
postqe Documentation

Release 0.1

M. Palumbo, D. Brunato, P. D. Delugas

July 19, 2017

CONTENTS

1	Introduction	3
1.1	Installation	3
1.2	General notes	4
1.2.1	Plotting	4
2	Tutorial	5
2.1	Fitting the total energy (examples 3)	5
3	postqe package	7
3.1	Submodules	7
3.2	postqe.constants module	7
3.3	postqe.eos module	7
3.4	postqe.plot module	7
3.5	postqe.readutils module	7
4	Indices and tables	9

Contents:

INTRODUCTION

`postqe` is a Python package to perform postprocessing calculations for results obtained with the Quantum Espresso (QE) code ¹. The package provides Python functions to post-process the results, such as plotting the charge density (or the bare/Hartree/total potentials) on 1D or 2D sections, fitting the total energy with Murnaghan Equation of State (EOS), etc. The package also exposes some QE functionalities in Python using the F2PY code ² and wrappers to generate Python modules from QE dynamically linked libraries.

It is meant to be imported in your own Python code or used from the command line (see the Tutorial part of this documentation). It is also meant for people who want to tinker with the code and adapt it to their own needs. The package is based on numpy, scipy and matplotlib libraries.

Current features of the package include:

- Plot 1D or 2D sections of the charge density
- Plot 1D or 2D sections of the bare, Hartree or total potential
- Fit the total energy $E_{tot}(V)$ with Murnaghan's equation of state

Installation

You can download all package files from GitHub and then install it with the command:

```
sudo python setup.py install
```

The most useful functions for the common user are directly accessible from the `postqe`. You can import all of them as:

```
from postqe import *
```

or you can import only the ones you need. The above command also makes available a number of useful constants that you can use for unit conversions.

More functions are available as submodules. See the related documentation for more details. Note, however, that most of these functions are less well documented and are meant for advanced users or if you want to tinker with the code.

¹ <http://www.quantum-espresso.org/>

² <https://docs.scipy.org/doc/numpy-dev/f2py/>

General notes

Plotting

`postqe` uses the *matplotlib* library for Plotting. Some functions in the package are simply useful wrappers of *matplotlib* functions for common uses. They return a *matplotlib* object which can be further adapted to specific needs and personal taste.

TUTORIAL

This is a simple tutorial demonstrating the main functionalities of `postqe`. The examples below show how to use the package to perform the most common tasks. The code examples can be found in the directory *examples* of the package and can be run either as interactive sessions in your Python interpreter or as scripts. The tutorial is based on the following examples:

Example n.	Description
1	Plot a 1D section of the charge density
2	Plot a 1D section of the charge density
3	Fit $E_{tot}(V)$ for a cubic (isotropic) system using Murnaghan EOS

Several simplified plotting functions are available in `postqe` and are used in the following tutorial to show what you can plot. Note however that all plotting functions need the `matplotlib` library, which must be available on your system and can be used to further tailor your plot.

Fitting the total energy (examples 3)

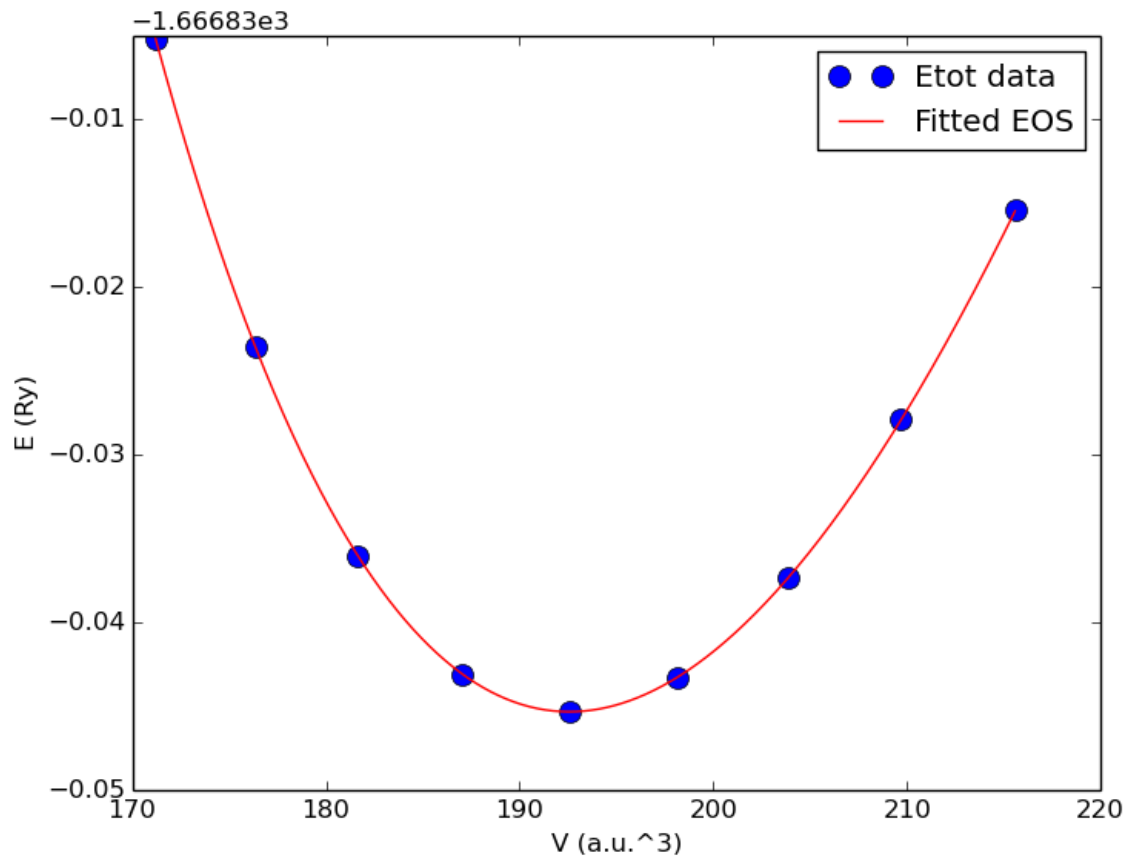
The simplest task you can do with `postqe` is to fit the total energy as a function of volume $E_{tot}(V)$ (example3). You can use an equation of state (EOS) such as Murnaghan's or similar. Currently the Murnaghan EOS and quadratic and quartic polynomials are implemented in `postqe`.

Let's see how to fit $E_{tot}(V)$. This is the case of isotropic cubic systems (simple cubic, body centered cubic, face centered cubic) or systems which can be approximated as isotropic (for example an hexagonal system with nearly constant c/a ratio).

The `fitEtotV()` needs in input a file with two columns: the first with the volumes (in $a.u.^3$), the second with energies (in $Ryd/cell$). It returns the volumes V and energies E from the input file plus the fitting coefficients a and the χ^2 *chi*. The fitting results are also written in details on the *stdout*:

```
# Murnaghan EOS                                chi squared= 6.3052568895e-09
# Etotmin= -1.6668753460e+03 Ry                  Vmin= 1.9256061524e+02 a.u.^3          B0= 3.9507615923e+03 kbar
#####
('# V *a.u.^3)', '\t\t', 'Etot', ' (Ry)\t\t', 'Etotfit', ' (Ry)\t\t', 'Etot-Etotfit (Ry)\t\tP (kbar)')
('1.7119697047e+02', '\t', '-1.6668351807e+03\t -1.6668351587e+03\t -2.2057946126e-05\t 6.2382144794e-05')
('1.7637989181e+02', '\t', '-1.6668536038e+03\t -1.6668536431e+03\t 3.9279193061e-05\t 4.3100002530e-05')
('1.8166637877e+02', '\t', '-1.6668660570e+03\t -1.6668660710e+03\t 1.4066826679e-05\t 2.6537032641e-05')
('1.8705745588e+02', '\t', '-1.6668731355e+03\t -1.6668731118e+03\t -2.3774691499e-05\t 1.2288570223e-05')
('1.9255414767e+02', '\t', '-1.6668753764e+03\t -1.6668753460e+03\t -3.0400133255e-05\t 1.3270797876e-05')
('1.9815747866e+02', '\t', '-1.6668732871e+03\t -1.6668732783e+03\t -8.8363487976e-06\t -1.057727393e-05')
('2.0386847338e+02', '\t', '-1.6668673220e+03\t -1.6668673472e+03\t 2.5137771445e-05\t -1.9727140701e-05')
('2.0968815635e+02', '\t', '-1.6668579007e+03\t -1.6668579345e+03\t 3.3763105193e-05\t -2.7643217490e-05')
('2.1561755211e+02', '\t', '-1.6668454001e+03\t -1.6668453730e+03\t -2.7177809670e-05\t -3.450114352e-05')
```

Optionally, you can plot the results with the `plot_EV()`. The original data are represented as points. If `a!=None`, a line with the fitting EOS will also be plotted. The output plot looks like the following:



POSTQE PACKAGE

Submodules

Additional functions are available as submodules. Please note the documentation of these functions is still ongoing and can be incomplete or wrong.

postqe.constants module

postqe.eos module

postqe.plot module

postqe.readutils module

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`