EPLF

Electron Pair Localization Function: User's Guide
May, 2010

Anthony Scemama

1 What is EPLF?

1.1 QMC Formulation

The Electron Pair Localization Function (EPLF) is a three-dimensional function measuring locally the electron pairing in a molecular system.[1] An electron i located at $\vec{r}_i$ is said to be paired to an electron j located at $\vec{r}_j$ if electron j is the closest electron to i. The amount of electron pairing at point $\vec{r}$ is therefore proportional to the inverse of

$$d(\vec{r}) = \sum_{i=1,N} \langle \Psi | \delta(\vec{r} - \vec{r}_i) \min_{j \neq i} r_{ij} | \Psi \rangle$$

where $d(\vec{r})$ is the shortest electron-electron distance at $\vec{r}$, $\Psi(\vec{r}_1, \ldots, \vec{r}_N)$ is the N-electron wave function and $r_{ij} = |\vec{r}_j - \vec{r}_i|$.

There are two different types of electron pairs: pairs of electrons with the same spin σ, and pairs of electrons with opposite spins σ and $\bar{\sigma}$. Hence, two quantities are introduced:

$$d_{\sigma\sigma}(\vec{r}) = \sum_{i=1,N} \langle \Psi | \delta(\vec{r} - \vec{r}_i) \min_{j \neq i; \sigma_i = \sigma_j} r_{ij} | \Psi \rangle$$

$$d_{\sigma\bar{\sigma}}(\vec{r}) = \sum_{i=1,N} \langle \Psi | \delta(\vec{r} - \vec{r}_i) \min_{j; \sigma_i \neq \sigma_j} r_{ij} | \Psi \rangle$$

The electron pair localization function is bounded in the $[-1, 1]$ interval, and is defined as

$$\mathrm{EPLF}(\vec{r}) = \frac{d_{\sigma\sigma}(\vec{r}) - d_{\sigma\bar{\sigma}}(\vec{r})}{d_{\sigma\sigma}(\vec{r}) + d_{\sigma\bar{\sigma}}(\vec{r})}$$

When the pairing of spin-unlike electrons is predominant, $d_{\sigma\sigma} > d_{\sigma\bar{\sigma}}$ and $\mathrm{EPLF}(\vec{r}) > 0$. When the pairing of spin-like electrons is predominant, $d_{\sigma\sigma} < d_{\sigma\bar{\sigma}}$ and $\mathrm{EPLF}(\vec{r}) < 0$. When the electron pairing of spin-like and spin-unlike electrons is equivalent, $\mathrm{EPLF}(\vec{r}) \sim 0$.

This localization function does not depend on the type of wave function, as opposed to the Electron Localization Function[2] (ELF) where the wave function has to be expressed on a single-electron basis set. The EPLF can therefore measure electron pairing using any kind of wave function: Hartree-Fock (HF), configuration interaction (CI), multi-configurational self consistent field (MCSCF) as well as Slater-Jastrow, Diffusion Monte Carlo (DMC), Hylleraas wave functions, *etc.* Due to the presence of the min function in the definition of $d_{\sigma\sigma}$ and $d_{\sigma\bar{\sigma}}$ these quantities cannot be evaluated analytically, so quantum Monte Carlo (QMC) methods have been used to compute three-dimensional EPLF grids via a statistical sampling of $|\Psi(\vec{r}_1, \ldots, \vec{r}_N)|^2$.

1.2 Analytical Formulation

Recently, we proposed an alternative definition of the EPLF which is analytically computable.[3] The min function is approximated using:

$$\min_{j \neq i} r_{ij} = \lim_{\gamma \to \infty} \sqrt{-\frac{1}{\gamma} \ln f(\gamma; r_{ij})}$$

[1] A. Scemama, P. Chaquin and M. Caffarel, *J. Chem. Phys.* **121**, 1725-1735 (2004).

[2] A. D. Becke and K. E. Edgecombe *J. Chem. Phys.* **92**, 5397 (1990).

[3] F. Alary, J.-L. Heully, A. Scemama, B. Garreau-de-Bonneval, K. I. Chane-Ching and M. Caffarel *Theor. Chem. Acc.* **126**, 243 (2010).

$$f(\gamma; r_{ij}) = \sum_{j \neq i} e^{-\gamma r_{ij}^2}$$

As $f(\gamma; r_{ij})$ has small fluctuations in the regions of interest (bonding regions, lone pairs,...), the integrals

$$\left\langle \Psi \left| \delta(\vec{r} - \vec{r}_i) \left(\lim_{\gamma \to \infty} \sqrt{-\frac{1}{\gamma} \ln f(\gamma; r_{ij})} \right) \right| \Psi \right\rangle$$

can be approximated as

$$\lim_{\gamma \to \infty} \sqrt{-\frac{1}{\gamma} \ln \langle \Psi | \delta(\vec{r} - \vec{r}_i) f(\gamma; r_{ij}) | \Psi \rangle}$$

The expectation values of the minimum distances are now given by:

$$d_{\sigma\sigma}(\vec{r}) \sim \lim_{\gamma \to \infty} \sqrt{-\frac{1}{\gamma} \ln f_{\sigma\sigma}(\gamma; \vec{r})}$$

$$d_{\sigma\bar{\sigma}}(\vec{r}) \sim \lim_{\gamma \to \infty} \sqrt{-\frac{1}{\gamma} \ln f_{\sigma\bar{\sigma}}(\gamma; \vec{r})}$$

with the two-electron integrals:

$$f_{\sigma\sigma}(\gamma; \vec{r}) = \left\langle \Psi \left| \sum_{i=1}^{N} \delta(\vec{r} - \vec{r}_i) \sum_{j \neq i; \sigma_i = \sigma_j}^{N} e^{-\gamma |\vec{r}_i - \vec{r}_j|^2} \right| \Psi \right\rangle$$

$$f_{\sigma\bar{\sigma}}(\gamma; \vec{r}) = \left\langle \Psi \left| \sum_{i=1}^{N} \delta(\vec{r} - \vec{r}_i) \sum_{j; \sigma_i \neq \sigma_j}^{N} e^{-\gamma |\vec{r}_i - \vec{r}_j|^2} \right| \Psi \right\rangle$$

If γ is not large enough in regions where electrons are close to each other, the value of $f_{\sigma\sigma}$ or $f_{\sigma\bar{\sigma}}$ can be greater than 1. In that case, the approximation of the min function is irrelevant since its value becomes negative. On the other hand, if γ is too large $f_{\sigma\sigma}$ and $f_{\sigma\bar{\sigma}}$ have almost always extremely small values. In this case, the values of $f_{\sigma\sigma}$ and $f_{\sigma\bar{\sigma}}$ are in practice not accurately representable with 64 bit floating point numbers, as they may be rounded to zero. We propose γ to depend on the density, such that the largest representable value of $d_{\sigma\sigma}$ is equal to the radius of a sphere that contains almost 0.1 electron:

$$\gamma(\vec{r}) = \left(\frac{4\pi}{3n} \rho(\vec{r}) \right)^{2/3} (-\ln(\epsilon))$$

where $n = 0.1$ and ϵ is the smallest floating point number representable with 64 bits ($\sim 2.10^{-308}$).

2 Structure of the EPLF code

The EPLF code is written using the IRPF90 environment[1], which needs Python>2.3. IRPF90 can be downloaded from http://irpf90.ups-tlse.fr

The input/output data are handled by the Eazy Fortran I/O library generator (EZFIO), which can be downloaded from http://ezfio.sourceforge.net. In this way, the data needed by the Fortran code is accessible both by Fortran an Python using the same API. The computational part of EPLF is written in IRPF90 and the user interfaces are written in Python.

[1] "IRPF90: a programming environment for high performance computing" A. Scemama, arXiv:0909.5012v1 [cs.SE] (2009)

3 Using the EPLF program

In theory, the EPLF can be computed from any kind of wave function (but note that the present code is written in the Restricted Hartree-Fock framework). The first step is to calculate a wave function using a quantum chemistry code. Then, the parameters of the wave function need to be saved in the EZFIO database in order to be communicated to the EPLF fortran program.

3.1 Wave function preparation

Wave functions calculated using Gaussian, Molpro and GAMESS can be read from the output files. A major constraint is to realize a *single point* calculation, all the following quantities appearing in the output file:

- The basis set
- The full set of MOs
- The coefficients of the Slater determinant expansion for CI wave functions.

3.1.1 Using Gaussian

In the Gaussian input file, use the keywords `GFPRINT` and `pop=Full`. In the case of CAS-SCF wave functions, use the `#p` keyword and the `SlaterDet` attribute of the `CAS` keyword. When doing a CAS-SCF with Gaussian, first do the Hartree-Fock calculation saving the checkpoint file and then do the CAS-SCF in a second step.

3.1.2 Using Molpro

Use the following options in the Molpro input file:

- `print,basis;`
- `gprint,civector;`
- `gprint,orbital;`
- `gthresh,printci=0.;` for MCSCF calculations

An RHF calculation is mandatory before any MCSCF calculation, since some information is printed only the RHF part. Be sure to print *all* molecular orbitals using the `orbprint` keyword, and to use the same spin multiplicity and charge between the RHF and the CAS-SCF .

3.1.3 Using GAMESS

For MCSCF calculations, first compute the MCSCF single-point wave function with the GUGA algorithm. Then, put the the MCSCF orbitals (of the `.dat` file) in the GAMESS input file, and run a single-point GUGA CI calculation with the following keywords:

- `PRTTOL=0.0001` in the `$GUGDIA` group to use a threshold of 10^{-4} on the CI coefficients
- `NPRT=2` in the `$CIDRT` group to print the CSF expansions in terms of Slater determinants
- `PRTMO=.T.` in the `$GUESS` group to print the molecular orbitals

3.1.4 Using your own code

Any other code producing a wave function can be used, as long as you are able to gather all the needed data.

3.2 Building the EZFIO database

An EZFIO database is a directory which contains all the needed input/output data needed by the code.

3.2.1 Using the provided wrapper

The output file from Molpro, GAMESS or Gaussian has to be transformed into an EZFIO database in order to be used by EPLF . This step is realized by calling the `to_ezfio.exe` command, followed by the name of the file containing the wave function:

```
to_ezfio.exe test.out
```

An EZFIO database is then produced, named '`test.out.ezfio`'.

3.2.2 Using your own code

Refer to the EZFIO web site[1] to learn how to use the EZFIO API to collect your data into an EZFIO database. The EZFIO definition file is the '`eplf.config`' file, located at the root of the EPLF package.

The EPLF code is able to compute the following functions on a regular three-dimensional grid:

- The electron pair localization function (EPLF)
- The electron localization function (ELF)
- The electron density

The gradients, the norm of the gradient and the laplacian of any of these three functions can also be requested. Note that the EPLF can only be calculated for a single Slater determinant.

3.3 Using the provided interface

The '`eplf_edit.py`' script allows to modify easily the conditions of the calculation: the choice of the functions to compute, the dimension of the grid, etc. Running the following command

```
eplf_edit.py test.out.ezfio
```

opens an input file in your default editor (defined by the `$EDITOR` environment variable). Lines starting with a `#` character are commented. The input file is separated in three sections.

3.3.1 The Computation section

If you are in a parallel environment, the first line of this section is

```
nproc = 0
```

which defines the number of processors to use. If nproc is chosen equal to 0, then the batch queuing system may choose itself the proper number of processors. Otherwise, specify the number of processors you want to use.

Then, all the computable functions appear, preceded by empty parentheses. Put an `X` between the parentheses for the functions you want to compute, for example

```
(X) elf
( ) density
(X) eplf
```

3.3.2 The Edit section

You can edit the nuclear coordinates and/or the parameters of the grid by un-commenting the `edit(geometry)` or `edit(grid_parameters)` statements.

When you save and quit the current input file, a new file will be open containing the parameters to edit. In the case of the grid parameters, as there is some redundance between the possible inputs quantities, the `Default` keyword can be used to replace the (x,y,z) specification.

[1] http://ezfio.sourceforge.net

```
##########################
# Grid : test.out.ezfio
##########################

# Number of points along x, y, z
40 40 40

# Coordinates of the origin (x,y,z)
-3.000000 -4.744648 -5.322027

# Coordinates of the opposite point (x,y,z)
Default

# Step sizes (x,y,z)
Default
```

3.3.3 The Clear section

All the previously calculated grids are saved in the EZFIO database. If the grid parameters are changed, these grids are inconsistent. Therefore, it may be necessary to clear the previously calculated grids. This can be realized by un-commenting the `clear(*)` statements. The `clear(all)` statement clears all the grids, and also the grid parameters.

3.3.4 Saving the input data

When the editor is closed, after saving the file, all the input data is saved into the EZFIO database. It can be modified later by running again the '`eplf_edit.py`' script. A shell script '`run_test.out.ezfio.sh`' is automatically created to launch the calculation.

3.4 Running the calculation

To run the calculation, run the `./run_test.out.ezfio.sh` command, or place it in a submission script of a batch queuing system.

Once the calculation is done, the grid is saved into the EZFIO database. It can be converted to a cube file using the `to_cube` tool, specifying as a first argument the EZFIO database, and which grid to convert as a second argument. For example:

```
to_cube test.out.ezfio eplf
```

produces the file '`test.out.ezfio_eplf.cube`'