ATOMPAW 'Explore' Function User Manual

Cameron M. Kates
katecm0@wfu.edu
Wake Forest University, Winston-Salem, NC 27109 – USA

Dr. N. A. W. Holzwarth
Wake Forest University, Winston-Salem, NC 27109 – USA

Last Edited : 8/4/2013
Using ATOMPAW 4.0.0.0 and ParameterExplore 1.6

Table of Contents

An Introduction to the *ATOMPAW* 'Explore' Function

The ATOMPAW software has added as part of its functionality a function titled 'Explore.' The Explore function allows for the ability to scan across a wide range of input parameters, meaning that data for many different pseudopotentials functions can analyzed using only one run of the ATOMPAW software. This allows us to easily compare the results for several different radii, basis energies, and matching radii all at once, helpfully increasing the speed of analysis.

The Explore function is rather simple to call. After the initial pseudo function parameters are entered in the standard ATOMPAW 'in' file, the user indicates to the program that they wish to explore several extra datasets by adding the tag 'EXPLORE' to the last line of the code, as shown in the example file displayed in Figure 1 below.

Figure 1: Sample of an ATOMPAW 'in' file calling the Explore function.

```
Co 27
LDA-PW loggrid 2001
4 3 3 0 0 0
3 2 7
0 0 0
c
c
v
v
c
v
v
2                      //Initial pseudo function
2.0 1.8 2.0 2.0
n
y
2
n
y
2
n
MODRRKJ VANDERBILTORTHO Besselshape
3 0    MTROULLIER
2.0
2.0
2.0
2.0
2.0
2.0
EXPLORE                //Explore function tag
```

Note, however, that this is all our file should contain initially. There are currently no pseudo functions

to be explored. It is possible to add these functions manually after the EXPLORE tag by repeatedly typing in the new pseudo function parameters in the same format as the initial pseudo function shown above. However, this method is prohibitively time consuming when a higher Explore range is desired. For this reason, a script known as *ParameterExplore* was created to automate this process.

An Introduction to the *ParameterExplore* Script

The *ParameterExplore* program is a simple C++ script which allows the user to input a range of values for radius Rc, binding energy, matching radius, and log derivative calculation range. It then takes advantage of the UNIX command 'sed' to generate and append pseudo functions representing every possible value contained within the given ranges. Source code for *ParameterExplore* is available online at WEBPAGE. *ParameterExplore* comes with no warranty whatsoever, explicitly or implied. The *ParameterExplore* script works without error on the machines of its host institution, Wake Forest University, but it is not guaranteed that it will work on any other computational set up. However, it is likely that it will function properly on any standard UNIX style terminal complete with the 'sed' functions and the like. A ReadMe file containing update reports as well as known bugs is available online at WEBPAGE.

Creating A Complete 'Explore' File

Creating a complete 'Explore' file for *ATOMPAW* requires several relatively simple steps, utilizing both the *ATOMPAW* and *ParameterExplore* software.

1. **Generating the initial 'in' file including the EXPLORE tag.**
   This step is very similar to creating any other input file for *ATOMPAW*. In fact, the only main difference is that the EXPLORE tag is the sole tag placed at the end of the file, as shown in *Figure 1* above. An in depth analysis of how to create the 'in' file up until this point has been created by Marc Torrent of the Commissariat à l'Energie Atomique et aux Energies Alternatives , France. The 'in' file should be created in the directory in which you wish to run your trial.

2. **Generate the *ParameterExplore* 'PseudoTemplate' file.**
   The *ParameterExplore* script requires that there be a 'PseudoTemplate' file present in the same directory as the 'in' file. The 'PseudoTemplate' file must be of the form of the initial pseudo function in your 'in' file and serves as a template foir the pseudofunctions to be explored. An example of this portion of the 'in' file is highlighted in *Figure 1* above. However, the 'PseudoTemplate' file will contain no numbers. Instead, it will be composed entirely of variables which will be recognized by the *ParameterExplore script.* An example of the 'PseudoTemplate' created in conjunction with the sample 'in' file picture in *Figure 1* above is shown in *Figure 2* below.

Figure 2: Sample 'PseudoTemplate' file, corresponding with the 'in' file shown above.

```
2
Rin    Rsmall   Rin     Rin
n
y
En
n
y
En
n
MODRRKJ VANDERBILTORTHO  Besselshape
3 0   MTROULLIER
Rmatch
Rin
Rmatch
Rin
Rmatch
Rin
```

Note how the variables in the 'PseudoTemplate' file match the position of the corresponding values in the 'in' file. However, the naming convention is slightly atypical. Rin represents the radius Rc, Rsmall is defined as Rsmall = (Rin -0.2), En is the basis energy (where En > 0.0), and Rmatch is the matching radius. The *ParameterExplore* script will essentially repeatedly fill in these variable values with every possible value in the range specified by the user, then append them to the end of the 'in' file. If a number is entered in place of a variable in the PseudoTemplate file, that particular parameter will remain fixed at the value entered Each of these 'PseudoTemplate' bits of text added to the end of the 'in' file represents a new pseudo function to be explored by *ATOMPAW*. Additionally, the keywords *MODRRKJ*, *VANDERBILTORTHO*, etc. are not the only ones possible. See the full *ATOMPAW* user manual by Marc Torrent for the other possible keyword choices.

3. Running *ParameterExplore*

After both the 'in' file and 'PseudoTemplate' file have been created in the same directory, it is time to run the *ParameterExplore* program. Note that the program must be called in the same directory as the 'in' and 'PseudoTemplate' files. *ParameterExplore* will create its output files here as well.

The executable version of the *ParameterExplore* program is intended to be run interactively. A short title screen and the first available input parameter will be displayed, as shown in *Figure 3* below.
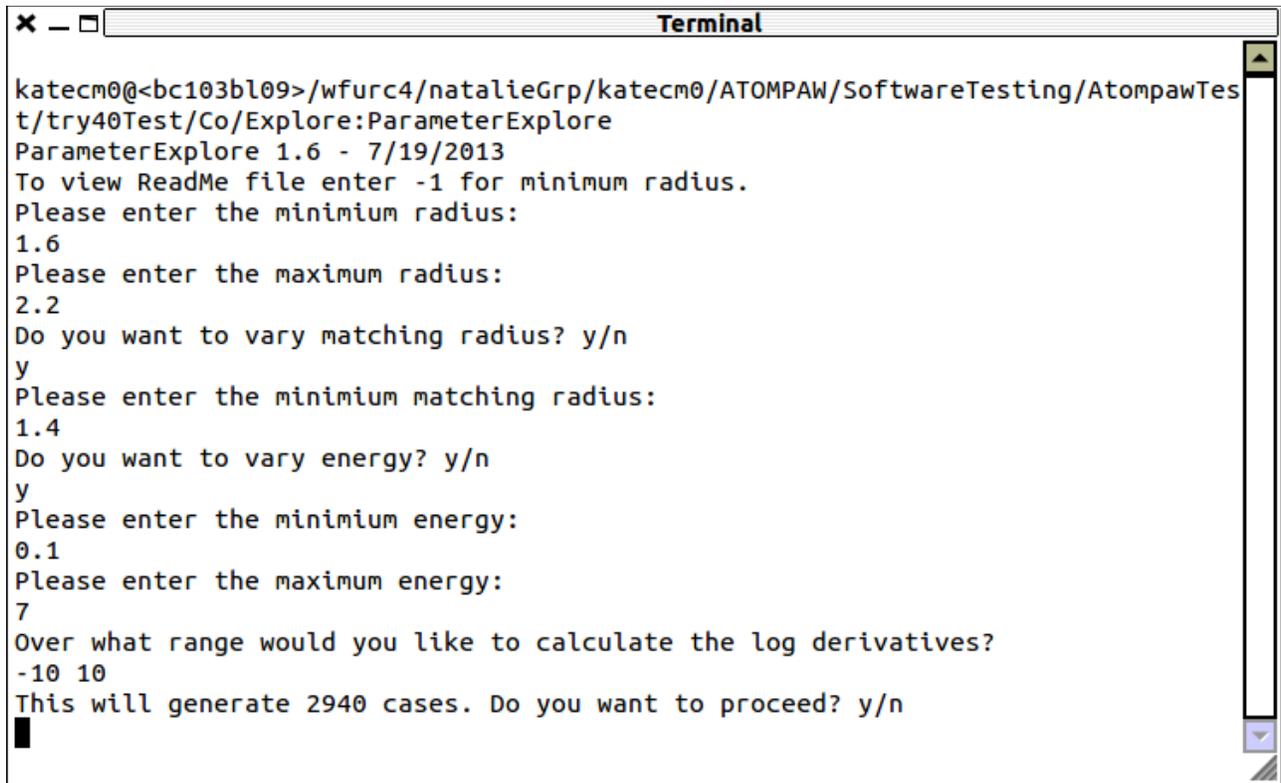
Figure 3: ParameterExplore Startup screen.

```
katecm0@<bc103bl09>/wfurc4/natalieGrp/katecm0/ATOMPAW/Elements/Al/changeinputScr
ParameterExplore 1.6 - 7/19/2013
To view ReadMe file enter -1 for minimum radius.
Please enter the minimium radius:
```

Note here that if you wish to view the *ParameterExplore* program ReadMe file, you can enter -1 for the first parameter, minimum radius.

ParameterExplore will ask the user for a series of input parameters, including minimum and maximum radii to vary between, the minimum matching radius (this is typically 0.2 less than the minimum radius specified by the user, but another value could be entered by the user if need be), minimum and maximum energies, and the range of log derivative values over which we wish to calculate. Additionally, *ParameterExplore* will give the user the choice to vary or not vary the matching radius and energy. In almost all cases it is best to vary these parameters. It is only in special cases that these options should be fixed.

Figure 4: Sample input for ParameterExplore.

```
╳ — □                              Terminal
katecm0@<bc103bl09>/wfurc4/natalieGrp/katecm0/ATOMPAW/SoftwareTesting/AtompawTes
t/try40Test/Co/Explore:ParameterExplore
ParameterExplore 1.6 - 7/19/2013
To view ReadMe file enter -1 for minimum radius.
Please enter the minimium radius:
1.6
Please enter the maximum radius:
2.2
Do you want to vary matching radius? y/n
y
Please enter the minimium matching radius:
1.4
Do you want to vary energy? y/n
y
Please enter the minimium energy:
0.1
Please enter the maximum energy:
7
Over what range would you like to calculate the log derivatives?
-10 10
This will generate 2940 cases. Do you want to proceed? y/n
▮
```

Note that after the user has entered the parameters they wish to vary, *ParameterExplore* displays the number of cases that will be generated and asks the user for a continuation confirmation. The cases mentioned are the number of different pseudo functions which will be generated. In general, it is best to keep the number of cases below 4000 to prevent excessively long computation times and large files. However, if computational power and file storage is not an issue, there is no harm in generating more cases.

Once the user prompts *ParameterExplore* to continue, it will generate an executable file in the background, which will then modify your prepared 'in' file. When complete, *ParameterExplore* will produce three files: the modified 'in' file, the changeInputRadius .exe file, and the InputRecords .txt file. The changeInputRadius.exe file should be ignored, but the InputRecords.txt can prove to be useful. This file keeps a record of the date this run of *ParameterExplore* was executed and the parameters entered by the user. This can be a helpful reference when attempting future trials.

4. **The Modified 'in' file.**
   After being modified by *ParameterExplore*, the 'in' file will likely be quite long, possibly upwards of 50,000 lines. There is no problem opening the 'in' file to inspect it after *ParameterExplore* has run, but typically this is not needed, and your text editor may be slow to respond due to the large size. Still, the 'in' file will now be of the form shown in *Figure 5 below.*

Figure 5: The appended 'in' file, including two new pseudo functions.

```
Co 27
LDA-PW loggrid 2001
4 3 3 0 0 0
3 2 7
0 0 0
c
c
v
v
c
v
v
2
2.0 1.8 2.0 2.0
n
y
2
n
y
2
n
MODRRKJ VANDERBILTORTHO Besselshape
3 0    MTROULLIER
2.0
2.0
2.0
2.0
2.0
2.0
EXPLORE              //Explore function tag
2940 7    LOGDERIVERANGE -10  10 //Initial ATOMPAW commands
2                    //New pseudo function 1
1.6   1.4   1.6    1.6
n
y
0.1
n
y
0.1
n
MODRRKJ VANDERBILTORTHO  Besselshape
3 0    MTROULLIER
1.4
1.6
1.4
1.6
1.4
```

```
1.6
2                      //New pseudo function 2
1.6   1.4   1.6     1.6
n
y
0.2
n
y
0.2
n
MODRRKJ VANDERBILTORTHO  Besselshape
3 0    MTROULLIER
1.4
1.6
1.4
1.6
1.4
1.6
                       //New pseudo functions continue
```

At this point, the *ATOMPAW* 'in' file is ready to be run by your regular means.

Analyzing the *ATOMPAW* Explore Output

As usual, *ATOMPAW* will output many files, including the EXPLORESUMMARY
EXPLORERESULTS, and usual 'out' files.

This file, however, will appear differently than it would had the Explore function not been used. The
Explore 'out' file contains a large amount of data, but the information we are most concerned with is at
the very end of the file, and can be easily accessed on its own using the commands:

: more EXPLORERESULTS

or

: more EXPLORESUMMARY

These results are shown below in *Figure 6* as displayed by EXPLORESUMMARY.

Figure 6: ATOMPAW Explore 'EXPLORESUMMARY' file from the sample dataset shown in Figure 5 above.

```
X — □                                    Terminal
 Logderiv errors based on energy range     -10.00        10.00
 Results for minimum logderiverror
=== Rc =                1.615605 ====
 l =     0       71  1.9705538E+01
 l =     1       31  4.5572809E-01
 l =     2       66  3.0606608E-01
 l =     3        1  2.9934471E+00
=== Rc =                1.712495 ====
 l =     0      281  1.3790137E+01
 l =     1      305  3.3885860E-01
 l =     2      338  4.8122558E-01
 l =     3      211  3.4475837E+00
=== Rc =                1.815195 ====
 l =     0      631  1.5028391E+01
 l =     1      719  3.3727139E-01
 l =     2      691  1.0171331E+01
 l =     3      491  2.8540337E+01
=== Rc =                1.905468 ====
 l =     0      841  8.9968449E-01
 l =     1     1137  2.0264873E-01
 l =     2     1113  9.9338237E+00
 l =     3      841  2.6225749E+01
=== Rc =                2.000231 ====
 l =     0     1261  9.6732598E+00
 l =     1     1557  1.2077969E+00
 l =     2     1597  1.5048818E+01
 l =     3     1261  2.7463601E+01
=== Rc =                2.120187 ====
 l =     0     1751  6.8951593E+01
 l =     1     2191  6.1724357E+00
 l =     2     2149  1.5262110E+01
 l =     3     1751  2.7546448E+01
=== Rc =                2.204129 ====
 l =     0     2370  9.4944927E+01
 l =     1     2750  2.1110602E+01
 l =     2     2776  2.1818289E+01
 l =     3     2311  2.9249297E+01
~
(END)
```

In this version of the *ATOMPAW* Explore function the log derivative errors are organized by case (which pseudo functions they represent) and radius, Rc. Although there is no one set way to determine which radius best represents the atom, there is a general process on which one can base their analysis. It is as follows:

1. **Determine the radius with the lowest log derivative error.**

   This is relatively simple to start with, as typically one can just pick the radius that, on average, has the lowest errors. Typically this is an order of magnitude analysis, so the individual numbers aren't of the greatest importance. Instead, a "good" log derivative error would be on the order of about 10^{-01} or 10^{-02} for a small core analysis. However, if the atom is being analyzed as a large core structure, or if the log derivative energy range is large, these values may increase and still

be acceptable.

2. Analyzing log derivative plots.
ATOMPAW will not only produce a numerical representation of the log derivative errors, but graphical plots as well. To examine these plots, enter the command:

```
:explorelogderiv l_value case#
```

Here we're telling the terminal to display a file which plots the log derivative errors for a particular case, which corresponds to an *l* value (*l=0,1,2,3, etc.).* For example, see in *Figure 7* below the log derivative plot for case 281 in the sample output listed above. This case represents the *l=0* case for the radius Rc=1.7.

Figure 7: Sample log derivative plot.
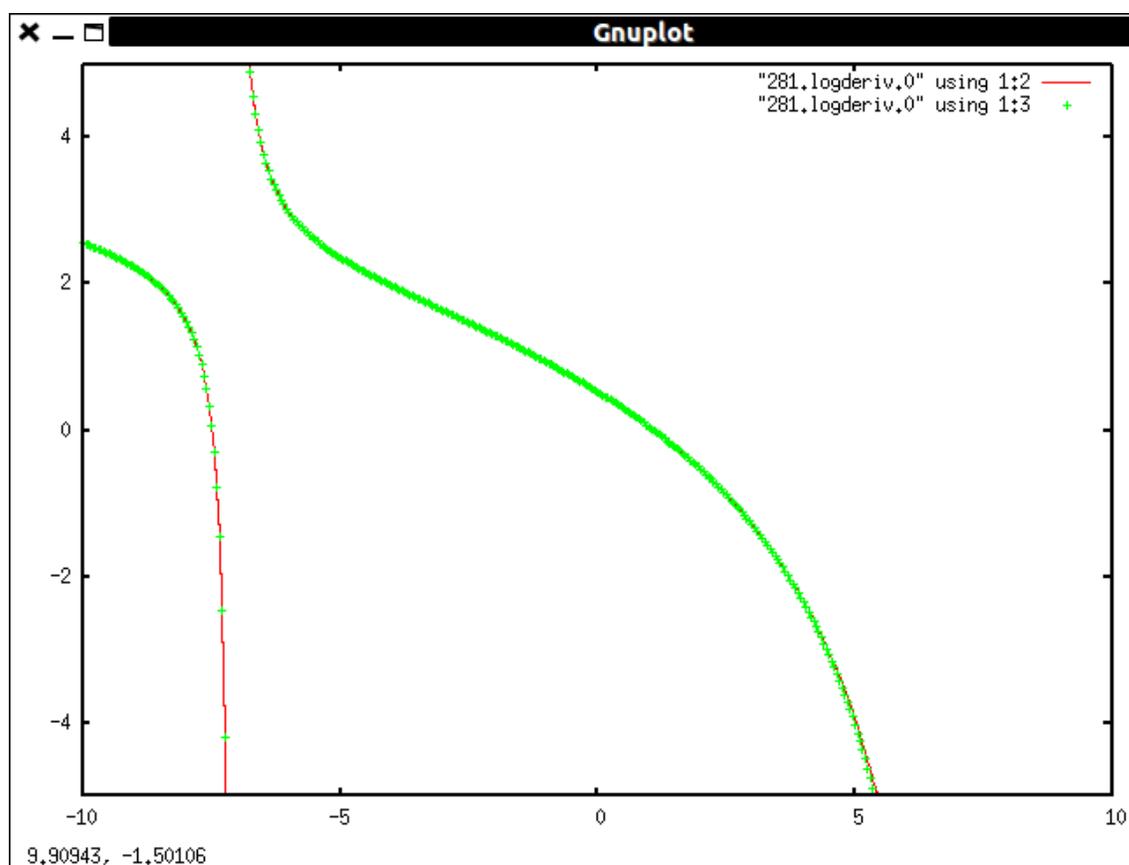
Note that the 'explorelogderiv' command built into the *ATOMPAW* Explore function automatically calls Gnuplot for us. If you look closely at this plot, you'll notice that there are two lines here, which match one another quite closely. This is a good log derivative plot. If the lines diverge from one another, the log derivative errors are likely not the best. However, it should be noted that this type of curve is not the only acceptable form for the log derivative plot. As long as the lines of the plot are still closely fit to one another, the plots are considered good. A discussion of other possible issues to be found in the log derivative plots can be found

in the "Other Details" portion of this manual.

3. Analyzing the wave function plots.

In addition to the log derivative plots, we have the multiple wave function plots to help us determine which radii and cases best represent our atom. To access these plots, we enter the command:

```
: explorewfn wavefunction# case#
```

Here, we're telling the terminal to display a file which plots the wave function specified (wave function 1, 2, 3, etc.) for a particular case. For example, see in *Figure 9* below the wave function 1 plot for case 281 from the sample output listed above.

Figure 9: Wave function 1 plot for case 281.

2.63051, 0.641121
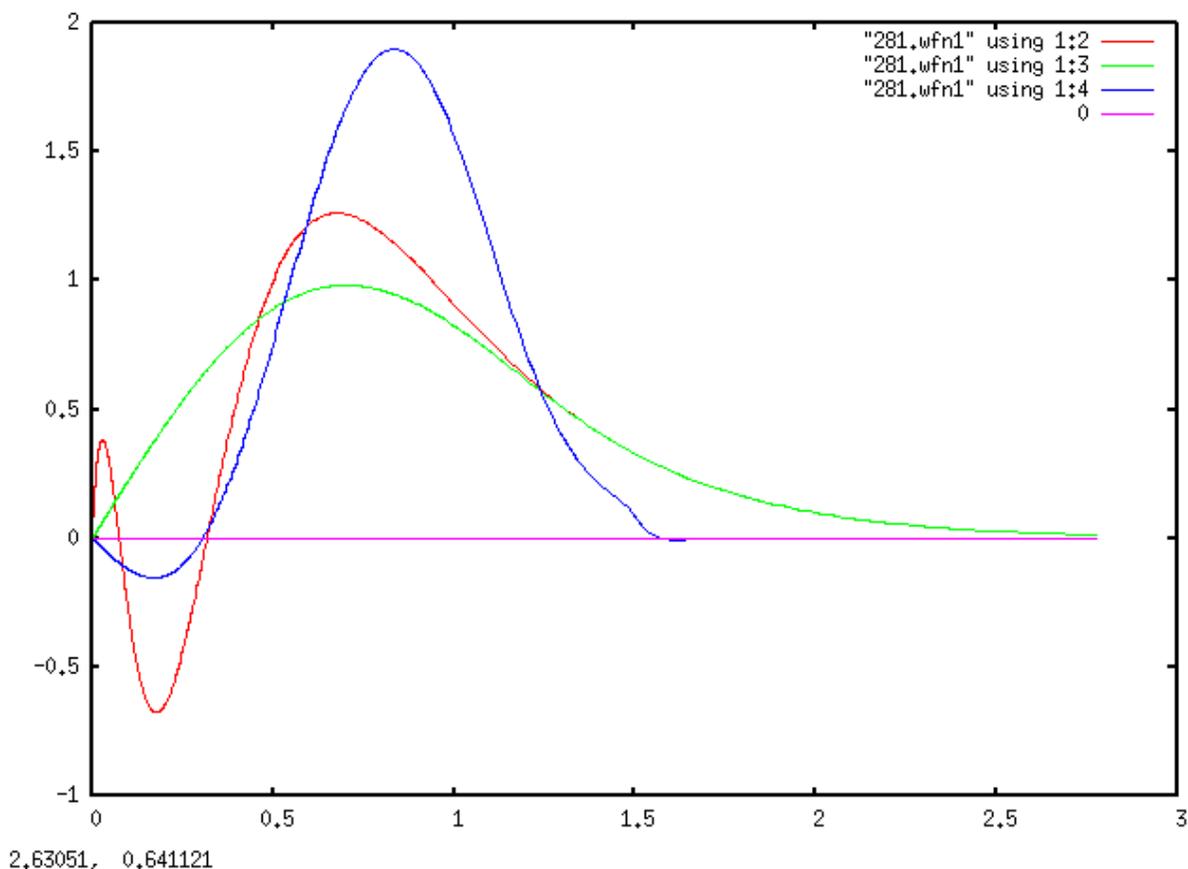
Note that we have three wave functions plotted here. In red we have the all electron function, in green the pseudo function, and the projector wave function in blue. Although all three of these are important, the projector wave function can often provide hints as to if the case is accurate. For the most part, a projector wave function that does not have an excessively large amplitude and is relatively smooth is preferred.

There is some degree of experience which is required to effectively analyze the log derivative and wave function plots produced by *ATOMPAW,* as there is no one plot form that is best or consistently correct.

As with the log derivative plots, further discussion on the wave function plots is included in section five of this manual, "Other Details." Once the most accurate radius and associated cases has been selected, the EXPLOREIN.# files produced by *ATOMPAW* with the Explore function enabled can be viewed to determine the individual matching radii and energies for each case and each *l* value. To access the EXPLOREIN files for a particular case, enter the following command:

```
: more EXPLOREIN.case#
```

This will display the information for the pseudo function represented by that particular case just as it was added to the end of the 'in' file. This information can then be used to generate an input file for the *Abinit* modeling and analysis software.

The EXPLORESUMMARY file lists the smallest logarithmic derivative errors for each augmentation radius included in the scan.   Since the logarithmic derivative error is not the only factor in determining a robust dataset, it is some times necessary to scan the EXPORERESULTS file which lists all of the results for all cases that were tried.

Figure 10:   Sample EXPLORESUMMARY file listing case #, rc (bohr), and logarithmic derivative errors for l=0,1,2,...

```
⊗⊖⊡   bc103bl06.deac.wfu.edu
#    dataset    rc        0              1              2
     1       1.60662  7.5329532E+01  3.3124126E+01  7.1043982E+00
     2       1.60662  7.5329532E+01  3.2152248E+01  7.1043982E+00
     3       1.60662  7.5329532E+01  3.1228246E+01  7.1043982E+00
     4       1.60662  7.5329532E+01  2.7262408E+01  7.1043982E+00
     5       1.60662  7.5329532E+01  2.6413175E+01  7.1043982E+00
     6       1.60662  7.5329532E+01  2.5596535E+01  7.1043982E+00
     7       1.60662  7.5329532E+01  2.4808546E+01  7.1043982E+00
     8       1.60662  7.5329532E+01  2.0947187E+01  7.1043982E+00
     9       1.60662  7.5329532E+01  2.0203108E+01  7.1043982E+00
    10       1.60662  7.5329532E+01  1.9478524E+01  7.1043982E+00
    11       1.60662  7.5329532E+01  1.8770974E+01  7.1043982E+00
    12       1.60662  7.5329532E+01  1.8078254E+01  7.1043982E+00
    13       1.60662  7.5329532E+01  1.7398379E+01  7.1043982E+00
    14       1.60662  7.5329532E+01  1.3613067E+01  7.1043982E+00
    15       1.60662  7.5329532E+01  1.2951005E+01  7.1043982E+00
    16       1.60662  7.5329532E+01  1.2296997E+01  7.1043982E+00
    17       1.60662  7.5329532E+01  1.1649707E+01  7.1043982E+00
    18       1.60662  7.5329532E+01  1.1007921E+01  7.1043982E+00
    19       1.60662  7.5329532E+01  1.0370544E+01  7.1043982E+00
    20       1.60662  7.5329532E+01  9.7365869E+00  7.1043982E+00
    21       1.60662  7.5329532E+01  9.1051735E+00  7.1043982E+00
    22       1.60662  7.5329532E+01  5.3400362E+00  7.1043982E+00
EXPLORERESULTS
```

For parameter values which include the wrong number of nodes within rc, the logarithmic derivative error is set at 9E20.   It is sometimes possible to construct robust datasets with better projector shapes that have logarithmic derivatives errors that are somewhat larger than the minimum value.

Other Details

Although the fidelity of the logarithmic derivatives of the pseudo wave functions in comparison with their all-electron counterparts is a necessary condition on developing a reliable PAW dataset, it is by no means sufficient. Some counter examples and their analysis are given here.

1. **Ghost state solutions**
   When the pseudo equations develop a unphysical ("ghost") solution, typically at low energy [X. Gonze, R. Stumpf, and M. Scheffler, Phys. Rev. **44**, 8503 (1991)], it becomes impossible to determine carry out the self-consistent calculations. Ideally, a PAW dataset would be constructed in such a way that such ghost state solutions do not haunt the simulations. The logarithmic derivatives of the ground state configuration of each atom as a function of energy provide a rough indication of the likelihood of unphysical bound state solutions as shown in the example below:

   Figure 11: Plot of logarithmic derivative of l=0 wavefunctions of Cs, comparing all-electron functions (red) with pseudo functions (green) as a function of energy in Ry. The logarithmic derivatives are evaluated at the augmentation radius and the spikes in the functions occur when the wavefunction passes through 0 at that radius. The spike in the all-electron function at E=-16 Ry is correlated with the 4s bound state which is not represented by the corresponding pseudo function as expected. However the spike in the pseudo function at E=-7 Ry is indicative of an unexpected node and possible ghost state in the pseudo function.
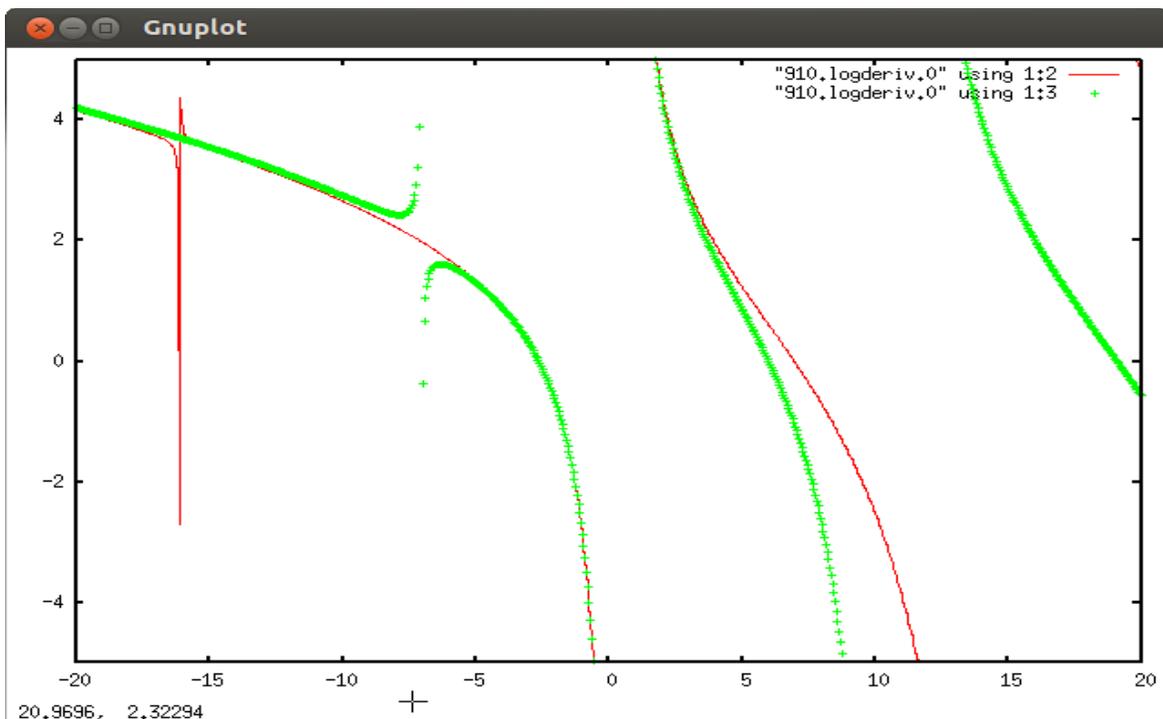

2. **Poorly scaled projector functions**
   In the VANDERBILTORTHO and related methods of generating the projector functions, it is sometimes the case that the derived projector function $p_l(r)$ is either very large or very small

compared with the scale of the all-electron and pseudo wavefunctions, which can cause instabilities in the calculations. In the current version of the EXPLORE subroutine, this problem is not indicated in the figure of merit (namely the sum of the magnitudes of the logarithmic derivative errors of each l-channel), but the problem can be detected by examining the plots of the all-electron and pseudo wavefunctions together with the projector functions such as shown in the example below:

Figure 12: Plot of 5s radial wavefunction for Cs, comparing all-electron function (red, scaled by 10x), pseudo function (green, scaled by 10x), and corresponding projector function (blue). This set of basis and projector functions is far from ideal.
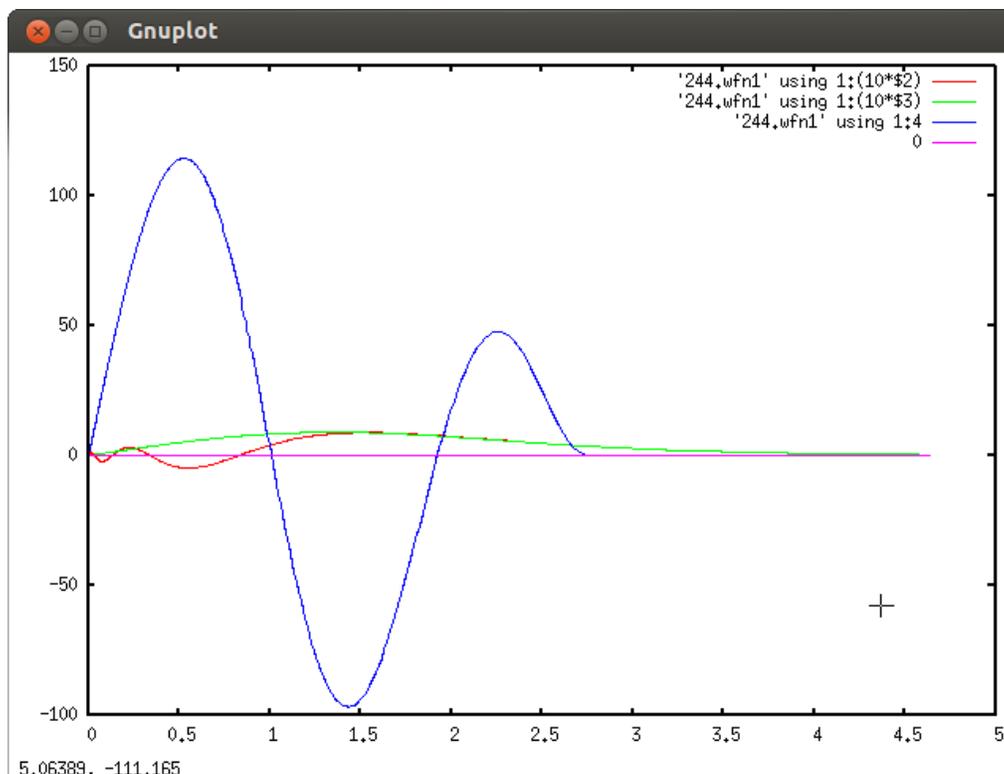
3. General comments
The zero order requirement for constructing a PAW dataset is that there be no ghost states generated during the solid state calculations.   Apart from rough indications from discontinuities in the logarithmic derivative curves versus energy, the only way we know to determine whether a dataset is ghost free, is to test the dataset in the generation of binding energy curves of simple materials containing the element represented by the dataset.   If the scf calculations using that dataset in *abinit* or *quantum-espresso* run does not converge or converges very slowly, that is a good indication of ghost states.   On the other hand, in the ghost-free parameter windows, we find that there is often a wide range of parameters that can generate good datasets. In our experience, there are several "windows" of parameter choices that can be used for each element.

In order to optimize a dataset in terms of its ability to reproduce the all-electron results, it may also be necessary to consider physical and/or chemical properties of the material such as the inclusion of semi-core states, or high angular momentum states in the basis.