# 1 Construction of norm-conserving semi-local pseudopotentials for Si

As discussed in class, it is desirable to replace the effective interaction of the valence electrons with the *ionic core*, i.e. nucleus plus core electrons, by the interaction with a much smoother *pseudopotential* that allows to expand the resulting *pseudo-wavefunctions* using a much smaller number of plane wave basis functions than required for the original potential. Ideally, the pseudopotential is constructed such that the corresponding pseudo-wavefunctions have the same eigenvalues as the original *all-electron wavefunctions*, and that they are identical to the all-electron wavefunctions beyond a certain *matching radius* $r_c$ around the atom.

Specifically, in this first exercise we want to generate a norm-conserving semi-local pseudopotential for Si. The exercise will guide you through the three main steps of pseudopotential generation, i.e:

1. Calculate all-electron wavefunctions for a chosen reference configuration for the atom under consideration. This is done by solving the Kohn-Sham equations for the free atom.
2. Construct smooth pseudo-wavefunctions for the valence electrons that match the corresponding all-electron wavefunctions beyond a suitably chosen matching radius $r_c$. From these pseudo-wavefunctions the ($l$-dependent) pseudopotential is then obtained by inverting the radial Schroedinger equation for the pseudo-wavefunction at the corresponding energy.
3. Test the "transferability" of the obtained pseudopotential by comparing all-electron calculations with pseudopotential calculations for some electronic configurations different from the reference configuration used to generate the pseudopotential.

**Disclaimer:** *Please note that the purpose of this exercise is to better understand certain aspectsof pseudopotential generation, not to teach you how to generate the best (optimally soft and transferable) pseudopotentials for arbitrary systems.* Generating good pseudopotentials that give reliable results when used in many different chemical environments is a difficult task that requires lots of experience.

For this exercise you will use the program '`ld1.x`' that is part of the freely available *Quantum-Espresso* package (see: `www.quantum-espresso.org`). The QuantumEspresso package has already been installed on chuck. The executable files that you need for these exercises are stored in `~kovacikr/src/espresso-4.0.1/bin`. In order to access them easily you should either add this directory to your search path, e.g. by adding the following line to your `~/.bash_profile` file:

```
export PATH=$PATH:/home/users/kovacikr/src/espresso-4.0.1/bin/
```

or establish a link to the required programs in your `~/bin` directory (be sure that `~/bin` is in your search `$PATH`):

```
[edererc@chuck ~]$ ln -s TARGET LINK
```

The program `ld1.x` reads input from `stdin` and writes lots of output to `stdout`. The best way to call it from the command line is the following:

```
[edererc@chuck ~]$ ld1.x < infile > outfile
```

This way `ld1.x` reads all its input from the file '`infile`' and writes all its output (except error messages) to the file '`outfile`'. This allows you to conveniently prepare a text file containing the required input beforehand using your favorite text editor, and both output and input are saved

for future reference. (If you don't know anything about input/output redirection under Linux please check the tutorial notes on the course webpage.)

### 1.1 All-electron calculation for the free Si atom

The first step is to choose a suitable electronic configuration for Si that you will use as reference configuration for the pseudopotential generation. You will then perform an all-electron calculation within the local density approximation (LDA) for a single Si atom in this configuration.

For this, you need to write an input file for the 'ld1.x' program. This input file is a regular ASCII text file that contains information about the calculation you want ld1.x to perform. This information has to be written in a special format, which is explained in detail in the file ~kovacikr/src/espresso-4.0.1/atomic_doc/INPUT_LD1.txt. An example is shown below:

```
&input
   title = 'All-electron calculation for Si atom',
   zed = 14.0,
   iswitch = 1,
   dft = 'LDA',
   prefix = 'Si-output',
   config = '[Ne] 3s2 3p2',
/
```

As you can see this file contains various variable assignments which specify the details of your calculation. The assignments are separated by commas and grouped together in different "namelists" (in fact the above example contains only one namelist called &input). Each namelist starts with the '&' symbol followed by the name of the namelist and ends with ' /'. The namelist "&input" contains all the input required for the all-electron calculation. Later we will add other namelists such as e.g. "&inputp", which contains the information required for the pseudopotential generation. The different variables in the above example have the following meaning:

`title` This is a string (= text) variable identifying your calculation. It is not really used by the program and is only useful for yourself, so that you remember what calculation your input/output files correspond to.

`zed` The nuclear charge number of the atom you want to calculate. (Si: $Z = 14$)

`iswitch` An integer variable indicating what type of calculation you want to do (1=all-electron calculation, 2=PP tests, 3=PP generation).

`dft` This specifies what approximation for the exchange-correlation energy functional $E[n]$ you want to use. So far we have discussed only the local density approximation (LDA) in class, so choose dft='LDA'.

`prefix` Here you can specify a prefix that the program uses for all output files it generates. Using a descriptive prefix makes your life easier if you have many output files lying around in your working directory.

`config` Here you specify the electronic reference configuration for the all-electron calculation. The above example corresponds to the configuration: [Ne], $3s^2$, $3p^2$, where [Ne] is an abbreviation for the noble gas configuration of neon ($1s^2$, $2s^2$, $2p^6$) and e.g. $3p^2$ means that there are 2 electrons in the $3p$ states.

Now, write an input file containing the above specifications and run `ld1.x` on it. Then inspect the output file. You can find the eigenvalues of the different all-electron functions in three different units (Rydberg, Hartree, and electron volts; 1 Ry = 0.5 Ha = 13.6057 eV). In addition, the total energy, and the various contributions to the total energy are printed out. As you can see, the exchange-correlation energy (`Exc`) is indeed the smallest contribution to the total energy. Congratulations, you have just performed your first DFT calculation!

You will also find a file called "*prefix*`.wfc`" in your working directory (where *prefix* is replaced by whatever you specified as `prefix` in your input file). Have a look at this file. It contains various columns of data. The first column contains values for $r$, i.e. the radial grid used in the calculation. The following columns contain the wavefunctions at the corresponding radii. The first line indicates which column belongs to which wavefunction.

You can load the data file containing the wavefunctions directly into a graphics program such as `xmgrace`, using the command:

`[edererc@chuck ~]$ xmgrace -nxy` *prefix*`.wfc`

Familiarize yourself a bit with `xmgrace`. Try to produce a nice graph by adjusting the $x$ and $y$ range of the axes (Menu: Plot → Axis properties), and adding labels to the axes and wavefunctions (Menu: Plot → Set appearance). Can you identify which line corresponds to which wavefunction just by looking at them? (Look at their radial extension and the number of nodes.)

## 1.2 Construction of smooth pseudo-wavefunctions

Next you have to construct smooth pseudo-wavefunctions corresponding to the all-electron valence states, i.e. $3s$ and $3p$ in this case. To do this you have to add a new namelist to your input file containing all the specifications about the pseudopotential generation. The corresponding namelist is called `&inputp` and should contain the following information:

```
&inputp
   pseudotype = 1,
   lloc = 1,
   file_pseudopw = 'Si.UPF',
   tm = .true.,
   author = 'claude',
/
2
3S  1  0  2.00  0.00  2.20  2.20
3P  2  1  2.00  0.00  2.20  2.20
```

The variables in this namelist have the following meaning:

`pseudotype` This selects the type of pseudopotential that will be generated. So far we have only discussed semi-local norm-conserving pseudopotentials, which are selected with `pseudotype = 1`. Other possibilities, such as separable norm-conserving (`pseudotype = 2`) and ultrasoft (`pseudotype = 3`) pseudopotentials will be discussed in class next week.

`lloc` This specifies which $l$ component is chosen as the local component of the pseudopotential.

`file_pseudopw` Name for the generated pseudopotential file. The file-extension specifies the format of the pseudopotential, i.e. how exactly the information is stored in the corresponding

file. ".UPF" stands for *unified pseudopotential format*, which is the format generally used by QuantumEspresso.

`tm` Troullier-Martins pseudization (see: Phys. Rev. B 43, 1993 (1991))

`author` Here, you can specify your name, so that people who use your pseudopotential for their calculations know who to complain to in case they obtain bad results.

In addition to specifying these variables, you also have to supply some additional data after the namelist. This data specifies which orbitals should be "pseudized" and what matching radii and reference energies should be used. The corresponding data has to be specified in the following format:

```
nwfs
nls(1)    nns(1)    lls(1)    ocs(1)    ener(1)    rcut(1)    rcutus(1)
nls(2)    nns(2)    lls(2)    ocs(2)    ener(2)    rcut(2)    rcutus(2)
...
nls(nwfs) nns(nwfs) lls(nwfs) ocs(nwfs) ener(nwfs) rcut(nwfs) rcutus(nwfs)
```

The first line specifies how many pseudo-wavefunctions should be generated (`nwfs`). The next lines specify the details for each of those pseudo-wavefunctions, line by line, i.e. `nwfs` lines altogether. The amount of space between the different items in each line is arbitrary (as long as there is some space), but each line has to contain 7 items in the correct order, with the following meaning:

`nls` Wavefunction label. Same as used in `config`, but S, P, D, ...has to be specified in capital letters!

`nns` 1 for S states, 2 for P states, ...This is the "main quantum number" of the pseudo-wavefunction, i.e. `nns-lls-1` should be equal to the number of nodes in the pseudo-wavefunction, which should generally be 0.

`lls` Angular momentum quantum number.

`ocs` Occupation (has to be the same as in the all-electron configuration).

`ener` Reference energy (in Ry) for the construction of the pseudo-wavefunction. For `ener(i) = 0.00` the corresponding eigenvalue of the all-electron orbital is used.

`rcut` Matching radius (in units of the Bohr radius $a_0 \approx 0.529$Å), beyond which all-electron and pseudo-wavefunctions should match.

`rcutus` Matching radius for ultrasoft pseudopotentials. This is not used, since we are constructing norm-conserving pseudopotentials. Simply set this equal to `rcut`.

Now add the namelist `&inputp` to your input file and specify all the additional data. Look at the graph with your all-electron wavefunctions and specify a matching radius somewhere around the position of the outermost maximum of the corresponding all-electron wavefunction. Don't forget to change `iswitch` to 3 in the `&input` namelist and run `ld1.x` again. Inspect the output file. A new file named *prefix*`ps.wfc` is generated, which contains the pseudo-wavefunctions on the radial grid. Compare the pseudo-wavefunction with the all-electron wavefunctions using `xmgrace`.

## 1.3 Transferability tests

### 1.3.1 Logarithmic derivative

As shown in class, the logarithmic derivative is defined as:

$$D^l(\epsilon, r) = \frac{1}{R^l(\epsilon, r)} \frac{\mathrm{d}R^l(\epsilon, r)}{\mathrm{d}r} \tag{1}$$

Here, $R^l(\epsilon, r)$ is the radial part of the wavefunction that is obtained by integrating the radial Schroedinger equation for some energy $\epsilon$. What is the meaning of the poles ($|D^l(\epsilon, r)| \to \infty$) and roots ($D^l(\epsilon, r) = 0$) of the logarithmic derivative?

Now calculate the energy dependence of the logarithmic derivatives for both the pseudo- and all-electron wavefunctions at a certain radius $r_{\mathrm{dl}} \geq r_c$. This radius is usually taken at half of the characteristic interatomic distance. (For crystalline Si the nearest neighbor interatomic distance is about 4.4 $a_0$.) To calculate the logarithmic derivatives edit your input file and add the following information to the `&input` namelist:

```
nld = 2,
rlderiv = 2.20,
eminld = -2.0,
emaxld = 1.0,
deld = 0.01,
```

These parameters have the following meaning:

`nld` The number of logarithmic derivatives to be calculated (= `nwfs`).

`rlderiv` Radius (in units of $a_0$) at which the logarithmic derivatives are calculated.

`eminld, emaxld` Upper and lower boundaries of the Energy range (in Ry) for which the logarithmic derivatives are calculated.

`deld` Difference (in Ry) between two adjacent energies on the numerical grid between `eminld` and `emaxld`.

In addition to all the previous files you now get two additional files containing the logarithmic derivatives: *prefix*`.dlog and` *prefix*`ps.dlog`. The first column in each file contains the energies for which the logarithmic derivatives are calculated. The remaining columns contain the logarithmic derivatives corresponding to the various wavefunctions. Plot the logarithmic derivatives using `xmgrace` and compare the energy dependence of the logarithmic derivatives of the pseudo wavefunctions with that of the all-electron functions.

### 1.3.2 Different ionic configurations and the effect of nonlinear core corrections

The program `ld1.x` can also do some simple transferability tests "on the spot": If you specify some additional configurations, different from the reference configuration, using the new namelist `&test` in your input file, the program automatically calculates the all-electron wavefunctions for these configurations and compares them with the corresponding results obtained from the previously constructed pseudopotential. To perform these test calculation change the `iswitch` parameter in the `&input` namelist to 2 and replace the `&inputp` namelist by the `&test` namelist with the following information:

```
&test
  nconf = 3,
  file_pseudo = 'Si.UPF',
  configts(1) = '3s2 3p2',
  configts(2) = '3s2 3p1',
  configts(3) = '3s2 3p0',
 /
```

The new parameters have the following meaning:

`nconf` The number of configurations to be tested.

`file_pseudo` File containing the pseudopotential.

`configts(i)` A string containing the `i`'th test valence electronic configuration (using the same syntax as for `config`), where `i = 1, 2, ..., nconf`.

Use your previously generated pseudopotential to calculate the electronic eigenvalues for various ionization states (+1, +2, +3, +4). After running the `ld.x` code you get two wavefunction files: *prefix*`i.wfc and` *prefix*`ips.wfc`, and two logarithmic derivative files: *prefix*`i.dlog and` *prefix*`ips.dlog` for each test configuration `i`. A summary of the test calculation is written to the file *prefix*`.test`, which contains the all -electron eigenvalues for the different configurations compared with the corresponding eigenvalues calculated from the pseudopotential. For a "good" pseudopotential, the all-electron eigenvalues should be reproduced to within a few mRy for all considered configurations. See how your pseudopotential performs in this test.

Make sure that you check the $Si^{4+}$ configuration, which is important if you want to use your pseudopotential to calculate the properties of $SiO_2$. You will see that the difference in the eigenvalues between the all-electron and pseudo-wavefunctions is unacceptably large in this case.

Compare the all-electron wavefunctions for this configuration with the ones for the reference configuration. What do you see? Do you think that the "frozen core" treatment of the pseudopotential method is justified? Describe the difference in the $3s$ and $3p$ valence functions between the two different configurations and explain the origin of this difference.

You might realize that there is quite some overlap between the $3s/3p$ valence functions and the $2s/2p$ core state. Generate a new pseudopotential using *nonlinear core corrections*. You do this by adding `nlcc = .true.` to the namelist `&inputp`. Repeat the test calculation for different configurations, including +4, and explain the observed effect of the nonlinear core corrections.

### 1.3.3 Effect of different matching radii

After this, generate two additional sets of pseudo-wavefunction and the corresponding pseudopotentials using two different matching radii: first, a matching radius that is clearly on the lower side of the outermost maximum of the corresponding all-electron wavefunction, and then a matching radius that is significantly larger than that. Don't forget to save all the wavefunction and pseudopotential files from your previous calculations.

Now compare the pseudo-wavefunctions corresponding to the different matching radii. Also compare the corresponding logarithmic derivatives (but make sure that this time you calculate all logarithmic derivatives for a radius larger than your largest matching radius!). How does the choice of the matching radius affect the transferability of the resulting pseudopotential? Can this be seen in the energy dependence of the logarithmic derivatives?

### 1.3.4 Estimate of required plane-wave cutoff

`ld1.x` also determines an estimate for the required energy cutoff when the pseudo wavefunctions are expanded in a plane wave basis. This is done using spherical Bessel functions $j_l(kr)$, which are closely related to plane waves. They are the radial coefficients in the expansion of a plane wave in terms of spherical harmonics ($P_l(\cos\theta)$ are the Legendre polynomials):

$$e^{i\vec{k}\cdot\vec{r}} = \sum_{l=0}^{\infty}(2l+1)i^l P_l(\cos\theta)j_l(kr) \tag{2}$$

and can be expressed as:

$$j_l(z) = (-z)^l \left(\frac{1}{z}\frac{\mathrm{d}}{\mathrm{d}z}\right)^l \frac{\sin z}{z} \tag{3}$$

This test can be done by adding the following parameters to the namelist `&test`:

```
ecutmin = 15.0,
ecutmax = 100.0,
decut = 5.0,
```

These parameters have the following meaning:

`ecutmin, ecutmax, decut` The pseudo-Hamiltonian for fixed SCF-potential is diagonalized in the basis of spherical Bessel functions for various values of ecut: `ecutmin`, `ecutmin+decut`, `ecutmin+2*decut`, ... up to `ecutmax`.

Use this feature to get an estimate for the required plane wave cutoff energies for the different pseudopotentials you generated. Plot the eigenvalues as a function of the cutoff energy. How does the choice of $r_c$ alter the required plane-wave cutoff?

## 2 Norm-conserving pseudopotential for O

Now try to construct a norm-conserving pseudopotential for the O atom. See how the estimate of the required plane wave cutoff energy depends on the choice of the matching radius and compare this with the previous case of Si. Why is it impossible to obtain a norm-conserving pseudopotential for O with comparable transferability and softness as for Si?