User's Guide for EMBEDDED QUANTUM ESPRESSO (eQE 0.1.0)

Contents

1 Introduction

This guide gives a general overview of the contents and of the installation of the SUBSYSTEM DFT (EMBEDDED) flavor of QUANTUM ESPRESSO, eQE version 0.1.0.

This distribution contains the core packages PWscf (Plane-Wave Self-Consistent Field) for the calculation of electronic-structure properties within Subsystem Density-Functional Theory (DFT), using a Plane-Wave (PW) basis set and pseudopotentials.

All trademarks mentioned in this guide belong to their respective owners.

1.1 People

The maintenance and further development of the eQE and eQEtddft distribution is promoted by the Pavanello Research group (PRG) at the Chemistry Department of Rutgers – Newark, NJ.

Contributors to eQE, include:

- All the QUANTUM ESPRESSO contributors for providing the open source code base.
- Alessandro Genova for the ground state code, the parallelization framework, and the non-additive potentials.
- Davide Ceresoli for setting the ground for both the eQE and eQEtddft projects.
- Alisa Krishtal for writing and maintaining the eQEtddft code.
- Michele Pavanello for the non-local kinetic energy functionals.
- Robert DiStasio.
- Oliviero Andreussi.

1.2 Contacts

The web site for eQE and eQEtddft is http://eqe.rutgers.edu/. Releases and work in progress code can be downloaded from this site.

If you need to contact the developers for *specific* questions about coding, proposals, offers of help, etc., please send a message to the email ales.genova@rutgers.edu.

1.3 Terms of use

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

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

A. Genova, D. Ceresoli, M. Pavanello, J. Chem Phys. 141, 174101 (2014)

2 Installation

2.1 Download

Presently, eQE is distributed in source form.

A snapshot of the code can be found at

Uncompress and unpack the base distribution using the command:

```
tar -zxvf eqe-X.Y.Z.tar.gz
```

where X.Y.Z stands for the version number. A directory eqe-X.Y.Z/ will be created.

Alternatively, for the adventurers who want access to the bleeding edge of the code, it can be found on our SVN repository.

```
svn checkout svn+ssh://username@qeforge.qe-forge.org/svnroot/fde/trunk fde_trunk
```

Before downloading it make sure you have an account on qe-forge.org.

2.2 Compile

Compilation is identical to the regular quantum espresso distribution, refer to their user manual for the fine tuning needed to obtain an efficient executable for your machine.

For the impatient:

```
cd eqe-X.Y.Z
./configure
make pw
```

NOTA BENE: this will more often than not produce a slow executable. Make sure that high performance compilers and libraries are properly selected by manually modifying make.sys after running configure.

eQE NEEDS a parallel environment to properly run and communicate information across the subsystems. Make sure all the needed parallel compilers are present in your machine.

3 Running

You will need a separate input file for each fragment composing your system. Each input file is essentially a regular PWscf input file with only the atoms belonging to that specific fragment. eQE introduces a series of new keywords, whose complete list can be found in the INPUT_FDEPW.txt included in this distribution. The input files need to share the same prefix and be numbered starting from 0. In this example we will simulate a water trimer, so we have the following input files:

```
trimer.scf_0.in
trimer.scf_1.in
trimer.scf_2.in
```

These input files can be found in the `eQE_input_template` directory inside the source code package.

Although most of the original QUANTUM ESPRESSO variables can be defined independently in each of the fragments input files, some of them need to be consistent across them. Most notably the following have to be the same:

- `prefix`
- The supersystem simulation cell.
- `ecutwfc` and `ecutrho`.
- `ntyp` and the `ATOMIC_SPECIES` section. You'll have to include here all the elements in the supersystem, even if they are not in the specific fragment.
- `conv_thr` and `mixing_beta`

The following keywords can be safely chosen to be different:

- `occupations` and `degauss`
- `nspin`
- The `K_POINT` section.
- `fde_cell_split(i)` to independently define the electronic simulation cell of each fragment in the i direction.

To run the calculation we need to run the following command (to be adapted depending on your needs and available resources):

```
mpirun -np 6 fdepw.x -ni 3 -in trimer.scf
```

where `-np` defines the total number of processors used, `-ni` defines the number of fragments in your system, and `-in` defines the prefix of the input and output files.

The calculation will run on 6 processors, assigning two processors to each of the fragment.

By default the code will try to split the number of processors evenly. But it may be the case where the size of the fragments is eterogenous, and an even split of the processors would be highly inefficient.

It is possible to assign an arbitrary number of processors to each fragment, by creating an additional input file named `fragments_procs.in` .

In this file you just need to write the number of processors you want assigned to each fragment, one per line. For example writing:

```
2
4
2
```

will assign 2 processors to the first and third fragment, and 4 processors to the second. Obviously the `-np` parameter will have to be set accordingly.

4 Exporting the embedding potential

In order for `eQE` to compute the embedding potential of a single fragment explicitly, the keywork

```
fde_print_embedpot = .true.
```

mush be included in the `system` namelist of the fragment's input file.

Because of parallelization complexity, currently the embedding potential can be printed only for a single fragment. If the user specifies the keyword for more than a fragment, the code will only print the embedding potential of the fragment with the lowest id.

The embedding potential will be represented in the small electronic cell of the fragment and stored in a `.pp` file at the end of the calculation. It is possible to export it in several other formats using the standard QUANTUM ESPRESSO `pp.x` program.

Additionally, the embedding potential can be interpolated on an arbitrarily defined grid (e.g. an atom centered grid generated from a molecular code).

To do so, the `pp.x` that comes with the `eQE` distribution needs to be used. A file containing the arbitrary grid named `grid.dat` needs to be in the same directory. The format of the file is the following:

```
  npoints
x_1   y_1   z_1   w_1
x_2   y_2   z_2   w_2
...
x_n   y_n   z_n   w_n
```

where `x,y,z` are the cartesian coordinates in bohr of the gridpoints and `w` is the weight of the point.

Create a modified `pp.x` input file based on this template and name it `interpolatepp.in`:

```
&inputpp
/
&plot
   nfile = 1
   filepp(1) = 'filename_embedpot_0_alpha.pp'
   iflag = 3                          ! 3D plot
   output_format = 6                  ! cube file
   interpolation = 'bspline_custom'    ! interpolate on the points specified in 'grid.dat'.
   fileout = 'filename_embedpot_0_alpha.customgrid'           ! interpolated potential output file
/
```

Finally, execute:

```
/path/to/eQE/bin/pp.x < interpolatepp.in
```