PostProc User's Guide (v.6.1)

Contents

1 Introduction

This guide covers the usage of PostProc, version 6.1: an open-source package for postprocessing of data produced by PWscf and CP. PostProc is part of the QUANTUM ESPRESSO distribution and requires PWscf to be installed.

This guide assumes that you know the physics that PostProc describes and the methods it implements. It also assumes that you have already installed, or know how to install, QUANTUM ESPRESSO. If not, please read the general User's Guide for QUANTUM ESPRESSO, found in directory Doc/ two levels above the one containing this guide; or consult the web site: http://www.quantum-espresso.org.

Further documentation, beyond what is provided in this guide, can be found in the directory PP/Doc/, containing a copy of this guide. People who want to contribute to QUANTUM ESPRESSO should read the Developer Manual, found in directory Doc/ two levels above the one containing this guide: Doc/developer_man.pdf.

2 People and terms of use

The PostProc package was originally developed by Stefano Baroni, Stefano de Gironcoli, Andrea Dal Corso (SISSA), Paolo Giannozzi (Univ. Udine), and many others. We mention in particular:

- Andrea Benassi (SISSA) for the `epsilon` utility, Tae Yun Kim and Cheol-Hwan Park (Seoul National University) for fixes to it;
- Dmitry Korotin (Inst. Met. Phys. Ekaterinburg) for the `wannier_ham` utility;
- Georgy Samsonidze (Bosch Research) for the interface with the Berkeley GW code;
- The late Prof. Eyvaz Isaev for the Fermi Surface code;
- Natalie Holzwarth (WFU) for the PAW projection in code `projwfc.f90`;
- Takashi Koretsune and Florian Thoele (ETHZ) for noncollinear magnetisation support with USPP and PAW pseudopotentials in code `pw2wannier.f90`.
- Leopold Talirz (U.York) for extensions and fixes to `pp.x`.

PostProc is free software, released under the GNU General Public License. See: `http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt`, or the file License in the distribution).

We shall greatly appreciate if scientific work done using this code will contain an explicit acknowledgment and the following reference:

> P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. Fabris, G. Fratesi, S. de Gironcoli, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovitch, J.Phys.:Condens.Matter 21, 395502 (2009), http://arxiv.org/abs/0906.2569

3 Compilation

PostProc is distributed together with QUANTUM ESPRESSO. For instruction on how to download and compile QUANTUM ESPRESSO, please refer to the general Users' Guide, available in file `Doc/user_guide.pdf` under the main QUANTUM ESPRESSO directory, or in web site `http://www.quantum-espresso.org`.

Once QUANTUM ESPRESSO is correctly configured, PostProc can be compiled by just typing `make pp`, from the main QUANTUM ESPRESSO directory; or typing `make` from the `PP/` subdirectory. Several executable codes are produced in `PP/bin` and linked to `bin/`.

4 Usage

All codes for which input documentation is not explicitly mentioned below have some documentation in the header of the fortran sources. In the following, "Example N" stands for subdirectory `examples/exampleN/`.

All quantities whose dimensions are not explicitly specified are in RYDBERG ATOMIC UNITS. Charge is "number" charge (i.e. not multiplied by e); potentials are in energy units (i.e. they are multiplied by e).

4.1 Plotting selected quantities

The main postprocessing code `pp.x` extracts the specified data from the data files produced by `PWscf` (`pw.x` executable) or `CP` (`cp.x` executable); prepares data for plotting by writing them into formats that can be read by several plotting programs.

Quantities that can be read or calculated are:

charge density
spin polarization
various potentials
local density of states at E_F
local density of electronic entropy
STM images
selected squared wavefunction
ELF (electron localization function)
RDG (reduced density gradient)
integrated local density of states

Various types of plotting (along a line, on a plane, three-dimensional, polar) and output formats (including the popular cube format) can be specified. Moreover data can be saved to an intermediate (formatted) file so that more data set can be summed or subracted in a later run. The output files can be directly read by the free plotting system Gnuplot (1D or 2D plots), or by code `plotrho.x` that comes with `PostProc` and produces PostScript 2D plots, or by advanced plotting software XCrySDen and gOpenMol (3D plots).

See file `Doc/INPUT_PP.*` for a detailed description of the input for code `pp.x`. See Example 01 for an example of a charge density plot, Example 03 for an example of STM image simulation.

Planar averages Code `plan_avg.x` calculates planar averages of Kohn-Sham orbitals. Code `average.x` calculates planar averages of quantities calculated by `pp.x` (e.g. potentials, charge, magnetization densities). Note that `average.x` reads the intermediate file produced by `pp.x`, not data files produced by `pw.x`. Examples of usage of `average.x` can be found in `examples/WorcFct_ex` and in `examples/dipole_example/`.

All-electron charge `pawplot.x` produces plots of the all-electron charge for PAW calculations.

About Bader's analysis In `http://theory.cm.utexas.edu/henkelman/code/bader/` one can find a software that performs Bader's analysis starting from charge on a regular grid.

One should use PAW to compute the charge density. The required "cube" format can be produced using `pp.x` (info by G. Lapenna who has successfully used this technique, but adds: "Problems occur with polar X-H bonds or in all cases where the zero-flux of density comes too close to atoms described with pseudo-potentials"). This code should perform decomposition into Voronoi polyhedra as well, in place of obsolete code `voronoy.x` (removed from distribution since v.4.2). Alternatively, you can use *CRITIC2*, available at `https://github.com/aoterodelaroza/critic2`, which can read directly `pw.x` output and "XSF" files. *CRITIC2* functionaly include Bader's AIM, ELF, laplacian of density and potentials, non-covalente interaction (NCI) plots and much more.

4.2 Band structure, Fermi surface

The code `bands.x` reads data file(s), extracts eigenvalues, regroups them into bands (the algorithm used to order bands and to resolve crossings may not work in all circumstances, though). The output is written to a file in a simple format that can be directly read and converted to plottable format by auxiliary code `plotband.x`. Unpredictable plots may results if k-points are not in sequence along lines, or if two consecutive points are the same. The code `bands.x` performs as well a symmetry analysis of the band structure. For a complete input description, see`Doc/INPUT_bands.*`. See Example 01, Example 04 and Example 06 for simple band plots.

The calculation of Fermi surface can be performed using code `fs.x`. The resulting file in .bxsf format can be read and plotted using XCrySDen. See Example 02 for an example of Fermi surface visualization (Ni, including the spin-polarized case).

4.3 Projection over atomic states, DOS

The code `projwfc.x` calculates projections of wavefunctions over atomic orbitals. The atomic wavefunctions are those contained in the pseudopotential file(s). The Löwdin population analysis (similar to Mulliken analysis) is presently implemented. The projected DOS (or PDOS: the DOS projected onto atomic orbitals) can also be calculated and written to file(s). More details on the input data are found in file `Doc/INPUT_PROJWFC.*`. The ordering of the various angular momentum components (defined in routine `flib/ylmr2.f90`) is as follows: $P_{0,0}(t)$, $P_{1,0}(t)$, $P_{1,1}(t)cos\phi$, $P_{1,1}(t)sin\phi$, $P_{2,0}(t)$, $P_{2,1}(t)cos\phi$, $P_{2,1}(t)sin\phi$, $P_{2,2}(t)cos2\phi$, $P_{2,2}(t)sin2\phi$ and so on, where $P_{l,m}$=Legendre Polynomials, $t = cos\theta = z/r$, $\phi = atan(y/x)$.

Data produced by code `projwfc.x` can be further analysed using auxiliary codes `sumpdos.x` (sums selected PDOS by specifying the names of files containing the desired PDOS: type `sumpdos.x -h` or look into the source code for more details) and `plotproj.x` .

The total electronic DOS can also be calculated by code `dos.x`, whose complete input documentation is in `Doc/INPUT_DOS.*` See Example 02 for total and projected electronic DOS calculations; see Example 03 for projected and local DOS calculations.

The DOS projected over *molecular* states (e.g. for a molecule on a surface system) can be computed using code `moleculardos.x` (courtesy of Guido Fratesi). See file `Doc/INPUT_MOLDOS.*` for input documentation and directory `MolDos_example/` for an example.

4.4 Wannier functions

There are several Wannier-related utilities in `PostProc`:

1. The "Poor Man Wannier" code `pmw.x`, to be used in conjunction with DFT+U calculations (see Example 05)

2. The interface with Wannier90 code, `pw2wannier.x`: see the documentation in `W90/` (you have to install the Wannier90 plug-in)

3. The `wannier_ham.x` code generates a model Hamiltonian in Wannier functions basis: see `examples/WannierHam_example/`.

Note that the `wfdd.x` code has been moved to `CP`.

4.5 Interfaces to/from other code

Codes `pw2bgw.x` and `bgw2pw.x` convert data files from `pw.x` to a format suitable for usage by the Berkeley GW code, and vice versa. See files `Doc/INPUT_pw2bgw.*` and `Doc/INPUT_bgw2pw.*` for input data documentation.

Undocumented code `pw2gw.x` converts data files from `pw.x` to a format suitable for usage by another GW code.

Code `pw_export.x`, not compiled by default, is an interface to other codes, documented in `Doc/INPUT_pw_export.*` Code `qexml.x`, not compiled by default, is a template that is useful to follow when wrting interfaces.

4.6 Other tools

Code `wfck2r.x` converts Kohn-Sham orbitals from reciprocal to real space. It is a useful starting point if you need to access wavefunctions and perform postprocessing operations that are not implemented in Quantum ESPRESSO.

Code `epsilon.x` calculates RPA frequency-dependent complex dielectric function. Documentation is in file `Doc/eps_man.tex`.

Code `initial_state.x` calculates the initial state contribution to the Core-level shift. See `examples/CLS_IS_example/` for an example, and `examples/CLS_FS_example/` for the corrsponding final state calculation of Core-level shifts.

5 Troubleshooting

Almost all problems in Quantum ESPRESSO arise from incorrect input data and result in error stops. Error messages should be self-explanatory, but unfortunately this is not always true. If the code issues a warning messages and continues, pay attention to it but do not assume that something is necessarily wrong in your calculation: most warning messages signal harmless problems.

Some postprocessing codes complain that they do not find some files For Linux PC clusters in parallel execution: in at least some versions of MPICH, the current directory is set to the directory where the executable code resides, instead of being set to the directory where the code is executed. This MPICH weirdness may cause unexpected failures in some postprocessing codes that expect a data file in the current directory. Workaround: use symbolic links, or copy the executable to the current directory.

***error in davcio* in postprocessing codes** Most likely you are not reading the correct data files, or you are not following the correct procedure for postprocessing. In parallel execution: if you did not set `wf_collect=.true.`, the number of processors and pools for the phonon run should be the same as for the self-consistent run; all files must be visible to all processors.