ATOM

S. Froyen, N.J. Troullier, J.L. Martins, A. Garcia

Jun 23, 2021

CONTENTS

CHAPTER ONE

TECHNICAL REFERENCE

1.1 ATOM User manual

author Alberto Garcıa, ICMAB-CSIC, Barcelona albertog@icmab.es

1.1.1 PREFACE

ATOM is the name of a program originally written (circa 1982) by Sverre Froyen at the University of California at Berkeley, modified starting in 1990 by Norman Troullier and Jose Luis Martins at the University of Minnesota, and currently maintained by Alberto Garcia, who added some features and made substantial structural changes to the April 1990 (5.0) Minnesota version while at Berkeley and elsewhere.

Jose Luis Martins is maintaining his own version of the code:

```
http://bohr.inesc-mn.pt/~jlm/pseudo.html
```

The program's basic capabilities are:

- All-electron DFT atomic calculations for arbitrary electronic configurations.
- Generation of ab-initio pseudopotentials (several flavors).
- Atomic calculations in which the effect of the core is represented by a previously generated pseudopotential. These are useful to make sure that the pseudopotential correctly reproduces the all-electron results for the valence complex.

1.1.2 A PRIMER ON AB-INITIO PSEUDOPOTENTIALS

In this case more than ever, there is a lot to be gained from reading the original literature… Here are some basic references:

- Original idea of the ab-initio pseudopotential:

  Kerker, J. Phys. C 13, L189-94 (1980)
  Hamann, Schluter, Chiang, Phys. Rev. Lett. 43, 1494 (1979)

- More on HSC scheme:

  Bachelet, Schluter, Phys. Rev. B 25, 2103 (1982)

Bachelet, Hamann, Schluter, Phys. Rev. B 26, 4199 (1982)

- Troullier-Martins elaboration of Kerker method:

  Troullier, Martins, Phys. Rev. B 43, 1993 (1991)
  Troullier, Martins, Phys. Rev. B 43, 8861 (1991)

- Core corrections:

  Louie, Froyen, Cohen, Phys. Rev. B 26, 1738 (1982)

- The full picture of plane-wave pseudopotential ab-initio calculations:

  W. E. Pickett, "Pseudopotential Methods in Condensed Matter Applications", Computer Physics Reports 9, 115 (1989)

  M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias and J. D. Joannopoulos, "Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients", Rev. Mod. Phys. 64, 1045, (1992)

  Also, the book by Richard Martin "Electronic Structure: Basic Theory and Practical Methods" (Cambridge University Press) has a chapter on pseudopotentials.

- Use in SIESTA:

  J.M. Soler, E. Artacho, J.D. Gale, A. Garcia, J. Junquera, P. Ordejon, D. Sanchez-Portal, "The SIESTA method for ab initio O(N) materials simulation", Jour. Phys.: Condens. Matter, 14, 2745-2779 (2002).

1.1.3 COMPILING THE PROGRAM

(Note that ATOM now depends on the GridXC and xmlf90 libraries for its correct compilation. Please follow the instructions in the `000_INSTALL` file.)

Directory `Tutorial` in the source distribution contains a set of scripts to automate the process of running ATOM and to analyze the results. The file-manipulation details involved in each of the basic functions of all-electron calculations, generation of pseudopotentials, and testing of the pseudopotentias, are taken care of by `ae.sh`, `pg.sh`, and `pt.sh`, respectively, all in the `Tutorial/Utils` directory. These scripts need to know where the ATOM executable `atm` is. If you have moved the `Tutorial` directory around, or you do not have the source, the default location might not be right for you. The easiest way to fix it is to define an environmental variable `ATOM_PROGRAM`. Assuming `atm` is in `/somedir/somewhere`, you would do:

```
ATOM_PROGRAM=/somedir/somewhere/atm ; export ATOM_PROGRAM  # sh-derived shells
setenv ATOM_PROGRAM /somedir/somewhere/atm                 # csh-derived shells
```

Due to the shortcommings of the basic (GNUplot) plotting package used in the Tutorial section, it is also necessary to copy some scripts from a central repository. Again, if the default does not work for you, define the `ATOM_UTILS_DIR` variable:

```
ATOM_UTILS_DIR=/somewhere ; export ATOM_UTILS_DIR  # sh-derived shells
setenv ATOM_UTILS_DIR /somewhere                   # csh-derived shells
```

1.1.4 USING THE ATOM PROGRAM

All-electron calculations

Assume we want to find the orbital eigenvalues, total energy, and/or charge density of Si in its ground state. (You should now go to the `Tutorial/All_electron` directory and try the following.) Our input file is named `si.ae.inp` and contains the lines (see *Input File description* for more details):

```
   ae Si ground state all-electron
   Si   ca
       0.0
    3    2
    3    0      2.00      0.00
    3    1      2.00      0.00

#234567890123456789012345678901234567890      Ruler
```

We can run the calculation by using the `ae.sh` script. Following the layout of the `Tutorial` directory, we will assume that the script is in the `Tutorial/Utils` directory. We run the script and go into the directory created for the calculation (named as the input file without the extension `.inp`):

```
$ sh ../Utils/ae.sh si.ae.inp
==> Output data in directory si.ae
$ cd si.ae
$ ls
AECHARGE  CHARGE  RHO       charge.gplot   vcharge.gps
AEWFNR0   INP     ae.gplot  charge.gps     vspin.gplot
AEWFNR1   OUT     ae.gps    vcharge.gplot  vspin.gps
$
```

We see some data files (those in all caps) and a few GNUPLOT plotting scripts[1] .

The files are:

- `INP:` A copy of the input file for the calculation.
- `OUT:` Contains detailed information about the run.
- `AECHARGE:` Contains in four columns values of r, the "up" and "down" parts of the *total* charge density, and the total core charge density (the charges multiplied by $4\pi r^2$). `CHARGE` is exactly identical and is generated for backwards compatibility.
- `RHO:` Like `CHARGE`, but without the $4\pi r^2$ factor.
- `AEWFNR0...AEWFNR3:` All-electron valence wavefunctions as function of radius, for s, p, d, and f valence orbitals (0, 1, 2, 3, respectively — some channels might not be available). They include a factor of r, the s orbitals also going to zero at the origin.

It is interesting to peruse the `OUT` file. In particular, it lists the orbital eigenvalues (in Rydbergs, as every other energy in the program):

```
nl    s     occ         eigenvalue    kinetic energy      pot energy

1s   0.0    2.0000    -130.36911241     183.01377616    -378.73491463
2s   0.0    2.0000     -10.14892694      25.89954259     -71.62102169
```

(continues on next page)

[1] GNUPLOT has its issues, but it is free, and installed almost everywhere. Hence we have chosen it as the lowest-common denominator for basic plotting

(continued from previous page)

```
2p   0.0    6.0000      -7.02876268      24.42537874     -68.74331203
3s   0.0    2.0000      -0.79662742       3.23745215     -17.68692611
3p   0.0    2.0000      -0.30705179       2.06135782     -13.62572515
```

(For a relativistic or spin-polarized calculation, there would be "up" and "down" flags in the `s` column above.)

The **plotting** scripts come in two flavors: `.gplot` for terminal use (default X11, use `gnuplot -persist`), and `.gps` for postscript output.

For all-electron calculations, the relevant scripts (without `.gplot` or `.gps` extensions) are:

- `charge:` Charge density (separated core and valence contributions, multiplied by $4\pi r^2$).
- `vcharge:` Valence charge density (same normalization).
- `ae:` Orbital valence wavefunctions (radial part multiplied by r)

Pseudopotential generation

(You should now go to the `Tutorial/Si` directory and try the following.) We are going to generate a pseudopotential for Si, using the Troullier-Martins scheme. The calculation is relativistic and we use the LDA (Ceperley-Alder flavor). The input file is named `Si.tm2.inp` and contains the lines (see *Input File description* for more details):

```
#
#  Pseudopotential generation for Silicon
#  pg: simple generation
#
   pg      Silicon
        tm2      3.0              # PS flavor, logder R
 n=Si  c=car                      # Symbol, XC flavor,{ |r|s}
      0.0       0.0       0.0       0.0       0.0       0.0
    3    4                        # norbs_core, norbs_valence
    3    0      2.00      0.00    # 3s2
    3    1      2.00      0.00    # 3p2
    3    2      0.00      0.00    # 3d0
    4    3      0.00      0.00    # 4f0
     1.90      1.90      1.90      1.90      0.00      0.00
#
# Last line (above):
#    rc(s)     rc(p)     rc(d)     rc(f)   rcore_flag  rcore
#
#234567890123456789012345678901234567890123456789012345678901234567890
```

Note the two extra lines with respect to an all-electron calculation. The pseudopotential core radii for all channels are 1.90 bohrs. Even though they are nominally empty in the ground state, we include the $3d$ and $4f$ states in order to generate the corresponding pseudopotentials.

We can run the calculation by using the `pg.sh` script. Following the layout of the `Tutorial` directory, we will assume that the script is in the `Tutorial/Utils` directory. We run the script and go into the directory created for the calculation (named as the input file without the extension `.inp`):

```
$ sh ../../Utils/pg.sh Si.tm2.inp
==> Output data in directory Si.tm2
==> Pseudopotential in Si.tm2.vps and Si.tm2.psf
$ cd Si.tm2
```

(continues on next page)

(continued from previous page)

```
$ ls [A-Z]*    # show only the data filesAE
CHARGE  AEWFNR3   PSLOGD3  PSPOTR3  PSWFNR3
AELOGD0   CHARGE    PSPOTQ0  PSWFNQ0  RHO
AELOGD1   INP       PSPOTQ1  PSWFNQ1  SCRPSPOTR0
AELOGD2   OUT       PSPOTQ2  PSWFNQ2  SCRPSPOTR1
AELOGD3   PSCHARGE  PSPOTQ3  PSWFNQ3  SCRPSPOTR2
AEWFNR0   PSLOGD0   PSPOTR0  PSWFNR0  SCRPSPOTR3
AEWFNR1   PSLOGD1   PSPOTR1  PSWFNR1  VPSFMT
AEWFNR2   PSLOGD2   PSPOTR2  PSWFNR2  VPSOUT
```

There are quite a few data files now. The new ones are:

- `PSCHARGE:` Contains in four columns values of r, the "up" and "down" parts of the *pseudo valence* charge density, and the pseudo core charge density (see Sect. *4.2.1*) (the charges multiplied by $4\pi r^2$).
- `PSWFNR0...PSWFNR3:` Valence pseudowavefunctions as function of radius, for s, p, d, and f valence orbitals (0, 1, 2, 3, respectively — some channels might not be available). They include a factor of r, the s orbitals also going to zero at the origin.
- `PSPOTR0...PSPOTR3:` Ionic pseudopotentials (i.e. unscreened) as a function of r, for s, p, d, and f channels (0, 1, 2, 3, respectively — some channels might not be available). The last column is $-2Z_{ps}/r$, that is, the Coulomb potential of the pseudo atom. All the ionic pseudopotentials tend to this Coulomb tail for r beyond the range of the core electrons.
- `PSPOTQ0...PSPOTQ3:` Fourier transform $V(q)$ (times q^2/Z_{ps}) of the ionic pseudopotentials as a function of q (in bohr^{-1}), for s, p, d, and f channels (0, 1, 2, 3, respectively — some channels might not be available).
- `PSWFNQ0...PSWFNQ3:` Fourier transform $\Psi(q)$ of the valence pseudowavefunctions as a function of q (in bohr^{-1}), for s, p, d, and f channels (0, 1, 2, 3, respectively — some channels might not be available).
- `VPSOUT, VPSFMT:` Files (formatted and unformatted) containing pseudopotential information. They are used for *ab-initio* codes such as SIESTA and PW. Copies of these files are deposited in the top directory after the run.

The `OUT` file has two sections, one for the all-electron (AE) run, and another for the pseudopotential (PS) generation itself. It is instructive to compare the AE and PS eigenvalues. Simply do

```
$ grep '&v' OUT
 ATM3      12-JUL-02        Silicon
 3s   0.5    2.0000      -0.79937161      0.00000000     -17.74263363
 3p  -0.5    0.6667      -0.30807129      0.00000000     -13.66178958
 3p   0.5    1.3333      -0.30567134      0.00000000     -13.60785822
 3d  -0.5    0.0000       0.00000000      0.00000000      -0.27407047
 3d   0.5    0.0000       0.00000000      0.00000000      -0.27407047
 4f  -0.5    0.0000       0.00000000      0.00000000      -0.26482365
 4f   0.5    0.0000       0.00000000      0.00000000      -0.26482365
---------------------------- &v
 3s   0.5    2.0000      -0.79936061      0.50555315      -3.74113059
 3p  -0.5    0.6667      -0.30804995      0.77243805      -3.26356669
 3p   0.5    1.3333      -0.30565760      0.76702460      -3.25197500
 3d  -0.5    0.0000       0.00000000      0.00140505      -0.07847269
 3d   0.5    0.0000       0.00000000      0.00140505      -0.07847269
 4f  -0.5    0.0000       0.00000000      0.00243411      -0.07586534
 4f   0.5    0.0000       0.00000000      0.00243411      -0.07586534
---------------------------- &v
```

(The AE and PS eigenvalues are not *exactly* identical because the pseudopotentials are changed slightly to make them approach their limit tails faster).

The relevant plotting scripts (without `.gplot` or `.gps` extensions) are:

- `charge:` It compares the AE and PS charge densities.
- `pseudo:` A multi-page plot showing, on one page/window per channel:
  - The AE and PS wavefunctions
  - The AE and PS logarithmic derivatives.
  - The real-space pseudopotential
  - The Fourier-transformed pseudopotential (times q^2/Z_{ps})
- `pots:` All the real-space pseudopotentials.

Core Corrections

The program can generate pseudopotentials with the non-linear exchange-correlation correction proposed in S. G. Louie, S. Froyen, and M. L. Cohen, Phys. Rev. B 26, 1738 (1982).

In the traditional approach (which is the default for LDA calculations), the pseudocore charge density equals the charge density outside a given radius r_{pc}, and has the smooth form

$$\rho_{pc}(r) = Ar\sin(br)$$

inside that radius. A smooth matching is provided with suitable A and b parameters calculated by the program.

A new scheme has been implemented to fix some problems in the generation of GGA pseudopotentials. The smooth function is now

$$\rho_{pc}(r) = r^2 \exp\left(a + br^2 + cr^4\right)$$

and derivatives up to the second are continuous at r_{pc}.

To use core corrections in the pseudopotential generation the jobcode in the first line should be `pe` instead of `pg`.

The radius r_{pc} should be given in the sixth slot in the last input line (see above). If it is negative or zero (or blank), the radius is then computed using the fifth number in that line (`rcore_flag`, see the example input file above) and the following criterion: at r_{pc} the core charge density equals `rcore_flag`*(valence charge density). It is *highly recommended* to set an explicit value for the pseudocore radius r_{pc}, rather than letting the program provide a default.

If `rcore_flag` is input as negative, the full core charge is used. If `rcore_flag` is input as zero, it is set equal to one, which will be thus the default if `pe` is given but no numbers are given for these two variables.

The output file contains the radius used and the A (a) and b (and c) parameters used for the matching. The `VPSOUT` and `VPSFMT` files will contain the pseudocore charge in addition to the pseudopotential.

It is possible to override the default (new scheme for GGA calculations, old scheme for LDA calculations) by using the directives

```
%define NEW_CC
%define OLD_CC
```

The program will issue the appropriate warnings. (See the *Input File description*)

Relevant files:

- `PSCHARGE:` Contains the pseudocore charge in column four. (multiplied by $4\pi r^2$).
- `COREQ:` Fourier transform of the pseudocore charge density $\rho_{pc}(q)$ in units of electrons, with q in bohr^{-1}.

Useful plotting scripts (without `.gplot` or `.gps` extensions) are:

- `charge`: Shows also the pseudocore charge.
- `coreq`: Shows the Fourier transform of the pseudocore charge.

Pseudopotential test

While it is helpful to "have a look" at the plots of the pseudopotential generation to get a feeling for its quality, there is no substitute for a proper **transferability testing**. A pseudopotential with good transferability will reproduce the all-electron energy levels and wavefunctions in arbitrary environments, (i.e., in the presence of bonding, which always takes place when forming solids and molecules). We know that norm conservation guarantees a certain degree of transferability (usually seen clearly in the plot of the logarithmic derivative), but we can get a better assessment by performing all-electron and "pseudo" calculations on the same series of atomic configurations, and comparing the eigenvalues and excitation energies.

In the same `Tutorial/Si` directory we can find file `Si.test.inp`, containing the concatenation of ten jobs. The first five are all-electron (`ae`) calculations, and the last five, pseudopotential test (`pt`) runs for the same configurations:

```
#
# All-electron calculations for a series of Si configurations
#
   ae Si Test -- GS 3s2 3p2
   Si   ca
       0.0
    3    2
    3    0      2.00
    3    1      2.00
   ae Si Test -- 3s2 3p1 3d1
   Si   ca
       0.0
    3    3
    3    0      2.00
    3    1      1.00
    3    2      1.00
   ae Si Test -- 3s1 3p3
   Si   ca
       0.0
    3    2
    3    0      1.00
    3    1      3.00
   ae Si Test -- 3s1 3p2 3d1
   Si   ca
       0.0
    3    3
    3    0      1.00
    3    1      2.00
    3    2      1.00
   ae Si Test -- 3s0 3p3 3d1
   Si   ca
       0.0
    3    3
    3    0      0.00
    3    1      3.00
```

(continues on next page)

(continued from previous page)

```
    3    2      1.00
#
# Pseudopotential test calculations
#
   pt Si Test -- GS 3s2 3p2
   Si   ca
       0.0
    3    2
    3    0      2.00
    3    1      2.00
   pt Si Test -- 3s2 3p1 3d1
   Si   ca
       0.0
    3    3
    3    0      2.00
    3    1      1.00
    3    2      1.00
   pt Si Test -- 3s1 3p3
   Si   ca
       0.0
    3    2
    3    0      1.00
    3    1      3.00
   pt Si Test -- 3s1 3p2 3d1
   Si   ca
       0.0
    3    3
    3    0      1.00
    3    1      2.00
    3    2      1.00
   pt Si Test -- 3s0 3p3 3d1
   Si   ca
       0.0
    3    3
    3    0      0.00
    3    1      3.00
    3    2      1.00
```

The configurations differ in the promotion of electrons from one level to another (it is also possible to transfer *fractions* of an electron).

We can run the file by using the `pt.sh` script. Following the layout of the `Tutorial` directory, we will assume that the script is in the directory directly above the current one. We need to give it **two** arguments: the calculation input file, and the file containing the pseudopotential we want to test. Let's make the latter `Si.tm2.vps`:

```
$ sh ../../Utils/pt.sh Si.test.inp Si.tm2.vps
==> Output data in directory Si.test-Si.tm2
$ cd Si.test-Si.tm2/
$ ls [A-Z]*
AECHARGE  AEWFNR1  CHARGE  OUT       PTWFNR0  PTWFNR2  VPSIN
AEWFNR0   AEWFNR2  INP     PTCHARGE  PTWFNR1  RHO
```

The working directory is named after *both* the test and pseudopotential files. It contains several new files:

- VPSIN: A copy of the pseudopotential file to be tested.
- PTCHARGE: Contains in four columns values of r, the "up" and "down" parts of the *pseudo valence* charge density, and the pseudo core charge density (see Sect. *4.2.1*) (the charges multiplied by $4\pi r^2$).
- PTWFNR0...PTWFNR3: Valence pseudowavefunctions as function of radius, for s, p, d, and f valence orbitals (0, 1, 2, 3, respectively — some channels might not be available). They include a factor of r, the s orbitals also going to zero at the origin.

The OUT file has two sections, one for the all-electron (AE) runs, and another for the pseudopotential tests (PT). At the end of each series of runs there is a table showing the excitation energies. A handy way to compare the AE and PT energies is:

```
$ grep '&d' OUT
[...elided...]
 &d total energy differences in series
 &d         1        2        3        4        5
 &d  1    0.0000
 &d  2    0.4308   0.0000
 &d  3    0.4961   0.0653   0.0000
 &d  4    0.9613   0.5305   0.4652   0.0000
 &d  5    1.4997   1.0689   1.0036   0.5384   0.0000
*----- End of series ----* spdfg &d&v
 ATM3      12-JUL-02   Si Test -- GS 3s2 3p2
 ATM3      12-JUL-02   Si Test -- 3s2 3p1 3d1
 ATM3      12-JUL-02   Si Test -- 3s1 3p3
 ATM3      12-JUL-02   Si Test -- 3s1 3p2 3d1
 ATM3      12-JUL-02   Si Test -- 3s0 3p3 3d1
 &d total energy differences in series
 &d         1        2        3        4        5
 &d  1    0.0000
 &d  2    0.4299   0.0000
 &d  3    0.4993   0.0694   0.0000
 &d  4    0.9635   0.5336   0.4642   0.0000
 &d  5    1.5044   1.0745   1.0051   0.5409   0.0000
*----- End of series ----* spdfg &d&v
```

The tables (top AE, bottom PT) give the cross-excitations among all configurations. Typically, one should be all right if the AE-PT differences are not much larger than 1 mRy.

You can also compare the AE and PT eigenvalues. Simply do

```
$ grep '&v' OUT | grep s
 ATM3      12-JUL-02   Si Test -- GS 3s2 3p2
 3s   0.0    2.0000      -0.79662742       3.23745215     -17.68692611
 ATM3      12-JUL-02   Si Test -- 3s2 3p1 3d1
 3s   0.0    2.0000      -1.08185979       3.53885995     -18.40569836
 ATM3      12-JUL-02   Si Test -- 3s1 3p3
 3s   0.0    1.0000      -0.85138783       3.35438895     -17.96219240
 ATM3      12-JUL-02   Si Test -- 3s1 3p2 3d1
 3s   0.0    1.0000      -1.11431855       3.62997498     -18.60814708
 ATM3      12-JUL-02   Si Test -- 3s0 3p3 3d1
 3s   0.0    0.0000      -1.14358268       3.71462770     -18.79448684
*----- End of series ----* spdfg &d&v
 ATM3      12-JUL-02   Si Test -- GS 3s2 3p2
 1s   0.0    2.0000      -0.79938037       0.50556261      -3.74114712
```

(continues on next page)

(continued from previous page)

```
 ATM3      12-JUL-02   Si Test -- 3s2 3p1 3d1
 1s   0.0    2.0000      -1.08384468       0.55070398      -3.81988817
 ATM3      12-JUL-02   Si Test -- 3s1 3p3
 1s   0.0    1.0000      -0.85392666       0.52020429      -3.76852577
 ATM3      12-JUL-02   Si Test -- 3s1 3p2 3d1
 1s   0.0    1.0000      -1.11546463       0.56048425      -3.83646615
 ATM3      12-JUL-02   Si Test -- 3s0 3p3 3d1
 1s   0.0    0.0000      -1.14353959       0.56945741      -3.85106049
*----- End of series ----* spdfg &d&v
```

(and similarly for p, d, and f, if desired). Again, the typical difference should be of around 1 mRyd for a "good" pseudopotential. (The *real* proof of good transferability, remember, can only come from a molecular or solid-state calculation). Note that the PT levels (in older versions of ATOM) are labeled starting from principal quantum number 1.

The relevant plotting scripts (without `.gplot` or `.gps` extensions) are:

- `charge:` It compares the AE and PT charge densities.
- `pt:` Compares the valence all-electron and pseudo-wavefunctions.

1.1.5 APPENDIX: THE INPUT FILE

For historical reasons, the input file is in a rigid column format. Fortunately, most of the column fields line up, so the possibility of errors is reduced. We will begin by describing in detail a very simple input file for an **all-electron calculation** for the ground state of Si. More examples can be found in the `Tutorial` directory.

The file itself is:

```
#
#  Comments allowed here
#
   ae Si ground state all-electron
   Si   car
       0.0
    3    2
    3    0      2.00      0.00
    3    1      2.00      0.00
#
# Comments allowed here
#
#234567890123456789012345678901234567890      Ruler
```

- The first line specifies:
  - The calculation code (`ae` here stands for "all-electron").
  - A title for the job (here `Si ground state all-electron`).

  (format 3x,a2,a50)
- Second line:
  - Chemical symbol of the nucleus (here `Si`, obviously)
  - Exchange-correlation type. Here, `ca` stands for Ceperley-Alder. The options are:

    Local density approximations (LDA):

- ∗ `wi`: Wigner, PR 46, 1002 (1934)
- ∗ `hl`: Hedin and Lundqvist, J. Phys. C 4, 2064 (1971)
- ∗ `gl`: Gunnarson and Lundqvist, PRB 13, 4274 (1976)
- ∗ `bh`: von Barth and Hedin, J. Phys. C 5, 1629 (1972)
- ∗ `ca`: Ceperley-Alder, parametrized by Perdew and Zunger, PRB 23, 5075 (1981)
- ∗ `pw`: PW92 (Perdew and Wang, PRB, 45, 13244 (1992)) (recommended LDA functional)

Generalized gradient approximations (GGA), implemented as Balbás, Martins, and Soler, PRB 64, 165110 (2001):

- ∗ `pb`: PBE (Perdew, Burke, and Ernzerhof, PRL 77, 3865 (1996)) (recommended GGA)
- ∗ `wp`: PW91 (Perdew and Wang, JCP, 100, 1290 (1994))
- ∗ `rp`: RPBE (Hammer, Hansen, and Norskov, PRB 59, 7413 (1999))
- ∗ `rv`: revPBE (Zhang and Yang, PRL 80, 890 (1998))
- ∗ `ps`: PBEsol (Perdew et al, PRL 100, 136406 (2008))
- ∗ `wc`: WC (Wu and Cohen, PRB 73, 235116 (2006))
- ∗ `jo`: PBEJsJrLO. This and the next three functionals are reparametrizations of the PBE functional by Pedroza et al, PRB 79, 201106 (2009) and Odashima et al, J. Chem. Theory Comp. 5, 798 (2009). Js and Jr refer to jellium surface energy and jellium response, respectively. LO refers to Lieb-Oxford bound. Gx and Gc refer to gradient expansions for exchange and correlation, respectively. HGE refers to the low-density limit of the homogeneous electron gas.
- ∗ `jh`: PBEJsJrHEG
- ∗ `go`: PBEGcGxLO
- ∗ `gh`: PBEGcGxHEG
- ∗ `am`: AM05 (Armiento and Mattsson PRB 72, 085108 (2005), PRB 79, 155101 (2009)),
- ∗ `bl`: BLYP (Becke, PRA 38, 3098 (1988) and Lee, Yang, and Parr, PRB 37, 785 (1988))

Van der Waals (VDW) density functionals, implemented as Román-Pérez and Soler, PRL 103, 096102 (2009):

- ∗ `vw` or `vf`: DRSLL (Dion et al, PRL 92, 246401 (2004))
- ∗ `vl`: LMKLL (Lee et al, PRB 82, 081101 (2010))
- ∗ `vk`: KBM (Klimes, Bowler, and Michaelides, JPCM 22, 022201 (2009))
- ∗ `vc`: C09 (Cooper, PRB 81, 161104 (2010))
- ∗ `vb`: BH (Berland and Hyldgaard, PRB 89, 035412 (2014))
- ∗ `vv`: VV (Vydrov and VanVoorhis, JCP 133, 244103 (2010))

Support for libxc functionals (if compiled in in libGridXC) is available through the use of the special code `xc`, and a composite integer of the form 0XXX0YYY that follows later in the line. For example:

```
#  Example with libxc functional
  pg      Silicon
       tm2      3.0             # PS flavor, logder R
 n=Si c=xcr 01010130      # Symbol, XC flavor,{ |r|s} {Libxc code}
       0.0      0.0      0.0      0.0      0.0      0.0
```

(continues on next page)

(continued from previous page)

```
     3    4                    # norbs_core, norbs_valence
     3    0      2.00      0.00    # 3s2
     3    1      2.00      0.00    # 3p2
     3    2      0.00      0.00    # 3d0
     4    3      0.00      0.00    # 4f0
       1.90      1.90      1.90      1.90      0.00      0.00
#234567890123456789012345678901234567890123456789012345678901234567890
```

where the code "01010130" packs the two codes "101" and "130" for the PBE functionals.

- The character `r` next to `ca` is a flag to perform the calculation relativistically, that is, solving the Dirac equation instead of the Schrodinger equation. The full range of options is:
  * `s` : Spin-polarized calculation, non-relativistic.
  * `r`: Relativistic calculation, obviously polarized.
  * (blank) : Non-polarized (spin ignored), non-relativistic calculation.

(format 3x,a2,3x,a2,a1,1x,i8)

- Third line. Its use is somewhat esoteric and for most calculations it should contain just a 0.0 in the position shown, but that first field might be useful to generate pseudopotentials for "atoms" with a fractional atomic number (see the example for ON in the `Tutorial/PS_Generation` directory).

The rest of the file is devoted to the specification of the electronic configuration:

- Fourth line:

  Number of core and valence orbitals. For example, for Si, we have $1s$, $2s$, and $2p$ in the core (a total of 3 orbitals), and $3s$ and $3p$ in the valence complex (2 orbitals).

  (format 2i5)

- Fifth, sixth… lines: (there is one line for each valence orbital)
  - `n` (principal quantum number)
  - `l` (angular momentum quantum number)
  - Occupation of the orbital in electrons.

  (format 2i5,2f10.3)

  (There are two f input descriptors to allow the input of "up" and "down" occupations in spin-polarized calculations (see example below))

Comments or blank lines may appear in the file at the beginning and at the end. It is possible to perform two or more calculations in succession by simply concatenating blocks as the one described above. For example, the following file is used to study the ground state of N and an excited state with one electron promoted from the $2s$ to the $2p$ orbital taking into account the spin polarization:

```
#
   ae N ground state all-electron
   N    cas
       0.0
    1    2
    2    0      2.00      0.00
    2    1      3.00      0.00
#
#  Second calculation starts here
```

(continues on next page)

(continued from previous page)

```
#
   ae N 1s2 2s1 2p4  all-electron
   N    cas
       0.0
    1    2
    2    0      1.00      0.00
    2    1      3.00      1.00

#234567890123456789012345678901234567890123456789      Ruler
```

The different treatment of core and valence orbitals in the input for an all-electron calculation is purely cosmetic. The program "knows" how to fill the internal orbitals in the right order, so it is only necessary to give their number. That is handy for heavy atoms… Overzealous users might want to check the output to make sure that the core orbitals are indeed correctly treated.

For a **pseudopotential test calculation**, the format is exactly the same, except that the job code is `pt` instead of `ae`.

For a **pseudopotential generation run**, in addition to the electronic configuration chosen for the generation of the pseudopotentials (which is input in the same manner as above), one has to specify the "flavor" (generation scheme) and the set of core radii r_c for the construction of the pseudowavefunction. Here is an example for Si using the Hamann-Schluter-Chiang scheme:

```
#
   pg Si Pseudopotencial
        hsc     2.00
   Si   ca
         0
    3    3
    3    0      2.00
    3    1      0.50
    3    2      0.50
      1.12      1.35      1.17       0.0       0.0       0.0
#
#2345678901234567890123456789012345678901234567890123456789   Ruler
---------------------------------------
```

Apart from the `pg` (pseudopotential generation) job code in the first line, there are two extra lines:

- Second line:

  Flavor and radius at which to compute logarithmic derivatives for test purposes.

  The flavor can be one of :

hsc	Hamann-Schluter-Chiang
ker	Kerker
tm2	Improved Troullier-Martins

  The `ker` and `tm2` schemes can get away with larger r_c, due to their wavefunction matching conditions.

  (format 8x, a3, f9.3)

- The last line (before the blank line) specifies:
  - The values of the r_c in atomic units (bohrs) for the s, p, d, and f orbitals (it is a good practice to input the valence orbitals in the order of increasing angular momentum, so that there is no possible confusion).

(format 4f10.5)

- Two extra fields (2f10.5) which are relevant only if non-local core corrections are used (see Sect *4.2.1*).

In the `hsc` example above, only s ,p, and d r_c's are given. Here is an example for Silicon in which we are only interested in the s and p channels for our pseudopotential, and use the Kerker scheme:

```
#
   pg Si Kerker generation
        ker     2.00
   Si   ca
         0
    3    2
    3    0      2.00
    3    1      2.00
      1.80      1.80      0.00       0.0       0.0       0.0

#234567890123456789012345678901234567890123456789012345678890   Ruler
```

This completes the discussion of the more common features of the input file. See the Appendix *6* for more advanced options.

1.1.6 APPENDIX: INPUT FILE DIRECTIVES

The fixed format can be a source of desperation for the beginner, and its rigidity means that it is not easy to add new items to the input. For this purpose, the program takes another route: several variables can be entered in a specially flexible format by means of *directives* at the top of the file. For example

```
%define NEW_CC
.... rest of the input file
```

would signal that we want to use a new core-correction scheme.

There are two kinds of directives, with syntax:

```
%VARIABLE=value
%define NAME
```

In the first case we assign the value `value` to the variable `VARIABLE`. The program can look at the value via a special subroutine call.

The second form is a bit more abstract, but can be understood as assigning a special "existence" value of 1 to the variable `NAME`. Again, the program can check for the existence of the variable via a special subroutine call.

Currently, the program understands the following `NAME`s:

- `COMPAT_UCB`: Revert to the standard circa 1990 UCB values. Note that these correspond to the first released version of Jose Luis Martins code, not to the old Froyen version. (The defaults are: use a denser grid up to larger radii. Use a larger value for the pseudopotential cutoff point. Use the Soler-Balbas XC package.)
- `NEW_CC`: New core-correction scheme
- `OLD_CC`: Old core-correction scheme (see Sect. *4.2.1*)
- `USE_OLD_EXCORR`: Use the old exchange-correlation package.
- `NO_PS_CUTOFFS`: Avoid cutting off the tails of the pseudopotentials. Currently, a simple exponential tapering function is used, which introduces a discontinuity in the first derivative of the ionic pseudopotential.

- `FREE_FORMAT_RC_INPUT:` Use free-format for the input of the cutoff radii and the specification of the core-correction parameters. This is useful for externally-driven runs of ATOM. In this case the user should make sure that all six values (four rc's plus the two cc parameters) are present in the input line.

CHAPTER

TWO

TUTORIALS

This set of tutorials will guide you in the exploration of ATOM's features

---

Note: Input files have a .inp extension, and should be run using the ae.sh (all-electron), pg.sh (ps generation), or pt.sh (ps test) shell scripts in directory Utils. See the manual (Docs/atom.tex) for details.

After the run, the output information and plotting scripts will reside in a sub-directory whose name is that of the input file without the .inp extension.

Please refer to the user manual for the ATOM program for details on how to run the program and how to make sense of the output.

(** The aliases referred to in this section apply only to the "live" tutorials, or to users who have set up the aliases mentioned —- For these exercises we have created the aliases ae, pg, and pt to perform all-electron, pseudopotential generation, and pseudopotential tests, respectively, and the alias gp to stand for gnuplot -persist. The alias energies, when used in the work directory, will show the inter-configuration energy changes (it is equivalent to grep "&d" OUT). The aliases eigenvalues X, with X being s, p, d, will display the appropriate eigenvalues when applied to the OUT file. — **)

2.1 All-electron calculations

This tutorial ontains some exercises to illustrate some general issues involved in computing the electronic structure of the atom. Run the examples by typing, for example:

```
ae si.ae.inp.
```

These exercises are not essential to follow the examples of pseudopotential generation, but they might help to broaden your understanding.

Guide to all-electron calculation examples:

Si: The ground state and the Si+3 ion (unpolarized calculations). Note the insensitivity of the core electrons to the ionization of three valence electrons. The file si.series.ae.inp contains a series of jobs to study several excited states of the atom.

N: We know that Hund's rule should be obeyed for orbitals not completely full. Use the n_hund.ae.inp file to test whether that is so. In the first calculation the effect of the spin is neglected. The second deals with spin polarization effects, but the occupation of the 2p level is "wrong": there are 2 electrons "down" and one "up". The third calculation should correspond to the true ground state, with all the 2p electrons with their spins aligned.

Fe: Here is another example of the importance of spin effects in certain cases. Iron has 6 electrons in the 3d orbital. The proper spin polarization lowers the energy.

Pb: Lead is a heavy atom, and most of the electrons have velocities which are a significant fraction of the speed of light. A relativistic calculation should then be essential. The Dirac equation is solved, and spin effects appear naturally through the j quantum number.

Ba: Its core is quite large, and it is customary to consider the 5s and 5p as "semicore states", putting them in the valence complex.

2.2 Pseudopotential generation

Exercises for pseudopotential generation and test. The material for each exercise is contained in a directory named after the element. Typically there are several XX.YYY.inp files, where XX is the element's symbol, and YYY is some identifier. Input files for tests have the word test somewhere in YYY. Please see the file Guide.txt for more information.

- Basic Example: Si

A simple example to get the mechanics right. Generate a pseudopotential by running pg Si.tm2.inp and analyze and plot the results. Then test the resulting pseudopotential by running pt Si.test.inp Si.tm2.vps.

- A hard element: C

C is a first-row element, and the 2p state does not have nodes, as there are no other p states below it. Thus the pseudization cannot soften the wavefunction by a whole lot, and the pseudopotential can be quite hard. Here we explore two schemes for pseudopotential generation: Hammann-Schluter-Chiang (code hsc), and Troullier-Martins (code tm2). Note how the rc's can be significantly larger for tm2 while maintaining the transferability. Check the "softness" or "hardness" of the resulting pseudopotentials by looking at their fourier transform.

- Core Corrections: Na

There are two pseudopotential input files: One for a normal case without core corrections, and another one with corrections. Note how the transferability of the pseudopotential improves with the use of non-local core corrections.

- Core or valence?: Cu

The d electrons in Cu (and in Ga, and others) can be treated either as "core" or "valence" (and actually as "core but corrected"). First generate and test the "3d in valence" pseudopotential found here (Cu.3dtm2.inp). (You will have to prepare an input file for the test.) Then prepare an input file for a "3d in core" pseudopotential. Generate the pseudo and test it. Finally, put core corrections to the pseudopotential of the "3d in core" case.

- Semicore states: Ba

A somewhat technical example involving semicore states. Both the 5s and 5p states, which are normally thought of as "core states", are put in the valence. As the program can only deal with one pseudized state per angular momentum channel, this implies the elimination of the "genuinely valence" 6s state from the calculation (and also the 6p, not occupied in the atom but involved in scattering of solid-state electrons). The pseudopotential constructed is not expected to reproduce perfectly the 6s and 6p states, as their eigenvalues are more than 1 eV from those of the reference states 5s and 5p, but the actual results are not bad at all. (Use the "gp pt.gplot" command in the test directory. You can change the order of the configurations in the Ba.test.inp file to look at the plots in sequence: only the last configuration is plotted.) Note that the 6s and 6p states have a node, as they must be orthogonal to the 5s and 5p states, respectively.

As explained in the lecture, SIESTA will generate extra Kleinman-Bylander projectors associated to the 5s and 5p orbitals.

*Fe

These are GGA pseudopotentials for Fe, all with core corrections. (Even if it were not strictly necessary, a pseudo-core helps to iron out some numerical instabilities which appear near the origin when the GGA is used.)

Fe.gga-cc.in: Pseudopotential with a 3d6 4s2 configuration. The p-pseudo looks a bit ugly. Increasing rc (p) (Fe.large-rp.inp) fixes this, but is the pseudopotential more or less transferable?

Fe.4s13d7.inp: Pseudopotential with a 3d7-4s1 configuration. (Test the above pseudopotentials with Fe.test.inp) Fe.sc.inp: Pseudopotential with the 3s and 3p electrons in the valence.

Fe.sc.opt.inp: Same as above, but with the rc parameters roughly optimized for transferability while keeping the pseudopotentials relatively soft.

(To test the small-core pseudos we need a special test file: Fe.test.sc.inp) The above examples used the GGA. It has been shown that the LDA predicts the wrong ground-state for bulk Fe!.

2.2.1 Additional notes

Extra exercises to illustrate some important concepts.

1. Consider the hydrogen atom. It might seem perverse to use a formalism that includes interaction between parts of the 'electronic cloud' (and also exchange and correlation effects!) when only one electron is concerned. And it might come as a surprise that the calculated total energy of the atom is only around 10% off (the eigenvalue is off by more than that, but we could claim that 'what is free comes with no guarantee'). What is going on is a near cancellation between the the Hartree term and the exchange-correlation term of the total energy. Perform the calculation and see for yourself. Incidentally, in a true Hartree-Fock calculation the cancellation is perfect.

The lesson of this extreme example is to remember that we are approximating the complicated many-body problem by a 'simple' sort of mean-field theory. There have been attempts to incorporate self-interaction overcounting into the LDA formalism. See for example:

J. Perdew and A. Zunger, Phys. Rev. B. 23, 5048 (1981)

which, by the way, contains also one of the more widely used parametrizations of the Ceperley-Alder exchange-correlation calculations for the electron gas.

2. To generate a pseudopotential one needs to start with an electronic configuration of the atom. Usually it is the ground state, but it need not be so, and in some cases one needs to artificially populate an orbital which is empty in the ground state, just so that the corresponding angular momentum is represented in the pseudopotential. For example, consider Si, whose ground state is [Ne]3s2 3p2. If one uses that configuration to construct the pseudopotential, only the s and p channels will be generated. That is why a configuration such as [Ne]3s2 3p0.50 3d0.5 is chosen (the appearance of fractional occupancies should not scare you – remember we consider the electrons only through their charge density–). The choice is more or less arbitrary, although sometimes it helps to know something about the environment in which the atom is going to find itself in the solid-state calculation (i.e., we would prefer an ionic configuration for Na to do calculations for NaCl). In any case, whatever the starting configuration, the pseudopotential, by definition and construction, should be basically the same, and should give equally satisfactory results when tested in any (up to a limit explored in the following exercise) configuration. Convince yourself of this by choosing different electronic configurations to generate the pseudopotential for a given element (Si, or whatever) and test them on a series of atomic configurations (such as those exemplified bu the ATOM/ae/si.series.ae.inp file).

3. In this exercise we look more closely at the role of the r_c 'core radius' parameter in the generation of a pseudopotential. We know that it is *not* (or it should not be used as) an adjustable parameter to fit condensed-state properties. By construction, the scattering properties of the pseudopotential and the true atomic potential are the same in an energy region around a given eigenvalue. So when we use the pseudopotential to calculate properties of a configuration different that that used for its generation (atomic or solid), we should expect good results, even if the eigenvalue changes due to hybridization, banding, etc. That is what is called transferability.

What is found 'experimentally' is that the larger the r_c, the lower the degree of transferability, but the softer the pseudopotential. By 'softer' we mean here that one needs fewer fourier components to represent it in Fourier space. The price of higher transferability is a 'harder' pseudopotential.

Test this 'empirical rule', using the plots you can generate after each pseudopotential generation.

4. While the 'rule' explored in the previous exercise is inescapable (if r_c diminishes we are closer and closer to the 'wiggly' core region), there are still some opportunities to play with the way in which the pseudo-wavefunction is constructed from the true wavefunction. The idea is to make the pseudopotentials softer for a given degree of

transferability. Thus the different methods: HSC (Hamann-Schluter-Chiang) KER (Kerker), TM2 (Improved Troullier-Martins), VAN (Vanderbilt 1988), BHS (Bachelet-Hamman-Schluter), and many others. (All of them retain the idea of norm conservation. There is a more recent method [Vanderbilt, 1990] in which that idea is abandoned, obtaining 'ultrasoft' pseudopotentials at the expense of some complications in the use of the potential.)

For the purposes of this exercise it will be enough to compare HSC and TM2 potentials. The way in which TM2 fits the pseudo-wavefunction to the true wavefunction allows the use of larger rc's (sometimes even larger than the position of the peak in the wavefunction).

Again, Si could serve as an example, but the true usefulness of the TM2 approach lies in the softer potentials obtained for some 'problem' elements. Try it for C and a transition metal.

5. To obtain the 'bare' pseudopotential one has to unscreen the total potential a valence electron sees. In the standard pseudopotential approximation we unscreen with only the valence charge density, so we are neglecting the effect of the overlap of the core and valence charge densities (we dump the 'core only' terms in the pseudopotential). It is not too serious for the Hartree term, since it is linear (it depends on (n_c + n_v), that is, linearly). But the exchange-correlation term depends on the 1/3th power of the total charge density… The problems associated with this and a way to fix them are explained in:

S.G. Louie, S. Froyen, and M.L. Cohen, Phys. Rev. 26, 1738 (1983) (the paper makes some emphasis on spin-polarized systems, but the method works for all cases).

The problem is more acute the larger the overlap of the 'valence' and 'core' densities. The quotes in the previous sentence refer to the arbitrariness in defining the terms 'core' and 'valence'. Take iron. It would be 'evident' to everybody that, since the 3d orbital is still filling up, one has to consider the 3d electrons as 'valence'. Now consider Zinc. The 3d orbital is full, and there is a strong temptation to consider it as 'core', since it makes up a full shell. If one takes that option (do it), one can see that there is an enormous overlap of the core and valence charge densities {Use the chargec macro [load chargec ; chargec] to plot it, to set a scale favoring the valence density}; it looks as if the valence charge is almost completely contained under the core charge. Clearly this is going to be a 'tough' case for a standard pseudopotential.