Quantum Package tutorial
Programmers mode

Diata Traore

January 2023

Quantum package documentation: Link
QP git: Link
Associated talk: Link

(This document is a first version. Don't hesitate to report any error or suggestion to improve it!)

Exercise 1

This exercise proposes to discover the programmer mode through the creation of a plugin to print the electronic density on a grid for a given molecule. (The suggested code is not aimed to be as efficient or clean as possible, it has been written for the sake of a simple tutorial to discover QP.)

1.1 Create your plugin folder

1. In the folder `qp2/` run the following command
   `./bin/qpsh`
   The later launch the Quantum Package shell mode. It allows you to benefit from the IRPF90 and Ocaml tools that simplify the compilation and code development process. For more information on IRPF90: Link.
2. Go in the `plugins/local` folder and run the following command
   `qp_plugins create -n print_density_on_grid -r local`

You should have the following output:

```
Created plugin:
/home_lct/${QP_PATH}/qp2/plugins/local/print_density_on_grid
Needed modules:
[]
This corresponds to using the following modules:
[]
Which is reduced to:
[]
Installation      [ OK ]
```

In Appendix 1, the documentations shows you how to consider script from existing modules. This modules are added at the end of the `qp_plugins create` command. If you forgot them at this step, just add them in the NEED file you find in the new module folder (See question 1.3.1).

3. To install the plugin, run `qp_plugins install print_density_on_grid`

In the plugin folder, a `build.ninja` file has been created : it is the Makefile. Do not modify it by hand. To run the compilation enter the command `ninja`.

You notice the compilation is way faster than during the QP installation. Performing your compilation from a plugin folder permits to avoid compiling the entire program. One of the numerous interests of this module structure!

1.2 Finally writing your first program

Moreover, you notice an executable file appeared: `print_density_on_grid`. It is associated with the default program in `print_density_on_grid.irp.f`. Open it.

You recognize the program block introduced in the talk and that you can find on the slides available in https://dtraore97.github.io/ressources/QP.

From here, the tutorial will need Fortran knowledge. If you are not familiar with it, you can copy paste the program in the Appendix 2 and continue with the next section where we will test the program.

We start by defining the grid. (Start by deleting the Hello world line.)

1. The coordinates of the starting point can depend on the first nucleus. In the website "Index for programmers" you can find the `nucl_coord` variable. The table depend on two variables : the first one is the nucleus number (in order of appearance in the xyz file), and the second one is the coordinate (1=x , 2=y, 3=z). Thus you can define:
   ```
   r(1) = nucl_coord(1, 1)
   r(2) = nucl_coord(1, 2)
   r(3) = nucl_coord(1, 3)
   ```
   where r(:) as been defined as a double precision table with
   ```
   double precision :: r(3)
   ```

I let you define the other parameters to define your grid. To make it simple, we will only work on the x axis. Therefore, you need to define :

- The number of points: `nx`
- The length of the grid: `xlength` (the program uses atomic units[1])
- The step: `dx`
- Redefine the origin (if needed): `r(1)+=-xlength*0.5d0`

1. Create the loop over the grid. It should looks like:
   ```
   do ipoint=0, nx
   ! A naive comment line
   density_at_x = ...
   ```

[1] ... but the xyz file are in angström...

```
print *, r(1), density_at_x
r(1)+=dx
enddo
```

1.3 Using existing routines

It turns out that it exists a (documented) routine that computes the electronic density at a given spatial point. You can find it in `qp2/src/dft_utils_r/dm_in_r_routines.irp.f`.

1. Open the `NEED` file and add in in the first line `dft_utils_in_r`, the name of the needed module.
2. Re-open the program file. To use the needed routine in your loop, replace the commented line by
   `call dm_dft_alpha_beta_at_r(r, dm_a, dm_b)`
   In the latter command,
   - `r` is an input : the three parameters coordinates table.
   - `dm_a` and `dm_b` are outputs : the density at r. If you read the `dm_in_r_routines.irp.f` script, you notice they are one dimensional tables. In fact, the table includes density value for the electronic states asked. As we only consider the ground state here, we use the values `dm_a(1)` and `dm_b(1)`.
3. Finally, `density_at_x = dm_a + dm_b`, the sum of alpha and beta electrons.

1.4 Compilation and test

1. The program is almost done ! Type `ninja` and correct your script according to the errors.
2. Once your program compiled (congratulations), create a calculation repository preferably outside the `qp2/` folder
3. Create a xyz file, `BH.xyz`, containing the following lines[2].
   ```
   2
   #BH (From Ref. [1])
   B    0.0    0.0    0.0
   H    1.2324   0.0    0.0
   ```
4. Copy the `master_template.txt` file from https://dtraore97.github.io/ressources/QP.
5. Add the following lines at the end of the file (to understand the command, See the "user mode" tutorial).
   ```
   basis=sto-3g
   xyz_file=BH.xyz
   qp create_ezfio -b ${basis} ${xyz_file}
   qp run scf | tee scf.out
   qp run print_density_on_grid | tee print_density_scf.out
   ```
6. Copy the output on your computer and use the data to plot the density.

[2] or the coordinates of any molecule of your interest

7. If you want to compute a correlated density, add
   ```
   qp run fci | fci.out
   qp run print_density_on_grid | tee print_density_fci.out
   ```
   (or cis, cisd, ...)

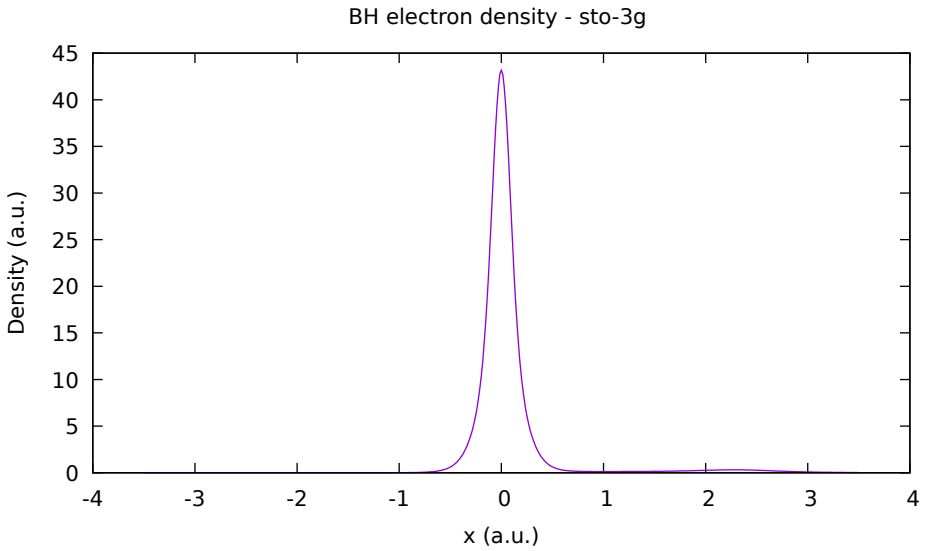

Figure 1: BH molecule electron density using sto-3g basis set at Hartree-Fock level. Both atoms are on the x-axis.

1.5 Conclusion

All you need know to program on Quantum Package is to get used with the available programs, routines and providers. You are free to code your own versions of existing script (highly recommended for a real understanding). Do not hesitate to create your github plugins repository to share your creations and to explore the one created by the QP contributors. Enjoy !

Appendix 1 : qp_plugins documentation

```
Usage:
        qp_plugins list [-iuq]
        qp_plugins download <url> [-n <name>]
        qp_plugins install <name>...
        qp_plugins uninstall <name>
        qp_plugins remove    <name>
        qp_plugins update [-r <repo>]
        qp_plugins create -n <name> [-r <repo>] [<needed_modules>...]

Options:
 list                        List
-i --installed              only the installed plugins
-u --uninstalled            only the uninstalled plugins
-q --repositories           the external repositories

 download <url>              Download an external repository.
                             The URL points to a tar.gz file or a git repository:
                             http://example.com/site/example.tar.gz
                             git@gitlab.com:user/example_repository

 install                     Install a plugin

 uninstall                   Uninstall a plugin

 remove                      Uninstall a plugin

 update                      Update the repository

 create
-n --name=<name>           Create a new plugin named <name>
-r --repository=<repo>     Name of the repository in which to create the plugin
```

Appendix 2 : program for the density on a grid

```
program print_density_on_grid
  implicit none
  BEGIN_DOC
! Print sum of alpha and beta electron density on the x axis
  END_DOC

  double precision :: r(3) ! Array for spatial coordinates 'r(x, y, z)' (a.u.)
  double precision :: dm_a(N_states), dm_b(N_states) ! densities at r
  double precision :: n_at_r
  double precision :: xlength, dx
  integer :: nx, ipoint

  r(1) = nucl_coord(1,1)
  r(2) = nucl_coord(1,2)
  r(3) = nucl_coord(1,3)

  nx = 500
  xlength = 7.d0
  dx =  xlength/dble(nx)
  r(1)+=-xlength*0.5d0

  do ipoint=0, nx
   call dm_dft_alpha_beta_at_r(r, dm_a, dm_b)
   n_at_r= dm_a(1) + dm_b(1)
   print *, r(1), n_at_r
   r(1)+=dx
  enddo

end
```