

HOMEWORK 8 (Optional)
Creating Pseudopotentials
Due March 16, 2009

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 us to expand the resulting pseudo-wavefunctions using a much smaller number of plane wave basis functions. 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 some radius r_c . Specifically, in this exercise we will be creating a norm-conserving semi-local pseudopotential for Si. There are 3 main steps for pseudopotential generation.

1. Calculate the all-electron wavefunctions for a chosen reference system. 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 chosen r_c . From these pseudo-wavefunctions the pseudopotential is then obtained by inverting the radial Schrodinger 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.

For this exercise you will use `ld1.x` which is available as part of the quantum espresso package. Using linux or a cygwin framework with Windows it should be relatively easy to compile this executable. Running `ld1.x` is done in the same manner we are used to with `pw.x`:

```
ld1.x <input.txt> output.txt
```

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 a reference configuration for the pseudopotential generation. You will then perform an all-electron calculation with the LDA approximation for a single Si atom in this configuration.

For this, you need to write an input file for `ld1.x`. The details for the format for `ld1.x` input is given at this link: http://www.democritos.it:8888/O-sesame/fileview?f=O-sesame/atomic_doc/INPUT_LD1&v=1.47.

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',
/
```

You can see that the format is similar to pw.x input which you are familiar with. The `&input` namelist is the only necessary namelist.

title is a string variable identifying your calculation. This is not used in any actual calculations, and is only for your convenience.

Zed is the nuclear charge of the atom you want to calculate (for Si this is 14).

Iswitch is an integer value indicating the what type of calculation you want to do (1 for all electron calculation, 2 for pseudopotential tests, and 3 for pseudopotential generation).

Dft This specifies what approximation for the exchange-correlation energy functional you want to use. We are most familiar with LDA and will be using that in this exercise. You should be aware that there are many more options though (of them, you are familiar also with the PZ parameterization, which is also an option.).

Prefix is a string that is automatically appended to output files that will be generated. Using a descriptive prefix will make your life easier when you are trying to differentiate between many output files you have lying around.

Config is the electronic reference configuration for the all-electron calculation. [Ne] is an abbreviation for the noble gas configuration of neon.

Now, write an input file with the above specifications and run `ld1.x`. Inspect the output files and the generated files created. You can find the eigenvalues of the different all-electron functions in three different units (Ryd, hartree, and eV). In addition, the total energy, and the various contributions to the total energy are printed out. As you can see, the exchange correlation is the smallest contribution to the total energy.

There will be a file of type `.wfc`. It contains various columns of data. The first is `r`, the radial grid used in the calculation. The following columns contain the wavefunctions at corresponding radii.

If you are using linux you can use `xmgrace` to plot this data using the command (you need to install `grace` first):

```
xmgrace -nxy prefix.wfc
```

In windows you can use excel to plot.

1.2 Construction of smooth pseudo-wavefunctions

Next you have to construct smooth pseudo-wavefunctions corresponding to the valence shells from your previous calculations (3s and 3p in our case. To do this we have to add a new namelist, `&inputp`, specifying the parameters for our pseudopotential generation. An example is given below.

```
&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
```

Pseudotype selects the type of pseudopotential that will be generated. 1 is semi-local norm-conserving. 2 is separable norm conserving, and 3 is ultrasoft.

Lloc specifies which l component is chosen as the local component of the pseudopotential.

File_pseudopw gives the filename for the generated pseudopotential file. You should recognize .UPF as the standard PP filetype you have been using with quantum espresso. It stands for unified pseudopotential format

Tm is Troullier-Martins pseudization (for more information see Phys. Rev. B 43, 1993 (1991)).

Author is to specify your name, so others who use your generated pseudopotentials know who to complain to or give praise when they use it.

The next few lines gives the orbitals which should be ‘pseudized’ and what matching radii and reference energies should be used. It is given 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)
```

Nwfs is the number of wavefunctions you will be making into pseudopotentials

Nls is the wavefunction label. Same as what is used in typical electronic configurations. Make sure S, P, D are capitalized.

Nns is 1 for s states, 2 for p states, etc.

Lls is the angular momentum quantum number. $Nns-ls-1$ should be equal to the number of nodes in the pseudo-wavefunction (generally 0)

Ocs is the occupation (should be the same as in the all electron configuration)

Ener is the reference energy in Rydberg. For $ener(i)=0.00$ the corresponding eigenvalue of the all-electron orbital is used.

Rcut is the cutoff radius, r_c , beyond which all-electron and pseudo-wavefunctions should match

Rcutus is used for ultrasoft pseudopotentials.

Now, add the namelist %inputpp to your input file and specify all the needed 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. Iswitch in the &input namelist must be changed to 3 since we are generating the PP and we are not interested in doing the all electron calculation again! Compare the pseudo-wavefunctions with the all electron wavefunctions by graphing them and then commenting.

1.3 Transferability Tests

1.3.1 Logarithmic Derivative

The logarithmic derivative is defined as:

$$D^l(\epsilon, r) = \frac{1}{R^l(\epsilon, r)} \frac{dR^l(\epsilon, r)}{dr}$$

Where R is the radial part of the wavefunction that is obtained by integrating the radial Schrodinger equation for some energy epsilon. What is the meaning of the poles ($D \rightarrow \infty$) and roots ($D \rightarrow 0$) of the logarithmic derivative?

Now calculate the energy dependence of the logarithmic derivatives for both the pseudo and all electron wavefunctions as a certain radius $r_{dl} > r_c$. This radius is usually taken at half of the characteristic interatomic distance. To calculate the logarithmic derivatives edit your input file and add the following information the &input namelist:

```

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

```

Nld is the number of logarithmic derivatives to be calculated (=nwfs)

Rlderiv is the radius in a_0 at which the logarithmic derivatives are calculated

Eminld, emaxld are the upper and lower boundaries of the energy range (in Rydberg) for which the logarithmic derivatives are calculated

Deld is the differences in Rydberg between two adjacent energies on the numerical grid between **eminld** and **emaxld**.

When you run this, you will get two additional files containing the logarithmic derivatives called `prefix.dlog` and `prixps.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 and compare the energy dependence of the logarithmic derivatives of the pseudo wavefunctions with that of the all electron wavefunctions.

1.3.2 Different ionic configurations and the effect of nonlinear core corrections

The program `ld1.x` can also do simple transferability tests on the spot if you specify some addition configurations using the namelist `&test` in your input file. The program will automatically calculate the all-electron wavefunction and compare them with the corresponding results obtained from the previously constructed pseudopotential. To perform these test calculation change the `iswitch` parameter I 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',
/

```

Nconf is the number of configurations to be tested

File_pseudo is the pseudopotential you previously generated

Configt*x*(i) is a string containing the test valence electronic configuration

Use the previously generated pseudopotential to calculate the electronic eigenvalues for various ionization states (+1,+2, +3, +4). After running `ld1.x` code you get two

wavefunction files: prefixi.wfc and prefixips.wfc , and two logarithmic derivative files: prefixi.dlog and prefixips.dlog for each test configuration i. A summary of the test calculation is written to the file prefix.test, which contains the all electron and pseudopotential eigenvalues for the different configurations. 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 to check the Si^{4+} configuration as this is important for the properties of SiO_2 . How well does this work?

Compare the all electron wavefunctions for this configuration with the ones for the reference configurations. 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. Read about nonlinear core correction [at this link](#). Generate a new pseudopotential using nonlinear core corrections. You do this by adding “nlcc = .true.” to the namelist &inputp. Repeat the test calculations for different configurations, including +4, and explain the observed effect of the nonlinear core corrections.

1.3.3 Effect of different cutoff radius

After this, generate two additional sets of pseudo-wavefunctions and the corresponding pseudopotentials using two different cutoff radii. First, a cutoff that is clearly on the lower side of the outermost maximum of the corresponding all electron wavefunction, and then a cutoff which 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 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 potential. Can this be seen in the energy dependence of the logarithmic derivatives?

1.3.4 Estimate of the required plane wave cutoff

The program `ld1.x` can also determine an estimate for the required energy cutoff when the pseudo wavefunctions are expanded in a plane wave basis (this is important information for when you want to use your .UPF with `pw.x`). This is done using a spherical Bessel function $j_l(k,r)$, which are closely related to plane waves. They are the radial coefficients in the expansion of a plane wave in terms of spherical harmonics:

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

And can be expressed as:

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

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

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

Where the parameters give the values of ecut where the pseudo-Hamiltonian for fixed SCF-potential is diagonalized in the basis of spherical Bessel functions. The values of ecut are given as ecutmin, ecutmin+decut, ecutmin+2decut, ..., 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?

For this exercise, be careful to keep track of input and output files (try not to overwrite previous work!) Go through the example step by step, and hand in plots with explanations and an overview of what you are doing. This exercise is an important part of research using DFT, and going through this will give a sense of the amount of work, consideration, and justification, it takes to make a useful pseudopotential.

If you want additional practice, try constructing a norm-conserving pseudopotential for an oxygen atom. See how the estimate of the required plane wave cutoff energy depends on the choice of cutoff 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?