Atomic Code Documentation *

October 22, 2010

1 About this code

This is a merged version of the pseudopotential generation code maintained by Prof. J. L. Martins (http://bohr.inesc-mn.pt/∼jlm) with the almost identical parsec version (http://parsec.ices.utexas.edu/). Copyright information from Martins' version is kept. This is the summary of recent modifications:

- Real-space pseudopotentials are printed out on formatted file that is compatible with PARSEC, the real-space DFT program developed in Prof. J.R. Chelikowsky's group. It is also compatible with the SIESTA program. The traditional unformatted file is also written.
- Pseudopotentials in Kleinman-Bylander form are printed out in ascii (formatted) file, as well as binary (unformatted).
- If working directory has output files from a previous run, those files are overwritten in subsequent runs of the code.

Current version number of this code is `5.70`.

1.1 How to compile

Look at the first lines in the `Makefile` file and check if they are consistent with the platform you are using. Also, make sure subroutines `zetime.f` and `zedate` are appropriate. If not, you can copy the appropriate subroutines from subdirectory `/tempo`. Construct the library of programs and compile the programs with (assuming that the f77 is the name of your fortran compiler):

*Written by M. Tiago, November 2004, based on documentation by B. Pfrommer, 1999. Revised by J. L. Martins in 2010.

```
make libatom
f77 atm.f libatom.a -o atom.exe    --> pseudopotential generator
f77 kbconv.f libatom.a -o kbconv.exe  --> Kleinman-Bylander transform
```

2 The input: `atom.dat` file

2.1 Job line

The first line of the `atom.dat` file determines the job performed by the code:

- `ae` Perform all-electron calculation
- `pg` Generate pseudopotential without any non-linear core correction
- `pe` Generate pseudopotential with non-linear partial core correction (core correction exchange)
- `ph` Generate pseudopotential with core correction (Hartree)
- `pt` Test a generated pseudopotential
- `pm` Pseudopotential test + valence charge modify

The first line also has an optional title for the calculation (just for reference purposes). Fortran format: `(3x,a2,a50)`.

2.2 Generator line

In case one of the `p?` job options is used, the second line must give the type ("flavor") of pseudopotential to be generated or tested. Possible choices are:

- `ker` Kerker pseudopotential, see J. Phys. C **13**, L189 (1980)
- `hsc` Haman, Schlüter, Chiang, see Phys. Rev. Lett. **43**, 1494 (1979)
- `bhs` Bachelet, Hammann, Schlüter, see Phys. Rev. B **26**, 4199 (1982)
- `van` Vanderbilt, see Phys. Rev. B **32**, 8412 (1985)
- `oth` datafile made for minimization and logarithm plot.
- `tam` old Troullier-Martins, see Solid State Comm. 74, 613 (1990).

- `tm2` Troullier-Martins, see Phys. Rev. B **43**, 1993 (1991). The most modern of the schemes. Recommended.

Fortran format: `(8x,a3)`.

2.3 Element line

The next line must have the element symbol, choice of exchange-correlation functional to be used and an optional letter for spin-polarization option. The current version has the following functionals implemented:

- `ca` Ceperley-Alder (LDA functional, parametrized by Perdew and Zunger), see Phys. Rev. Lett. **45**, 566 (1980) or Phys. Rev. B **23**, 5048 (1981)
- `pw` Perdew-Wang (GGA functional), see Phys. Rev. B**45**, 13244 (1992)
- `pb` Perdew-Burke-Ernzerhof (simplified GGA functional), see Phys. Rev. Lett. **77**, 3865 (1996)

Spin-polarization options are `s` (non-relativistic, spin-polarized system) or `r` (scalar-relativistic system) and are given together with the exchange-correlation option. For example, an LDA pseudopotential with relativistic corrections included is obtained using the keyword `car`. Non-relativistic and non-spin polarized LDA pseudopotential has the keyword `ca`.

Fortran format: `(3x,a2,3x,a2,a1)`.

2.4 Grid line

In most cases, this line will contain several `0.0` . Those who are "initiated" can choose their own values for nucleus charge (first value), shell charge and radius (next two values), and parameters for the logarithmic grid (next three values). Fortran format: `(6f10.3)`.

2.5 Orbital information block

Then follows a line with the number of core orbitals and the number of valence orbitals (i.e. the number of lines that will follow), and then a list of occupation numbers. Example:

```
1    2
2    0      1.00      1.00
2    1      3.00      1.00
```

This is the input for oxygen. We have 1 core orbital (we count only the l quantum number), and two valence orbitals, the occupation numbers of which are given by the succeeding lines. The first number in each line is the principal quantum number (n), the second one the angular momentum quantum number (l), then the occupation number (or numbers for a spin polarized calculation, as in the above example). Currently, only one valence orbital is allowed per angular momentum quantum number. Maximum value supported for l is 4 (up to five valence orbitals). Format for all line in the block is `(2i5,4f10.3)`.

2.6 Cutoff radii line

The last line finally gives the cutoff radii for each one of the valence orbitals (up to five).

The sixth number in the line is the so-called cfac for the nonlinear core correction. The core charge is cut off by an exponential of a polynomial in the square of the radius. This cut-off is set at the point where the ratio of (core charge)/(valence charge)= cfac. This functional form is better then the Bessel function of the original program, and is smoother to avoid the pesky GGA instabilities . If cfac is entered as zero, it defaults to 1.0, i.e. the core charge is smoothened at the radius where core and valence charge have the same magnitude. CAUTION! If you want to specify a value for cfac and have less than five valence orbitals (which is almost always the case), you *must* give five numbers before the cfac value. Cutoff radii for orbitals that are not used will be ignored. Fortran format for this line is `(7f10.5)`.

The seventh number in the line is the so-called rcfac for the nonlinear core correction. It is the radius of the core charge. If rcfac is zero or neglected then cfac is used to determine rcfac. Note that in literature typically the rcfac is specified, not cfac.

2.7 Successive calculations

Successive calculations can be done by simply concatenating groups of input lines. A blank line just after the cutoff radii line indicates the end of input. For example

```
   ae      Silicon
 n=Si c=ca
       0.0       0.0       0.0       0.0       0.0       0.0
    3    3
    3    0     2.00      0.00
    3    1     2.00      0.00
    3    2     0.00      0.00
```

```
  ae      Silicon
n=Si c=ca
      0.0       0.0       0.0       0.0       0.0       0.0
   3    3
   3    0     2.00      0.00
   3    1     1.00      0.00
   3    2     1.00      0.00
```

will do two calculations in a row and as a bonus will calculate the $p - d$ excitation energy from the difference in total energies.

3 The output: `atom.out` file

This file is largely self-explanatory. All energies are in Rydbergs unless explicitely stated, and all lengths in Bohr radius.

For the wave function output:

- `a extr` is the value of the wave function ψ at the point where $|r\psi|$ is extremal.
- `r extr` is the radius for which $|r\psi|$ is extremal.
- `r zero` is the radius for which $\psi = 0$
- `r 90/99%` is the radius within which 90 or 99 % of the wave functions charge are contained.

In a pseudopotential generation run, all the information needed for parsec is contained in the file `psd.pot`. Just rename it as *xx*`_POTRE.DAT`, where *xx* denotes the chemical symbol for the element you are working on. For example, a carbon pseudpotential file should be renamed as `C_POTRE.DAT`. For silicon, it will be `Si_POTRE.DAT`. These files are readable and easily ported to different machines, but they should not be edited. Blank spaces may be inportant when parsec reads the files.

Those who use the SIESTA program can also use a *xx*`_POTRE.DAT` file as input to SIESTA, with the appropriate name (*xx*`.psf`). The only difference between *xx*`_POTRE.DAT` and *xx*`.psf` files is that the former ones have appended pseudo-wave functions.

4 Partial core correction

In cases where core orbitals have non-negligible interaction with valence orbitals via exchange-correlation or Hartree interaction, that effect can be incorporated in the pseudopotential method using the algorithm proposed in this reference: S. G. Louie, S. Froyen, and M. L. Cohen, Phys. Rev. B **26**, 1738 (1982). This is done by using options `pe` or `ph` in the job line.

The original algorithm by Louie-Froyen-Cohen used a spherical Bessel function to smooth out the pseudocore density. This code uses a Kerker-like smoothing function:

$$p_c(r) = r^2 e^{a_c + b_c r^2 + c_c r^4}$$

where constants a_c, b_c, c_c are chosen so that the pseudocore density and its derivates up to the second are continuous. This new scheme may give results slightly different from the ones obtained with the original smoothing function.

5 Kleinman-Bylander transformation

Pseudopotentials in Kleinman-Bylander (Phys. Rev. Lett. **48**, 1425 (1982)) form, suitable for reciprocal-space calculations, can be generated using the code `kbconv.f`. Input files are `kb.dat` (provided by the user) and `pseudo.dat` or `pseudo.dat`*01* (created by the pseudopotential generator). File `kb.dat` has the choice of local component (first line), grid size and spacing (second line). It first tries to use the `pseudo.dat` file, and if does not exist tries `pseudo.dat`*01*. If you just generated one pseudopotential from one configuration, you do not have to do anything. Otherwise just do `cp pseudo.dat`*XX* `pseudo.dat` with *XX* replaced by the configuration you want to try before running the program. Here is an example of the `kb.dat` file

```
local 0
numberqpt-1600q-spaceing0.015
     0.00
     0.00
     0.00
     0.00
     0.00
```

meaning that the local potential is $\ell = 0$, and that the pseudopotential will be sampled by 1600 points spaced by 0.015 Bohr radii.

To run the program you just have to compile it against the library and run, for example

```
f77 kbconv.f libatom.a -o kbconv.exe
kbconv.exe
```

Output files are:

```
{\tt kb.out}      ---------> Normal human readable output
{\tt potfourkb.dat} ------->  binary for cpw program
{\tt pot\_kb\_format.dat} -> formatted version for cpw program
{\tt potfourkb.asc.dat} ---> formatted for parsec program
{\tt kbplot.dat}  ---------> formatted for conv_plot_kb.f program
{\tt pseudokb.dat} --------> binary for atomkb.f and lnplot.f programs
```

To use the files with the cpw or parsec codes you have to rename them from `potfourkb.asc.dat` to *xx*`_POTKB.ASC.DAT`, `potfourkb.dat` to *xx*`_POTKB.DAT`, and `pot_kb_format.dat` to *xx*`_POTKB_F.DAT`, where it xx is the chemical symbol of your atom. The first two can be used with parsec, the last two can be used with cpw. The formatted files are different for each program , the unformatted is common to both programs.

Original reference for the Kleinman-Bylander transformation is: Phys. Rev. Lett. **48**, 1425 (1982).

The Kleinman-Bylander form of the pseudopotential can have ghost states. Their existence is tested by the method of Phys. Rev. B **41**, 12264 (1990), and you are strongly warned in the `kb.out` file if they are present. There is an "area" criterion (the positive or negative part of the KB operator should dominate) that gives you a mild warning. If you do further tests (next section) and do not see anything bizarre, just ignore the mild warning. As a rule of thumb, just choose the least atractive potential as the local part. If you have trouble, try increasing the core radii.

6 Auxiliary programs

Warning: The changes to the atm.f, kbconv.f and source/*.f are kept to a minimum to avoid introducing bugs (if ain't broken, don't fix it). They could be more elegant, object oriented, you name it, but it is reassuring that lots of people have been using some of those subroutines since the eighties, maybe before you were born...

The auxiliary programs that you find here are not frequently used (although they should be!!!) and have been modified as the graphics environment has been changed. They are less reliable and you should use them with some care.

6.1 Testing the atomic excitation energies for the semi-local pseudopotential

Pseudopotential testing means that we use the pseudopotential in an environment that is different from the one used to generate the pseudopotential, and for which we have reliable data without the pseudopotential aproximation. The simplest test is for the atoms in a different configuration. For example after generating a pseudopotential for silicon with the s^2p^2 ground state configuration, we run all electron calculations for the ionized configurations s^1p^2 and s^2p^1, and for the excited states configuration s^1p^3, $s^1p^2d^1$, and $s^2p^1d^1$, with the pseudopotential and with all-electrons.

Those tests cam be easily done with the `atm.f` program. Using copy and paste we edit `atom.dat` cloning first the input data the necessary number of times, and modifying the occupations. We can then run `atm.f` program with either the all-electron option, in which case we have to remove the lines with the pseudopotential type and core radii, or we can generate a few pseudopotentials, in which case we have to be careful with overwriting the `pseudo.dat`*xx* files. One should also save the original `atom.out` file for future reference.

In previous versions of the program you couldn't overwrite those files to oblige people to rename and save them. Since version 5.70 it is your responsability to take care of that.

Reopen the `atom.dat` file and use the pseudopotential test `pt` as the type of calculation, copy the desired pseudopotential file to `pseudo.dat` and rerun `atm.f`. Compare the energy differences, valence eigenvalues, and outermost valence wavefunction maxima between the two files, and you will get an idea of the transferability of your pseudopotential. Here is an idea of what you would do on the command line.

6.2 Testing the atomic excitation energies for the Kleinman-Bylander pseudopotential

If you are going to use the Kleinman-Bylander form of the pseudopotential it is a good idea to repeat the test above with that form. That can be done with the `atomkb.f` program. Just compile it against the library and repeat the steps of the semi-local test but the second time run the `atomkb.f` program instead of the `atm.f` program. The `atomkb.f` program is going to read the `pseudokb.dat` file which should be around from the last time you run the `kbconv.f` program. Just make sure it is the correct one.

If you are paranoid about ghost states, this would be the ultimate test for their

absence.

6.3 Logarithmic derivatives of the wave-function

For the **screened** pseudopotential we can do a test based on scattering theory. Two potentials have the same scatering at a radius r_0 for an energy E and angular momentum l if

$$\frac{\partial}{\partial E}\frac{\partial}{\partial r}\log(\psi_l^{PP}(r_0;E)) = \frac{\partial}{\partial E}\frac{\partial}{\partial r}\log(\psi_l^{AE}(r_0;E)) \tag{1}$$

where $\psi_l(r;E)$ is the all-electron or pseudo-potential solution of the Schrödinger equation that is regular at the origin.

Using the never cited Topp-Hopfield theorem you can verify that most recipes make sure that the scattering is the same at the eigenvalue energies, and very close near those energies (for r_0 outside the core radius obviously). Using the `lnplot.f` program you can see how well the above condition is satisfied.

To run the test you have first to run the `atm.f` program with the `min` option as the pseudopotential generation choice. It will generate a file `datafile.dat` but no pseudopotential per se. The `lnplot.f` program will use that file and either `pseudokb.dat` or `pseudo.dat01` from the **original** pseudopotential generation, so make sure you have not overwritten them in the mean time. Once you have those files you can run the `lnplot.f` program, answer a couple of questions, and you will have some nice plots on your screen **provided** you have gnuplot on your machine.

If you do not have gnuplot installed you still get a couple of *.gp files that you can use with another plotting package, or you can change the code (send us a copy) to get an interface to your favorite package. I find that gnuplot is very convenient but it is not the nicest package for publication quality figures. In that case I just import the *.gp files into xmgrace and take the time to generate a nice figure.

6.4 Plotting the wavefunctions and potentials

The wavefunctions, potentials and respective transforms can be plotted with the `conv_plot.f` program. The Kleinman-Bylander operators can be plotted with `conv_plot_kb.f` program. They read the `plot.dat01` or `kbplot.dat` from the last time you run the `atm.f` or `kbconv.f` by default. However they will ask you if you want to use other files. After answering a few simple questions you should have the plots on your screen **provided** you have gnuplot on your machine.

If you do not have gnuplot installed you still get a couple of *.gp files that you can use with another plotting package, or you can change the code (send us a

copy) so that it interfaces with your favorite package. I find that gnuplot is very convenient, but maybe not the nicest package for publication quality figures. For that I use grace/xmgrace. I just import the *.gp files into xmgrace and take the time to generate a publication quality figure. For example, to plot the wavefunctions I go to the menu Data—Import—ASCII and choose to load the `wfct.gp` file as a block file instead of a single set. In the new menu I use X from the first column and apply successively the choice of Y to the other columns. In the end I have a basic plot that I can improve at leisure.

6.5 The log of a session

Sometimes it is easier to have just the list of commands as they were executed in a session, so here they are:

```
cd source
make
mv libatom.a ..
cd ..
gfortran atm.f libatom.a -o atom.exe
vi atom.dat        (set up initial calculation)
atom.exe
cat atom.out
gfortran kbconv.f libatom.a -o kbconv.exe
vi kb.dat          (set the choices for transform)
kbconv.exe
cat kb.out
gfortran conv_plot.f -o conv_plot.exe
conv_plot.exe
vi atom.dat        (set the pseudo type to min)
mv atom.out atom_ae.out      (keep it for reference)
atom.exe
cat atom.out
gfortran lnplot.f libatom.a -o lnplot.exe
lnplot.exe
gfortran conv_plot_kb.f -o conv_plot_kb.exe
conv_plot_kb.exe
gfortran atomkb.f libatom.a -o atomkb.exe
vi atom.dat        (set the calculation to pt)
atomkb.exe
```

```
cat atomkb.out
cp pseudo.dat01 pseudo.dat
vi atom.dat
atom.exe
cat atom.out
```

6.6 Diagram of program and data files