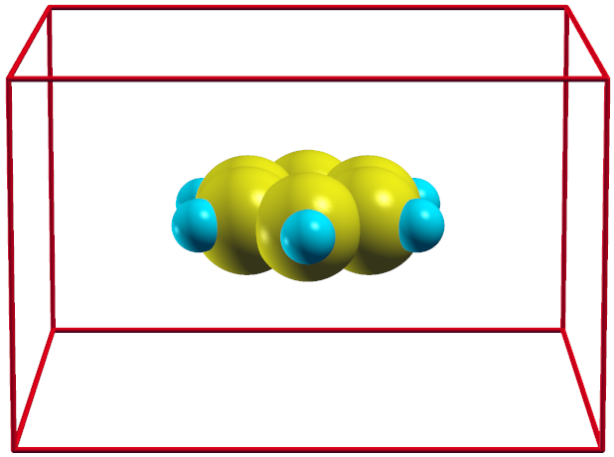
- `example1.benzene/` – hands-on exercise 1
- `example2.graphene/` ... `example5.Fe/`

- All directories contain a `README.md` file with instructions how to run exercise(s)
- Naming of files is described in `README-filenames.md`
- To help recognizing for which program a given input file is intended, the filename starts with the name of the program, i.e.:
  - `pw.*.in` – input file for `pw.x` program
  - `pp.*.in` – input file for `pp.x` program
  - etc.

Disclaimer: *many examples use lousy convergence thresholds to speed-up calculations*

1. How to describe a molecule with Quantum ESPRESSO

With Quantum ESPRESSO we can describe a molecule by putting it in a big box.



- move to `example1.benzene/` directory
- look at the input file `pw.benzene.scf.in`. It is composed of three "namelists" `&CONTROL` (note that `calculation ='scf'` is the default value), `&SYSTEM`, `&ELECTRONS`, followed by three "cards" `ATOMIC_SPECIES`, `ATOMIC_POSITIONS`, `K_POINTS`. In the terminal execute `more pw.benzene.scf.in`
- instructions for how to run the example are in `README.md`

Disclaimer: *the box used in this example is very small as to speed-up calculations*

1. How to calculate and plot molecular orbitals

Here are the two needed input files for calculation of signed molecular-orbital densities of benzene (i.e., $\mathrm{sign}(\psi_i(\boldsymbol{r}))|\psi_i(\boldsymbol{r})|^2$), opened with `emacs` using `QE-emacs-modes`:

```
emacs@cl
File  Edit  Options  Buffers  Tools  Help
 &CONTROL
    prefix='benzene',
 /
 &SYSTEM
    ibrav = 6
    A = 11.0
    C = 7.0

    nat =  12,
    ntyp =  2,
    nbnd = 16

    ecutwfc = 20.0,
    ecutrho = 200.0,
    assume_isolated  = 'martyna-tuckerman',
 /
 &ELECTRONS
 /
ATOMIC_SPECIES
   C   1.0   C.pbe-rrkjus.UPF
   H   1.0   H.pbe-rrkjus.UPF
ATOMIC_POSITIONS angstrom
   H   5.5000000   7.98563953   3.5
   C   5.5000000   6.89520922   3.5
   C   6.7089386   6.19812524   3.5
   H   7.6529918   6.74309454   3.5
   C   6.7089386   4.80187470   3.5
   H   7.6529918   4.25690561   3.5
   C   5.5000000   4.10479062   3.5
   H   5.5000000   3.01436043   3.5
   C   4.2910612   4.80187468   3.5
   H   3.3470081   4.25690556   3.5
   C   4.2910613   6.19812528   3.5
   H   3.3470082   6.74309458   3.5

K_POINTS gamma

-:---  pw.benzene.scf.in   All (18,2)     (QE-pw.x)
Beginning of buffer
```

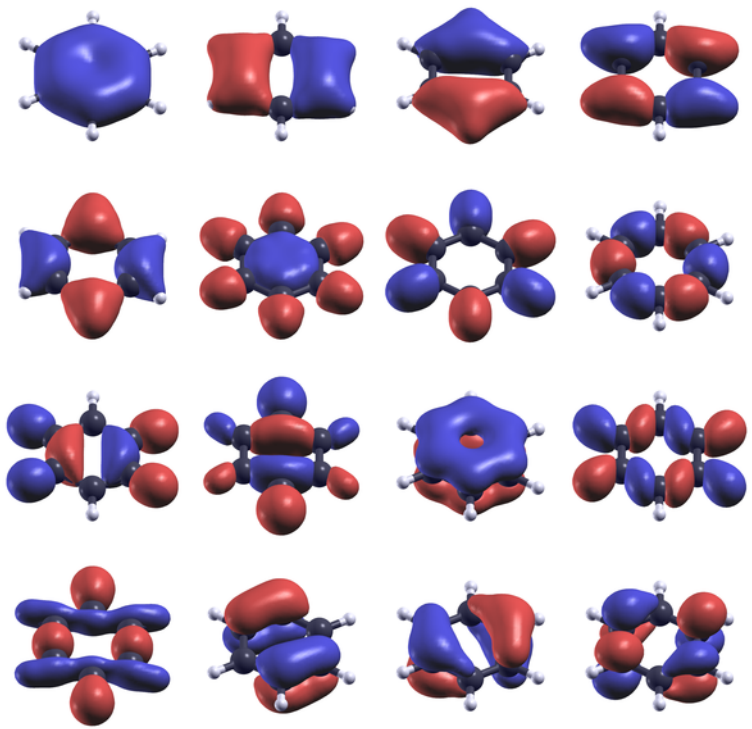
```
emacs@cl
File  Edit  Options  Buffers  Tools  Help
&INPUTPP
    prefix     =  'benzene',
    filplot    =  'psi2.benzene',

    plot_num   =  7,
    kpoint     =  1,
    kband(1)   =  1,
    kband(2)   =  16,
    lsign      =  .true.,
 /
 &PLOT
    fileout       =  '.xsf',
    iflag         =  3,
    nfile         =  1,
    output_format =  5,
    weight(1)     =  1.0,
 /

-:---  pp.benzene.psi2.in   All (1,0)     (QE-pp.x)
```

1. How to calculate and plot molecular orbitals

To plot signed molecular-orbital densities ($\mathrm{sign}(\psi_i(\boldsymbol{r}))|\psi_i(\boldsymbol{r})|^2$), we need to:

- calculate Kohn-Sham states with `pw.x` (i.e. make an SCF calculation)
- instruct `pp.x` to write them in a suitable format to specified files
- plot orbitals with `xcrysden`

See `README.md` for detailed instructions.



1. How to plot molecular orbitals with xcrysden

- Execute in the terminal:
  - `pw.x < pw.benzene.scf.in > pw.benzene.scf.out`
  - `pp.x < pp.benzene.scf.in > pp.benzene.scf.out`

  The resulting molecular orbitals (i.e., $\mathrm{sign}(\psi(\boldsymbol{r}))|\psi(\boldsymbol{r})|^2$) are written to `psi2.benzene_*.xsf`

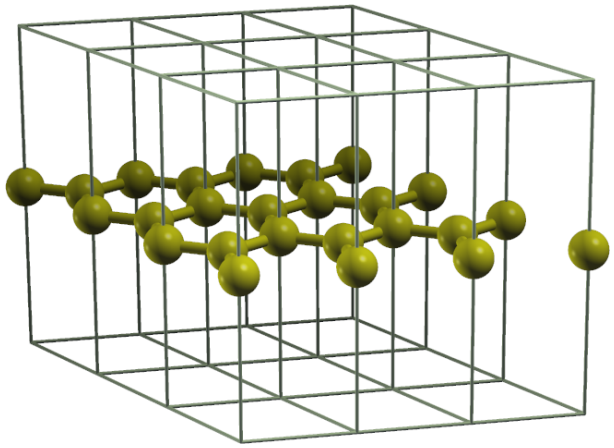
- Plot one of the generated XSF files with `xcrysden`, e.g.:
  - `xcrysden --xsf psi2.benzene_K001_B006.xsf`

  and follow these instructions:
  - use the menu `Tools-->Data Grid`; a new window opens, press `[OK]`
  - an isosurface-control window appear; specify the `Isovalue`, say `0.005` and press `[Submit]`
  - click the `Render +/- isovalue` radiobutton and again press `[Submit]`
  - rotate and zoom the structure according to your preference
  - save the displayed *state* via the menu `File-->Save Current State` (e.g., save to `my-display.xcrysden`)
  - try this with other orbitals, e.g.:
    - `xcrysden --xsf psi2.benzene_K001_B005.xsf --script my-display.xcrysden`

- To plot all orbitals, execute: `./plot-psi2.sh`

2. How to calculate a 2D-periodic system: graphene

A 2D-periodic system (e.g., a graphene sheet) is modelled by adding a vacuum layer in the 3rd direction.



- move to `example2.graphene/` directory

- look at the input file `pw.graphene.scf.in`; graphene has a 2-atom hexagonal unit cell in the xy plane: `ibrav=4`, `celldm(1)=4.654`, `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=7.388` 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 0.666667 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 $\mathbf{k} = (0, 0, 0)$)

2. Graphene: DOS and bands (spaghetti)

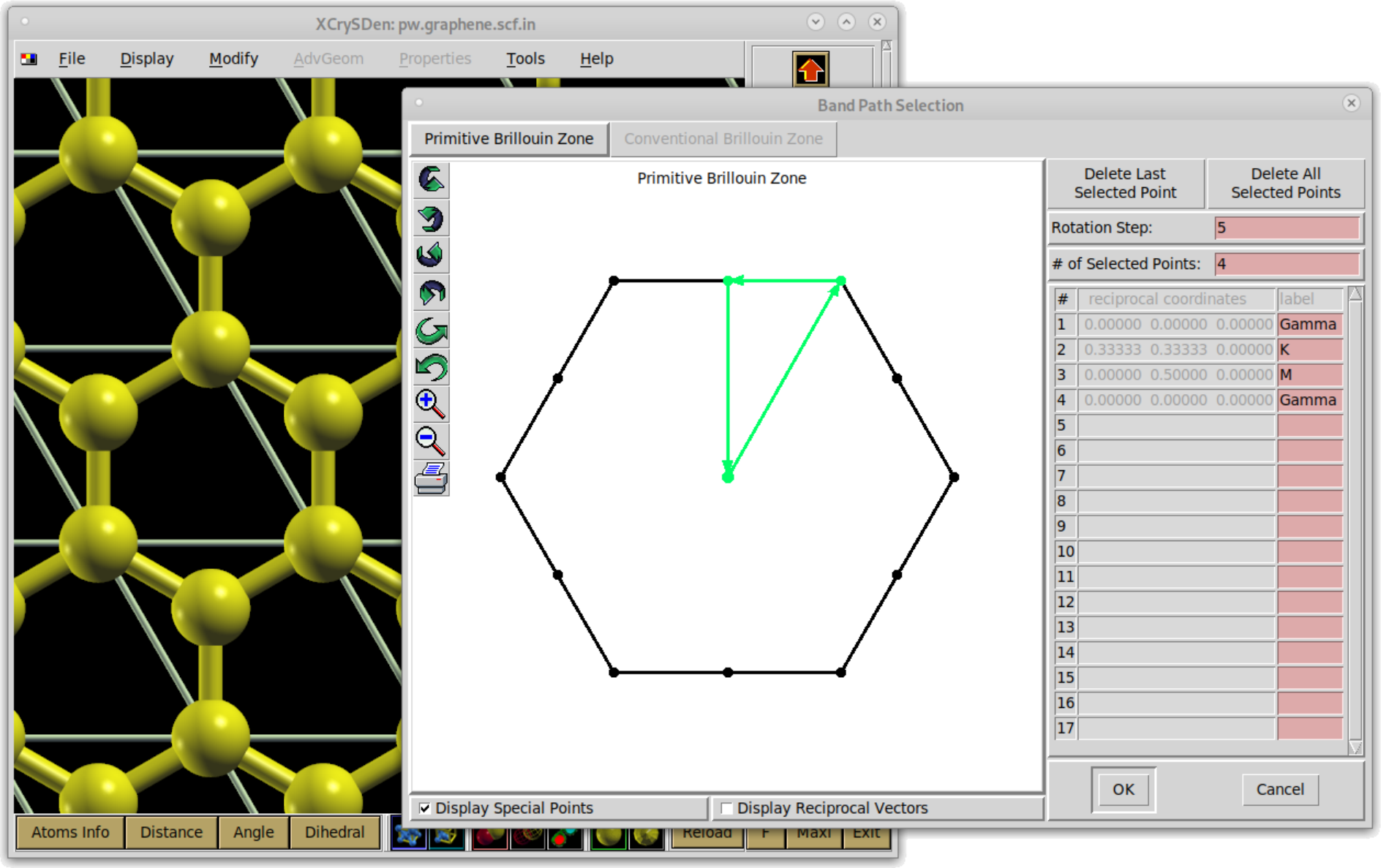
- DOS is typically calculated by a `pw.x` SCF calculation followed by a `pw.x` non-SCF calculation (`calculation = 'nscf'`) with a denser k-point grid, and finally using `dos.x` post-processing code.

- to calculate the bands (spaghetti plot), the `pw.x` SCF calculation is followed by a `pw.x` "bands"-type non-SCF calculation (`calculation = 'bands'`), for which we need a suitable path of k-points. 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`.

- instructions for how to calculate DOS and bands are in `README.md`

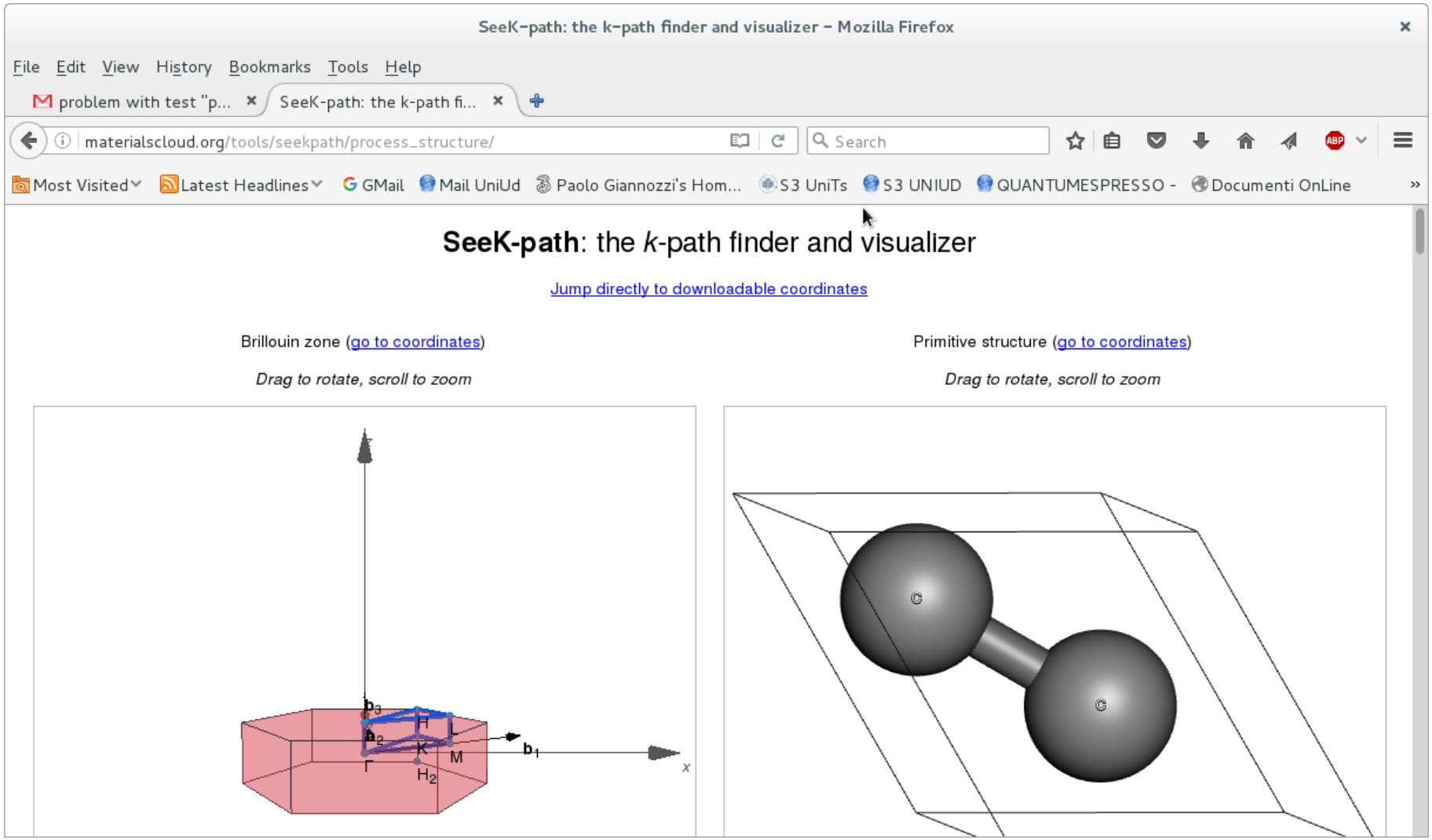
K-path selection tool of xcrysden



(**important:** to save k-path in Quantum ESPRESSO format, explicitly specify the `*.pwscf` extension)

Example: `example2.graphene/`

SeeK-path @ http://materialscloud.org/tools/seekpath



Example: `example2.graphene/`

3. Bulk system: Silicon

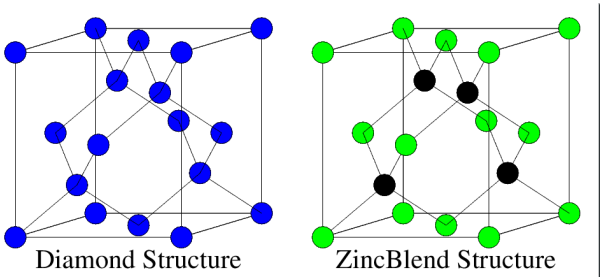
Self-consistent calculation (and a series of tests) for Silicon in the diamond structure:

- move to `example3.Si/` directory
- look at the input file `pw.si.scf.in`. It is composed of three "namelists" `&CONTROL` (note that `calculation = 'scf'` is the default value), `&SYSTEM`, `&ELECTRONS`, followed by three "cards" `ATOMIC_SPECIES`, `ATOMIC_POSITIONS`, `K_POINTS`
- in the `&CONTROL` namelist notice the following two variables (they are commented):
  - `outdir`: temporary directory for large files. Must be writable, will be created if not existent. You may set environment variable `ESPRESSO_TMPDIR` instead.
  - `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.

(note that for the hands-on exercises we rely on `outdir` and `pseudo_dir` variables which are set in the input files.)

Providing atomic structure in input

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



- What is the Bravais lattice?
  `ibrav=2`, meaning fcc lattice
- How many and which parameters are needed to completely define Bravais lattice?
  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 axes, in units of a ("`alat`")

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

Brillouin zone (BZ) sampling

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

- Whether to provide a list of k-points or a uniform grid

- If a list is chosen: provide a 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 an auxiliary code `kpoints.x`, 3) **use uniform grids**

- If a uniform grid is chosen: specify Monkhorst-Pack parameters (*Phys. Rev. B* **13**, 5188 (1976)) and offsets along the three directions (uniform k-point grids are covered in more detail in section 1.2 below).

Running the pw.x code

For serial (single processor) execution you can use

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

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

Look at the directory specified by `outdir` (in our case `example3.Si`) and its content:

- `ls ./`
  `silicon.save silicon.xml`

(to see only these files, you may need to use `ls silicon.*`)

The directory contains a data directory (`silicon.save/`) with binary data files for further processing and an XML file (`silicon.xml`) with general information on the run. The name of the various files is determined by the value of the `prefix` variable and by their content.

Do not run two instances of `pw.x` that access the same `outdir` with the same `prefix`! Unpredictable behavior may follow (the directory is used for temporary files as well). In case of trouble, clean `outdir`.

Running the pw.x code (II)

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

```
$ grep -e "total energy" -e estimated pw.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.

1. Convergence tests for Si bulk

Convergence tests for Si bulk consist of the following steps:

1. convergence with respect to basis-set, i.e., kinetic energy cutoff (variable `ecutwfc`)
2. convergence with respect to k-points (card `K_POINTS`)
3. with converged `ecutwfc` and k-points, determine the lattice parameter of Si bulk
4. Bonus: with converged parameters (`ecutwfc`, k-points, and lattice parameter), calculate a band structure of Si-bulk

1.1 Convergence w.r.t. the kinetic energy cutoff

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

(the default for the charge density is `ecutrho=4*ecutwfc`, which is OK for norm-conserving PPs)

- A manual test of convergence w.r.t. kinetic energy cutoff entails the following tasks (**BEWARE: we will not do it manually!**)

  1. change `ecutwfc` in the `pw.si.scf.in` input to, e.g., 16, 20, 24, 28, 32 Ry
  2. for each value of `ecutwfc`, run `pw.x` and collect the final total energy
  3. collect the data in a file, say `si.etot_vs_ecut` (i.e. each line should contain two values: `ecutwfc` and "total-energy")
  4. plot the energies collected in `si.etot_vs_ecut` using your preferred plotting program, for instance:

     - `gnuplot`

```
gnuplot> plot 'si.etot_vs_ecut' with lines
```

- because such a manual procedure is very cumbersome we use scripts instead

1.1 Convergence w.r.t. kinetic energy cutoff (II)

To make convergence tests easier and faster, scripts are commonly used. To this end, Unix shell-scripts have been traditionally used.

- A Unix shell-script is located in `ex1.ecutwfc.classic/` sub-directory (file: `ecutwfc.sh`)

Unix shell-script

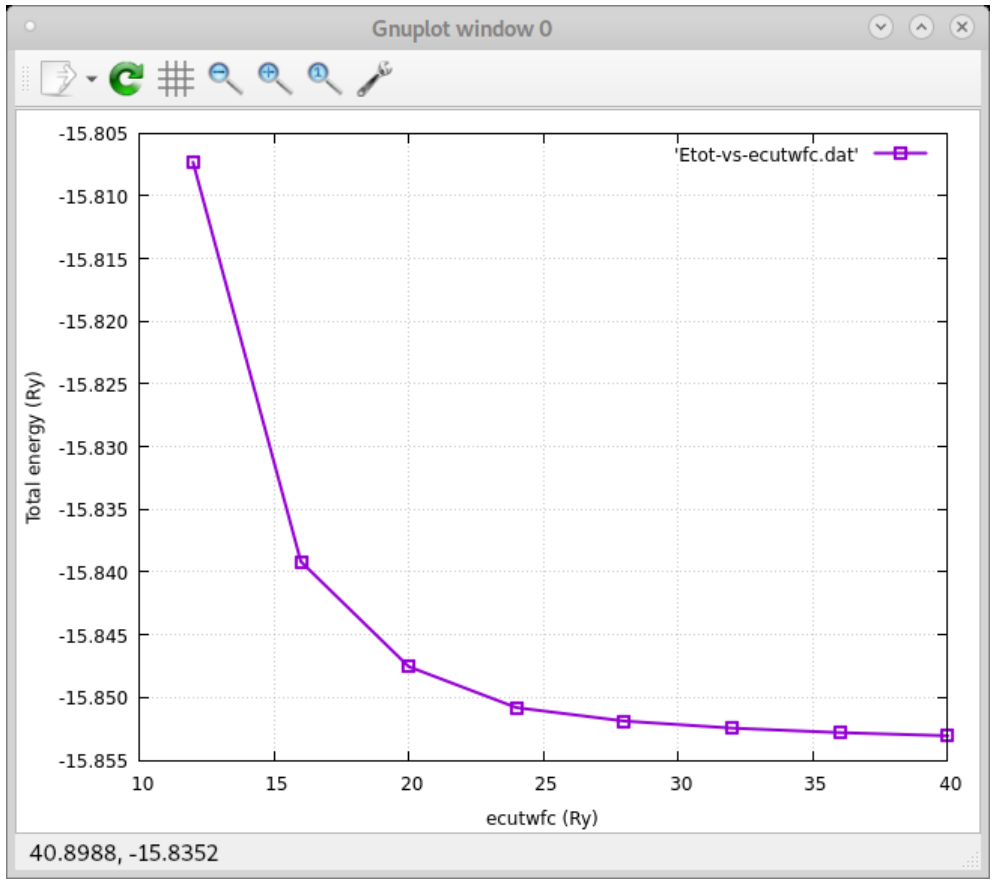
```
#!/bin/sh

rm -f si.etot_vs_ecut.dat

for ecut in 12 16 20 24 28 32
do
    cat > pw.si.scf.$ecut.in << EOF
 &CONTROL
    prefix='silicon',
 /
 &SYSTEM
    ibrav =  2,
    celldm(1) = 10.2,
    nat =  2,
    ntyp = 1,
    ecutwfc = $ecut,
 /
 &ELECTRONS
 /
ATOMIC_SPECIES
   Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS
   Si 0.00 0.00 0.00
   Si 0.25 0.25 0.25
K_POINTS automatic
   4 4 4   1 1 1
EOF

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

    grep -e 'kinetic-energy cutoff' -e ! pw.si.scf.$ecut.out | \
        awk '/kinetic-energy/ {ecut=$(NF-1)}
             /!/              {print ecut, $(NF-1)}' >> si.etot_vs_ecut.dat

done
```

PWTK script

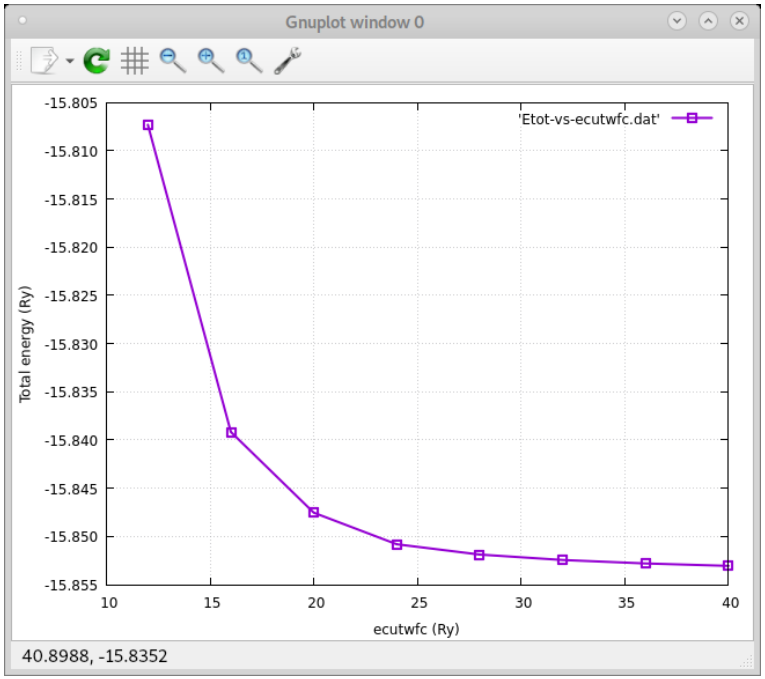
```
load_fromPWI ../pw.si.scf.in

set fid [open si.etot_vs_ecut.dat w]

foreach ecut {12 16 20 24 28 32} {
    SYSTEM "ecutwfc = $ecut"
    runPW pw.Si.scf.$ecut.in

    puts $fid "$ecut [::pwtk::pwo::totene pw.Si.scf.$ecut.out]"
}
close $fid
```

1.1 Convergence w.r.t. the kinetic energy cutoff (III)

- To run the convergence test via the Unix shell-script move to `ex1.ecutwfc.classic/` sub-directory (read the `README.md` file) and execute
  - `./ecutwfc.sh`



1.1 Convergence w.r.t. the kinetic energy cutoff (IV)



Notes:

- Convergence w.r.t. cutoff is a property of the *pseudopotential(s) used*.
- Convergence of the *absolute energy* is typically slower than convergence of *interesting physical properties*, e.g. structure.
- Absolute values of total energy do not have any physical meaning (and depend upon the specific pseudopotential): only energy *differences* have meaning

1.2 Convergence w.r.t. k-points

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

To test the convergence w.r.t. k-points, you need to edit the `K_POINTS` card and request *automatic* Monkhorst-Pack grids:

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

then step-wise increase `nk1=nk2=nk3` to, e.g., `2, 4, 6, 8` (keep `k1=k2=k3=1`) and run `pw.x` calculation for each value of `nk1=nk2=nk3`.

For example, with PWTK this can be achieved with the following snippet:

```
load_fromPWI pw.si.scf.in

foreach k {2 4 6 8} {
   K_POINTS automatic "$k $k $k  1 1 1"
   runPW pw.si.scf.$k.in
}
```

1.2 Convergence w.r.t. k-points (II)

Description of the `K_POINTS` card for *automatic* mode:

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

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. number of k-points
- The `2 2 2  1 1 1` Monkhorst-Pack grid is the same as the "two Chadi-Cohen points" (see: D.J. Chadi and M.L. Cohen, Phys. Rev. B **8**, 5747 (1973))

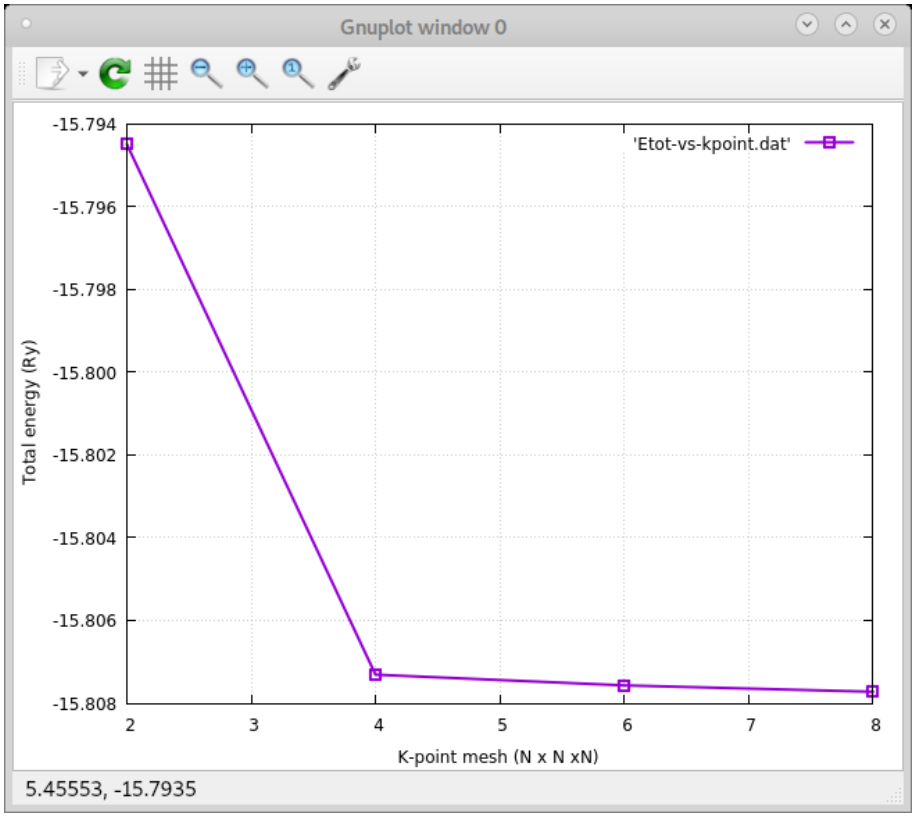
1.2 Convergence w.r.t. k-points (III)

The Unix shell-script for testing the convergence with respect to k-points is located in `example3.Si/ex2.kpoints.classic/` directory (see `README.md` for detailed instructions).

Within this directory execute:

- `./kpoints.sh`

You should get a plot like this one:



1.3 Equation of State: silicon

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

To find the lattice parameter:

- Choose suitable values for `ecutwfc` and k-point grid (e.g. `30` Ry and `4 4 4  1 1 1`)
- Run `pw.x` for values of `celldm(1)` ranging from 9.7 to 10.7 in steps of 0.1 a.u.

With PWTK this can be achieved with the following snippet:

```
load_fromPWI pw.si.scf.in

foreach alat [seq 9.7 0.1 10.7] {
   SYSTEM "celldm(1) = $alat"
   runPW pw.si.scf.$alat.in
}
```

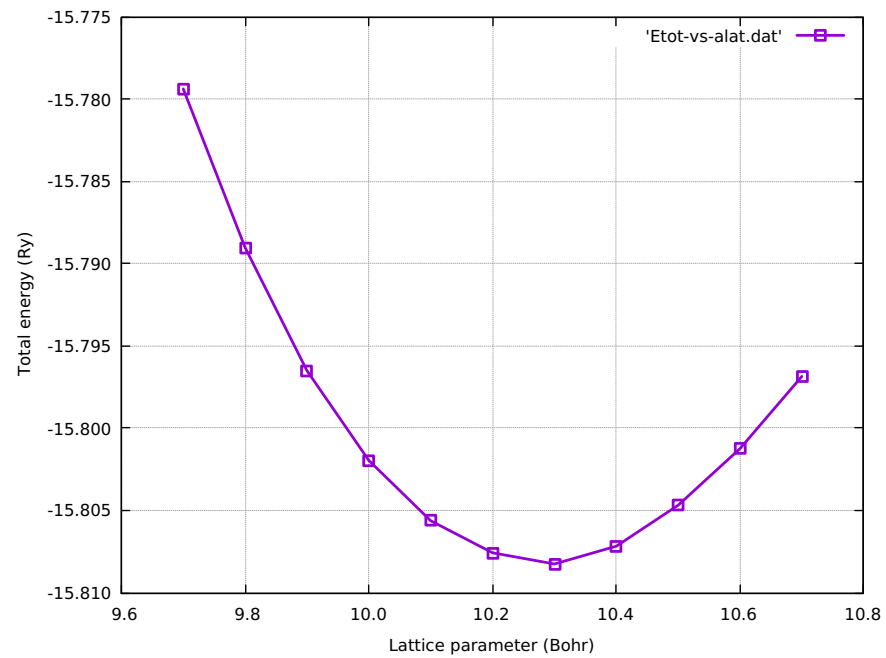
But we are going to use a shell-script.

1.3 Equation of State: silicon (II)

The corresponding Unix shell-script is located in `example3.Si/ex3.alat.classic/` directory (see `README.md` for detailed instructions).

Within this directory execute:

- `./alat.sh`



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 than total energies.

1.3 Equation of State: silicon (III)

Use the code `ev.x` to fit your results to a phenomenological equation-of-state (EOS, e.g. Murnaghan) and to get accurate values for the lattice parameter and for the bulk modulus.

The `ev.x` code prompts for some data and reads a data-file like the one produced by the `alat.sh` script (the data-file is `Etot-vs-alat.dat`). For cubic systems a data-file should contain the following rows:

$$
\begin{array}{ll}
a_1 & E(a_1) \\
a_2 & E(a_2) \\
a_3 & E(a_3) \\
\ldots &
\end{array}
$$

1.4 Band Structure of Silicon

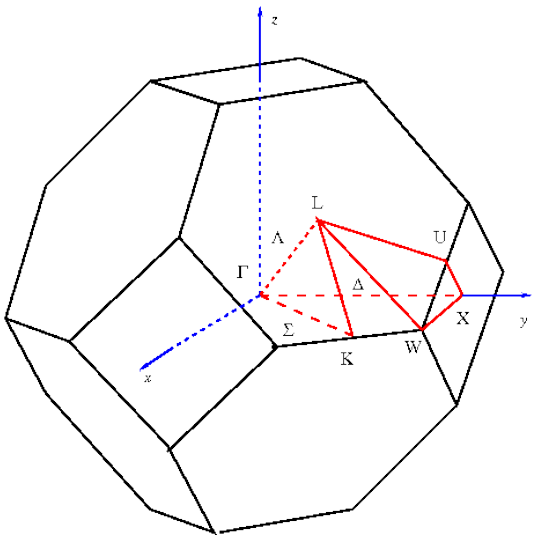
The scheme to calculate the bands (spaghetti plot) is the following:

1. SCF `pw.x` calculation (`calculation = 'scf'`)

2. "bands"-type non-SCF `pw.x` calculation (fixed-potential) with:

   - `calculation = 'bands'`
   - the number of Kohn-Sham states explicitly set (variable `nbnd`)
   - a suitable path of k-points specified in `K_POINTS` (in this example we use the $L-\Gamma-X-W-K-L$ path)

3. `bands.x` calculation, which, among others, produces data-files for the spaghetti plot



Important: `outdir` and `prefix` must be the same for "bands" and "scf" `pw.x` calculations and for the `bands.x` calculation
Important: the k-point path must be continuous in k-space

1.4 Band Structure of Silicon (II)

The input for the bands.x program is the following:

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

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

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

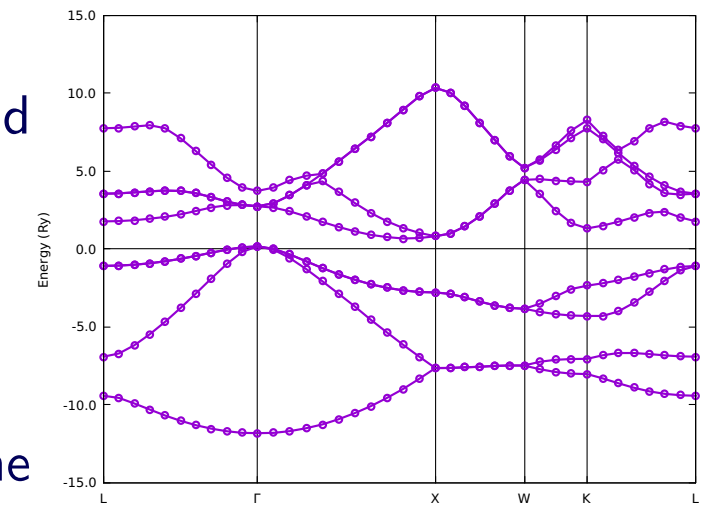
The snippet for running all three calculations manually is:

- pw.x -in pw.Si.scf.in > pw.Si.scf.out
- pw.x -in pw.Si.bands.in > pw.Si.bands.out
- bands.x -in pp.Si.bands.in > pp.Si.bands.out

1.4 Band Structure of Silicon (III)

To execute the PWTK script that will perform all the needed calculations for plotting the bands, proceed as follows:

- move to directory `example1.Si/ex4.bands/` and read the `README.md` file
- set suitable values for `celldm(1)`, `ecutwfc`, and `K_POINTS`
- and execute: `bands.sh`
- you may set the `Efermi` value to the top of the occupied bands in the gnuplot file `plot.gp` (see the instructions in `README.md`); then re-plot spaghetti with: `gnuplot plot.gp`



Auxiliary program plotband.x

plotband.x prompts for terminal input:

```
• plotband.x
Input file > Si.bands.dat
Reading    8 bands at     39 k-points
Range:   -5.6940   16.4680eV  Emin, Emax > -5.6940   16.4680
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 0.7500 0.7500   x coordinate   2.7196
high-symmetry point: -0.5000 0.5000 0.5000   x coordinate   3.3320
output file (gnuplot/xmgr) > Si.bands.plot
bands in gnuplot/xmgr format written to file Si.bands.plot
output file (ps) > (press Return)
```

If symmetry analysis was performed in the previous step, the output is written to several plottable files `Si.bands.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 an fcc lattice.
BUT: it is a metal, only valence bands and a few k-points will not suffice.

- move to the `example4.Al/` directory
- read the `pw.x` input file `pw.al.scf.in`
- notice the presence of new variables: `occupations`, `smearing`, `degauss`;
- run `pw.x` as:
  - `pw.x -in pw.al.scf.in > pw.al.scf.out`
- in the output file notice 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.

2.1 Convergence with respect to k-points, degauss, and smearing

This is a *"three-dimensional"* convergence test, where the number of k-points and values of `degauss` and `smearing` variables are varied. In particular, we will vary:

- `smearing` variable, possible values: `'gauss'` (or `'g'`), `'marzari-vanderbilt'` (or `'m-v'`), `'methfessel-paxton'` (or `'m-p'`)
- `degauss` variable, in range from 0.003 to 0.1
- k-points using the *automatic* grids of `4 4 4`, `8 8 8`, `12 12 12`, and `16 16 16`

With PWTK this can be achieved with the following snippet:CHECK IF SH SCRIPT

```
foreach nk {4 8 12 16} {
    foreach smear {'gauss'  'm-p'  'm-v'} {
        foreach degauss {0.003 0.01 0.03 0.1} {
            SYSTEM " smearing = $smear
                     degauss  = $degauss "
            K_POINTS automatic "$nk $nk $nk   1 1 1"
            runPW pw.Al.scf.$nk.$smear.$degauss.in
        }
    }
}
```

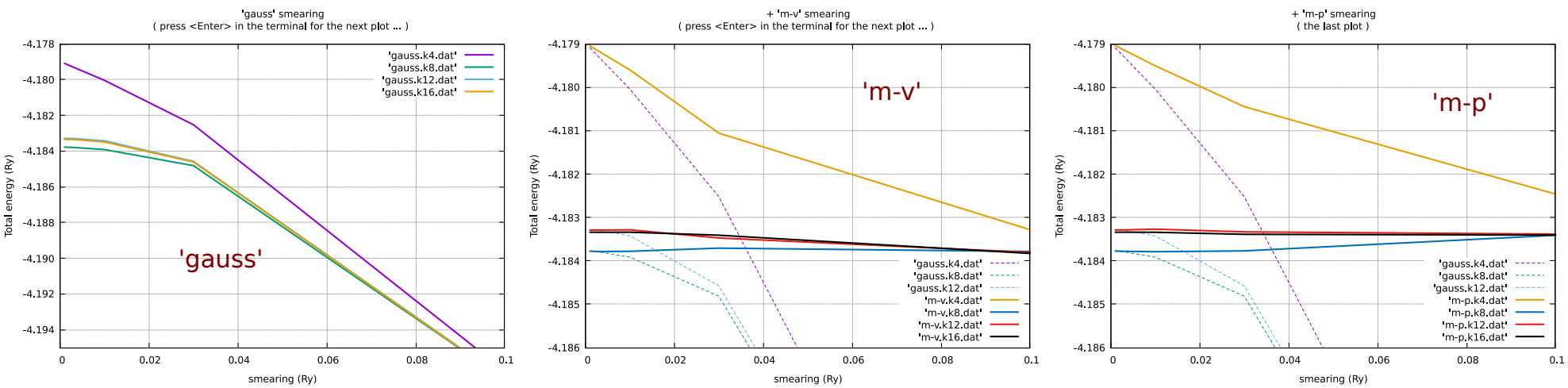
2.1 Convergence with respect to k-points, degauss, and smearing

- move to `example4.Al/ex1.degauss/` directory

- execute: `pwtk degauss.pwtk`

Notice how much slower the convergence is for metals than for insulators!

Both `m-v` and `m-p` depend much less 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` k-point grid and `degauss` ~ 0.01 to 0.05 Ry.



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

2.2 How to plot charge-density

Example `example4.Al/ex2.chdens/` shows how to calculate the valence and the all electron charge density (the latter requires a PAW potential and a very large cutoff energy)

- move to `example4.Al/ex2.chdens/` directory (*chdens* is an acronym for charge-density)

- execute: `pwtk 1-chdens.pwtk`
  this script calculates and "plots" the valence charge density; notice that the electron charge is located mainly in interstitial regions (due to the use of a pseudo-potential, there is *almost no* charge in close vicinity of nuclei; see the next page)

- the scheme to calculate and plot the charge-density is:

  1. make an SCF `pw.x` calculation
  2. make a post-processing `pp.x` calculation (`plot_num=0` for charge density) and instruct the program to write charge density in a suitable format
  3. plot the charge density by `xcrysden` (let's plot density in contour/colorplane style; follow the instructions of tutor and select density range from `0.0` to `0.05`)

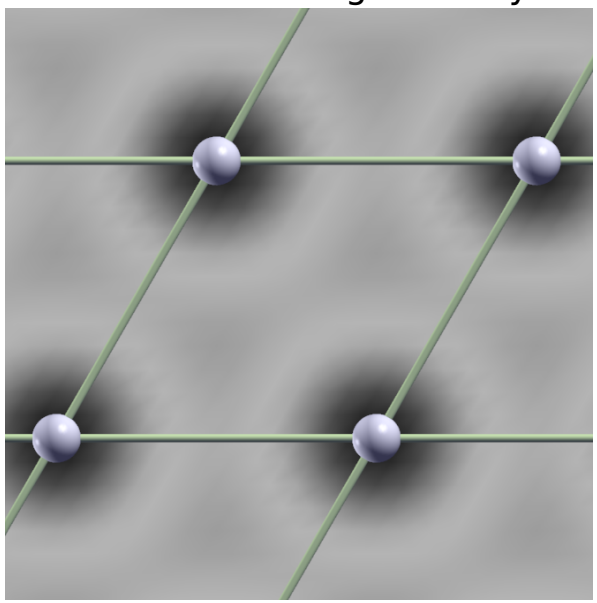
2.2 How to plot charge-density (II)

- to calculate all-electron **valence** and **total** charge densities, execute:
  - `pwtk 2-chdens-paw.pwtk`

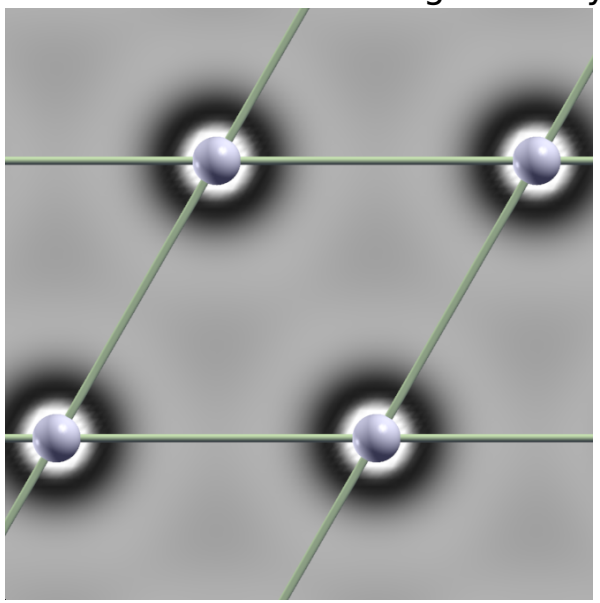
  (note that `plot_num=17` for all-electron valence density and `plot_num=21` for all-electron total density)

- comparison between pseudopotential (PP) valence-density vs. all-electron densities (valence and total):

PP valence charge density



all-electron valence charge density



all-electron total charge density