DensToolKit 1.0.0

J. M. Solano-Altamirano & J. M. Hernández-Pérez

January 10, 2014

DensToolKit

Version 1.0.0

User manual

Disclaimer

This manual is not complete and for the time being it will continue to be so. This is a project that we have been developing essentially during our spare time, therefore it does not have priority over our current research work. However, we will try to update this manual as often as we can, specially when some major feature is implemented or an important bug is fixed.

J.M.S.A.
J.M.H.P.

Acknowledgements

We are grateful to the University of Guelph for providing us access to its library, DensToolKit had not been possible without it. JMSA is also thankful to the International Centre for Theoretical Physics. Many computational aspects of the DensToolKit's code, as well as its performance, are result of JMSA's attendance to the *Advanced School on High Performance and Grid Computing*, 2011.

Contents

Chapter 1

Introduction

1.1 About DensToolKit

DensToolKit (DTK) is a suite of programs to analize scalar and vector fields related to the electron density of a molecule. The purpose of breaking down the tasks into small programs is to have a small suite that can be used in scripts, much in the same sense as gnuplot.

While a decent plot for scientific publishing requires a reasonable amount of effort, spending time on doing plots to view intermmediate results can result in a waste of time.

After using gnuplot for a couple of years, I realized that it is far more efficient to work in this way. One simply needs to write a script, run it, and then occupy one's time and effort in doing something other than introducing over and over again the parameters for viewing volatile information.

On another hand, there is a considerable variety of programs written in fortran (especially in fortran 77) in the computational chemistry field. However, in order to maintain the compatibility with fortran compilers, the programmers are not taking the advantage of one of the greatest features of the object-oriented programming: the reusability of the software.

One of the features of DensToolKit is that it can be compiled in at least three of the major operative systems, namely Linux/Unix, Darwin(Mac OSX) and Windows (cygwin), using the gnu-gcc compiler.

1.2 Installation guide

Here we describe how to install DensToolKit on a range of UNIX-like operating systems.

DensToolKit have been tested on Linux, MacOSX, and Microsoft Windows 7, and we believe it should work under almost any POSIX system. The compilation on MS Windows 7 was performed under cygwin; regretably, full support for this operative system is not within our immediate plans.

1.2.1 Prerequisites

Every program of the suite DensToolKit is divided into two main parts. The most fundamental (and the core of DensToolKit) is the calculation of scalar and vector fields related to the electron density. Generally, the data obtained from the programs are saved into *.log, *.dat, *.tsv, or *.cub files.

Additionally, DensToolKit can perform calls to third party programs to generate some plots. Most of the programs use gnuplot, but some times povray is also called. As a general rule, the programs create a small script that can be passed directly to gnuplot/povray. The idea is to facilitate the detailed creation of a high-quality plot using those programs, but having a tool to easyly and fastly create plots of what has been calculated, skipping the time-wasting steps of using graphical interfaces.

That being said, before you compile DensToolKit, please ensure the following packages are installed in your system:

- **gnuplot 4.6 or later**. Under Linux, you can simply type

  ```
  #apt-get install gnuplot
  #yum install gnuplot
  ```

  (apt-get for Debian/Linux and its variants, yum for systems using RPMs), and under MacOSX

  ```
  #port install gnuplot
  ```

  where it has been assumed that macports is already installed in your system, and the character # means that you should run this commands as root/superuser.

  Unless stated otherwise, all of the third-party programs can be installed using this same method under Linux or MacOSX.

For MS Windows, you can get almost all the required programs by running the cygwin installer, and choosing the packages from the list. Gnuplot and povray also have independent installers on their websites.

More information about gnuplot can be found in: http://www.gnuplot.info

Below there is a list of the additional programs required to fully exploit the capabilities of DensToolKit.

- **povray 3.6 or later**: http://www.povray.org
- **ghostscript 9 or later**: http://www.ghostscript.com
- **epstool 3.08 or later**: http://pages.cs.wisc.edu/~ghost/gsview/epstool.htm
- **epstopdf 2.19 or later**: http://www.ctan.org/pkg/epstopdf
- **graphics magick 1.3.18 or later** http://www.graphicsmagick.org
- **gzip 1.6 or later** http://www.gzip.org

None of the listed programs are required to perform the calculations nor for generating *.log, *.dat, *.tsv, nor *.cub files. If you do not wish to install them (although we strongly recommend it), you can still run the programs and all the data files will be generated anyway (perhaps with several error messages). Later you can generate plots with your preferred software.

1.3 Installation from source archive

This section assumes you have basic knowledge of how to install simple programs in Linux through the command line (same applies for MacOSX and MS Windows —cygwin—).

- After you have received a copy of the source file, unpack the distributed .tar.gz:

```
$tar zxvf denstoolkit-YYYYMMDD-HHMM.tar.gz
$cd denstoolkit-YYYYMMDD-HHMM/src
```

- Just to verify that you have all the required packages installed, run

```
$./checkdependencies
```

If some package is missing, you will receive a message. You can skip this step, however, if this simple script works, then the remaining of the installation should be easy.

Especial attention must be paid to the povray program if you are using cygwin: After installing the povray binaries, look at the direct access created by the installer and (after right-clicking) check properties. In the properties window, you can see the full path of the program. You must add the path to your local `.bashrc` by adding at the end of this file the line

```
PATH=$PATH:/directory/where/povray/is
```

After this is done, type from the command line

```
$pvengine.exe
```

(if you use the new pvengine64.exe, then that is what you should type.) If povray opens, then you have configured right.

When all the required programs are installed, the script `checkdependencies` will display the message: `All required packages are installed, you can proceed to build and install denstoolkit!`

- Set the `INSTALL_PATH`. By default, the final binaries of DensToolKit will be installed in `/usr/local/bin`. You can change this by editing the Makefile file; in the first non empty line of Makefile, change `/usr/local/bin` by the directory of your choice.
- Set the correct command for povray (MS Windows only): Please, make yourself sure that you can call povray from the command line (if you have succesfully run the script `checkdependencies`, see above, then the next step is straightforward). Modify the Makefile by changing the line

  ```
  USERPOVCMD = povray
  ```

  by

  ```
  USERPOVCMD = pvengine.exe
  ```

  (or "`USERPOVCMD = pvengine64.exe`").
- Build the DensToolKit binaries:

  ```
  $make
  ```
- Install the binaries into the `INSTALL_PATH` directory

  ```
  $sudo make install
  ```
- Run some tests (optional).

  ```
  $make runtest
  ```

  You can review the sample output files in the directory

  `denstoolkit-YYYYMMDD-HHMM/outputs`

1.4 Uninstalling DensToolKit

In the source directory `denstoolkit-YYYYMMDD-HHMM/src` type:

```
$sudo make distclean
```

This will remove all the intermediate files used during compilation, all the binaries, and all the outputs produced (if you ran `$make runtest`). After this, you can safely remove the entire directory `denstoolkit-YYYYMMDD-HHMM`.

Chapter 2

Available density fields

As is well known from Quantum Mechanics theory, all the information of a quantum system is contained in the wave function. For a molecular system, the wave function, $\psi(\boldsymbol{r})$, can be spanned by M molecular orbitals $\chi_m(\boldsymbol{r})$ ($m = 1, \ldots, M$, and these molecular orbitals can be obtained by any of the self consisten field methods available)

$$\psi(\boldsymbol{r}) = \sum_{m=1}^{M} \sqrt{C_m}\chi_m(\boldsymbol{r}). \tag{2.1}$$

In turn, each molecular orbital can be spanned by a linear combination of Gaussian basis functions:

$$\chi_m(\boldsymbol{r}) = \sum_{\dot{A}=\dot{1}}^{\dot{N}} D_{m\dot{A}}\phi_{\dot{A}}(\boldsymbol{r} - \boldsymbol{R}_{\dot{A}}). \tag{2.2}$$

We have used here a slightly different notation in order to clearly separate the different index types that one encounters while handling molecular wave functions. We use upper case dotted indices for identifying Gaussian basis functions. Under our notation, each individual basis function is uniquely identified with a dotted index, and each basis function has associated three integers ($a^1_{\dot{A}}$, $a^2_{\dot{A}}$, and $a^3_{\dot{A}}$), and a point which denotes the center of the primitive $\boldsymbol{R}_{\dot{A}}$ (with this notation a primitive center $\boldsymbol{R}_{\dot{A}}$ can be the same than another primitive center $\boldsymbol{R}_{\dot{B}}$, *i.e.* the index associates the Cartesian coordinates of a nucleus to a particular basis function, rather than associating a basis function with a nucleus; similar behavior is observed for the integers $a^{\dot{i}}_{\dot{A}}$). This index naming is actually more natural for reading and understanding the `wfn` and `wfx` files.

Each Gaussian basis function is given by

$$\phi_{\dot{A}}(\boldsymbol{r}) = (x - R^1_{\dot{A}})^{a^1_{\dot{A}}}(x - R^2_{\dot{A}})^{a^2_{\dot{A}}}(x - R^3_{\dot{A}})^{a^3_{\dot{A}}} \exp\left(-\alpha_{\dot{A}}(\boldsymbol{r} - \boldsymbol{R}_{\dot{A}})^2\right), \tag{2.3}$$

and the normalization constants are already included in the coefficients $D_{m\dot{A}}$.

Finally, unless otherwise especified, we will assume that $\mathrm{Im}(\phi_{\dot{A}}) = 0$. While this is not true in general, it does hold for the gausian basis sets we will be interested in (*i.e.,* those obtained as `wfn` or `wfx` files from Gaussian, Gamess, MolPro, NWChem, etc.)

2.1 Electron density (ρ)

The electron density is therefore given by

$$\rho(\boldsymbol{r}) = \sum_{m=1}^{M} C_m \sum_{\dot{A}=\dot{1}}^{\dot{P}} \sum_{\dot{B}=\dot{1}}^{\dot{P}} D_{m\dot{A}} D_{m\dot{B}} \phi_{\dot{A}}(\boldsymbol{r}) \phi_{\dot{B}}(\boldsymbol{r}), \tag{2.4}$$

where $\dot{P}$ is the number of primitives used in the expansion of the wave function.

Defining the density matrix, $c_{\dot{A}\dot{B}}$, as

$$c_{\dot{A}\dot{B}} \equiv \sum_{m=1}^{M} C_m D_{m\dot{A}} D_{m\dot{B}}, \tag{2.5}$$

the electron density can be written as follows:

$$\rho(\boldsymbol{r}) = \sum_{\dot{A}=\dot{1}}^{\dot{P}} \sum_{\dot{B}=\dot{1}}^{\dot{P}} c_{\dot{A}\dot{B}} \phi_{\dot{A}}(\boldsymbol{r}) \phi_{\dot{B}}(\boldsymbol{r}). \tag{2.6}$$

Computationally, we can reduce the number of computations by using the following algorithm

$$\rho(\boldsymbol{r}) = \sum_{\dot{A}} \left\{ \phi_{\dot{A}} \left[c_{\dot{A}\dot{A}} \phi_{\dot{A}} + \frac{1}{2} \sum_{\dot{B}>\dot{A}} c_{\dot{A}\dot{B}} \phi_{\dot{B}} \right] \right\}. \tag{2.7}$$

This is the algorithm implemented in DensToolKit whenever the function allows it, and is twice as efficient as the direct computation posed by Eq. (2.6).

In what follows, unless stated otherwise, we will follow the Einstein summation convention, this is, a summation will be implied every time an index appears repeated in a single term, and the trivial vector arguments (such as the spatial dependence) will be dropped. Hence, the electron density is simply written as

$$\rho = c_{\dot{A}\dot{B}} \phi_{\dot{A}} \phi_{\dot{B}}. \tag{2.8}$$

While this notation is not customary in chemistry fields, it allows oneself to perform all the tensor gymnastics developed in other fields of physics, as we will see below.

Also, we will use latin indices to denote the Cartesian components of a vector. Therefore the i-th componen of a vector $\boldsymbol{A}$ is

$$[\boldsymbol{A}]^i \equiv A^i, \tag{2.9}$$

and the i-th partial derivative of a function φ is

$$\frac{\partial \varphi}{\partial x^i} \equiv \partial_i \varphi. \tag{2.10}$$

The Einstein summation convention is also implied every time a latin index appears twice in the same term. The limits of the summation should be obvious depending on the index type. For instance upper case, dotted indices will run from $\dot{1}$ to the number of primitives in the expansion ($\dot{P}$); while latin, lower case indices will run from 1 to 3 (since they are used for cartesian components). This is one of the advantages of introducing such index naming system.

2.2 Gradient of the electron density ($\nabla\rho$)

With the notation proposed in the previous section, the gradient of the electron density can be evaluated easily:

$$\partial_i \rho = c_{\dot{A}\dot{B}} \partial_i (\phi_{\dot{A}} \phi_{\dot{B}}) = 2 c_{\dot{A}\dot{B}} \phi_{\dot{A}} \partial_i \phi_{\dot{B}}, \tag{2.11}$$

where we have used the symmetry properties of $c_{\dot{A}\dot{B}}$ in the last equality, and

$$\begin{aligned} \partial_i \phi_{\dot{B}} &= \left(-2\alpha_{\dot{B}} (x^i - R^i_{\dot{B}})^{a^i_{\dot{B}}+1} + a^i_{\dot{B}} (x^i - R^i_{\dot{B}})^{a^i-1}\right) \prod_{j\neq i} (x^j - R^j_{\dot{B}})^{a^j_{\dot{B}}} \\ &\quad \times \exp\left(-\alpha_{\dot{B}} (\boldsymbol{r} - \boldsymbol{R}_{\dot{B}})^2\right) \end{aligned} \tag{2.12}$$

(no summation in Eq. (2.12)).

2.2.1 Higher derivatives the electron density

By defining the objects

$$\phi^i_{\dot{A}}(x^i) \equiv (x^i - R^i_{\dot{A}})^{a^i_{\dot{A}}} \exp\left(-\alpha_{\dot{A}} (x^i - R^i_{\dot{A}})^2\right), \qquad \text{(no summation)} \tag{2.13}$$

$$\begin{aligned} D^0_i &\equiv \phi^i_{\dot{A}}, \\ D^n_i &\equiv \partial^n_i \phi^i_{\dot{A}}, \qquad \text{(no summation)} \end{aligned} \tag{2.14}$$

and recalling that

$$\partial_i \phi_{\dot{A}}^{j} = 0, \qquad \text{for } i \neq j, \tag{2.15}$$

we can obtain any derivative of the primitives in any combination:

$$\partial_1^l \partial_2^m \partial_3^n \phi_{\dot{A}} = D_1^l D_2^m D_3^n. \tag{2.16}$$

The object D_i^n has the properties:

$$D_i^a D_i^a = D_i^{a+b}, \qquad \text{(no sum)} \tag{2.17}$$

Therefore, higher derivatives of the electron density, such as the Hessian, can be obtained from expression as

$$\partial_i \partial_k \phi_{\dot{A}} = D_i^1 D_j^1 \prod_{k \neq i,j} D_k^0. \tag{2.18}$$

While this notation may seem cumbersome, it simplifies the implementation of higher derivatives of the electron density by the use of arrays. For example, the Hessian of the electron density is given by

$$\partial_i \partial_j \rho = 2 c_{\dot{A}\dot{B}} \left(\phi_{\dot{A}} \partial_i \partial_j \phi_{\dot{B}} + \partial_i \phi_{\dot{A}} \partial_j \phi_{\dot{B}} \right). \tag{2.19}$$

In DensToolKit we evaluate up to fourth derivatives (when seeking critical points of LOL, see §3).

2.3 Magnitude of $\nabla\rho$

This function follows directly from the gradient of the electron density, and is given by

$$|\nabla\rho| = \sqrt{\nabla\rho \cdot \nabla\rho} = \sqrt{\partial_k \rho \, \partial_k \rho}, \tag{2.20}$$

and $\partial_i \rho$ is evaluated as explained in §2.2.

2.4 Laplacian of ρ ($\nabla^2\rho$)

The Laplacian of the electron density is evaluated by means of

$$\nabla^2 \rho = \partial_k \partial_k \rho = 2 c_{\dot{A}\dot{B}} \left(\partial_k \phi_{\dot{A}} \partial_k \phi_{\dot{B}} + \phi_{\dot{A}} \partial_k \partial_k \phi_{\dot{B}} \right) \tag{2.21}$$

2.5 Kinetic Energy G

The definition of the Kinetic energy density G we implemented in DensToolKit is (in terms of the density matrix $c_{\dot{A}\dot{B}}$)

$$G = \frac{1}{2} c_{\dot{A}\dot{B}} \nabla \phi^*_{\dot{A}} \cdot \nabla \phi_{\dot{B}} = \frac{1}{2} \partial_k \phi_{\dot{A}} (c_{\dot{A}\dot{B}} \partial_k \phi_{\dot{B}}) . \tag{2.22}$$

Let us recall that some improvements on the speed of DensToolKit rely on the fact that we are assuming that the primitives are pure-real. The parenthesis in the last equation denotes that in DensToolKit we first save the sum of the three terms $c_{\dot{A}\dot{B}} \partial_i \phi_{\dot{B}}$ within the inner loop, and after this loop is completed, we multiply and add each one of these sums by $\partial_i \phi_{\dot{A}}$ on the outer loop. While this procedure may seem a bit obscure, it improves the speed of the computations.

2.6 Kinetic Energy K

The Kinetic energy density K definition we implemented in DensToolKit is

$$K(\boldsymbol{r}) = \frac{1}{4} c_{\dot{A}\dot{B}} \left(\phi^*_{\dot{A}} \nabla^2 \phi_{\dot{B}} + \phi^*_{\dot{B}} \nabla^2 \phi_{\dot{A}} \right) = \frac{1}{2} \phi_{\dot{A}} (c_{\dot{A}\dot{B}} \partial_k \partial_k \phi_{\dot{B}}) . \tag{2.23}$$

2.7 Electron Localization Function (ELF)

We follow the standard definition of ELF, which we denote as $\eta(\boldsymbol{r})$:

$$\eta(\boldsymbol{r}) = \frac{1}{1 + [D(\boldsymbol{r})/D_h(\boldsymbol{r})]^2} . \tag{2.24}$$

Here $D(\boldsymbol{r})$ gives the probability density of finding a same-spin electron [1], and has the value

$$D = G - \frac{1}{8} \frac{|\nabla \rho|^2}{\rho} , \tag{2.25}$$

with G (the Kinetic energy density G), $\nabla \rho$, and ρ evaluated as in previous sections, while D_h is given by

$$D_h = \frac{3}{10} (3\pi^2)^{2/3} \rho^{5/3} . \tag{2.26}$$

2.8 Localized Orbital Locator (LOL)

DensToolKit evaluates the localized orbital locator (denoted by the letter γ) as

$$\gamma = \frac{1}{1+\tau}, \tag{2.27}$$

where

$$\tau = \frac{G}{D_h}, \tag{2.28}$$

and G and D_h as defined in previous sections.

2.9 Momentum density

For the implementation of this field, we closely follow the procedure given in Ref. [2]. In terms of the density matrix, the momentum density is given by

$$\pi(\boldsymbol{p}) = \hat{\phi}^*_{\dot{A}}(\boldsymbol{p}) c_{\dot{A}\dot{B}} \hat{\phi}_{\dot{B}}(\boldsymbol{p}), \tag{2.29}$$

where $\hat{\phi}_{\dot{A}}(\boldsymbol{p})$ is the Fourier transform of the primitive $\phi_{\dot{A}}(\boldsymbol{r})$, defined as follows:

$$\hat{\phi}_{\dot{A}}(\boldsymbol{p}) = \frac{1}{(2\pi)^{2/3}} \int \exp(-i\boldsymbol{p}\cdot\boldsymbol{r})\, \phi_{\dot{A}}(\boldsymbol{r}) d\boldsymbol{r}. \tag{2.30}$$

A table of the Fourier transforms of the cartesian primitives is listed in Ref. [2]. We reproduce here the first five terms, which are the ones we used in this version of DensToolKit (since the normalization constants of the primitives are already taken into account in the density matrix, the expressions in Table 2.1 do not include the normalization constants).

For the evaluation of this field, we must account for the imaginary terms of the Fourier terms. DensToolKit exploit the library `<complex>` distributed as standard library of c++.

2.10 Shannon entropy density

This field cannot be further optimized beyond the optimization achieved in the electron density. In fact, the Shannon entropy density field is given by

$$S_\rho(\boldsymbol{r}) = -\rho(\boldsymbol{r}) \ln\big(\rho(\boldsymbol{r})\big), \tag{2.31}$$

n	$\phi^k(x_k)$	$\hat{\phi}^k(p_k)$
0	$\exp(-\alpha x_k^2)$	$\frac{1}{(2\alpha)^{1/2}}\exp(-ip_kR_k)\exp(-p_k^2/(4\alpha))$
1	$x_k\exp(-\alpha x_k^2)$	$-i\frac{p_k}{(2\alpha)^{3/2}}\exp(-ip_kR_k)\exp(-p_k^2/(4\alpha))$
2	$x_k^2\exp(-\alpha x_k^2)$	$\frac{2\alpha-p_k^2}{(2\alpha)^{5/2}}\exp(-ip_kR_k)\exp(-p_k^2/(4\alpha))$
3	$x_k^3\exp(-\alpha x_k^2)$	$i\frac{p_k(p_k^2-6\alpha)}{(2\alpha)^{7/2}}\exp(-ip_kR_k)\exp(-p_k^2/(4\alpha))$
4	$x_k^4\exp(-\alpha x_k^2)$	$\frac{p_k^4-12\alpha p_k^2+12\alpha^2}{(2\alpha)^{9/2}}\exp(-ip_kR_k)\exp(-p_k^2/(4\alpha))$

Table 2.1: Fourier transforms of Cartesian GTOs with orbital exponent α, center R_k, and angular power n (single coordinate). No summation is implied here.

which uses the implemented algorithm for evaluating $\rho(\boldsymbol{r})$.

Let us not forget this is a density field, not the integrated Shannon entropy. Whether or not this field is useful as a density will not be discussed here.

The Shannon entropy in momentum space is given by (also a density field):

$$S_\pi(\boldsymbol{p}) = -\pi(\boldsymbol{p})\ln\big(\pi(\boldsymbol{p})\big), \tag{2.32}$$

and it also uses a simple call to evaluate $\pi(\boldsymbol{p})$, as given by 2.29.

2.11 Density matrix of order 1

In terms of the density matrix, the Density Matrix of order 1 (DM1, and denoted by the symbol $\Gamma(\boldsymbol{r},\boldsymbol{r}')$) is given by

$$\Gamma(\boldsymbol{r},\boldsymbol{r}') = \phi_{\dot{A}}(\boldsymbol{r})\big\{c_{\dot{A}\dot{B}}\phi_{\dot{B}}(\boldsymbol{r}')\big\}. \tag{2.33}$$

2.12 Electrostatic potential

The electrostatic potential is given by

$$V(\boldsymbol{r}) = \sum_{a=1}^{N_{nuc}}\frac{Z_a}{|\boldsymbol{r}-\boldsymbol{R}_a|} - \int\frac{\rho(\boldsymbol{x})}{|\boldsymbol{r}-\boldsymbol{x}|}d\boldsymbol{x}, \tag{2.34}$$

where $\boldsymbol{R}_A$ is the vector indicating the spatial position of nuclei A, whose charge is Z_A, and $\rho(\boldsymbol{x})$ is the electron density at the dummy point $\boldsymbol{x}$. In terms of the density matrix, the

electrostatic potential is given by

$$V(\boldsymbol{r}) = \sum_{a=1}^{N_{nuc}} \frac{Z_a}{|\boldsymbol{r}-\boldsymbol{R}_a|} - c_{\dot{A}\dot{B}} \int \frac{\phi_{\dot{A}}(\boldsymbol{x})\phi_{\dot{B}}(\boldsymbol{x})}{|\boldsymbol{r}-\boldsymbol{x}|} d\boldsymbol{x}. \tag{2.35}$$

Hence, the evaluation of the electrostatic potential is reduced to evaluate the integral

$$\mathfrak{I}_{\dot{A}\dot{B}} = \mathfrak{I}_{\dot{A}\dot{B}}(\boldsymbol{r}) = \int \frac{\phi_{\dot{A}}(\boldsymbol{x})\phi_{\dot{B}}(\boldsymbol{x})}{|\boldsymbol{r}-\boldsymbol{x}|} d\boldsymbol{x}. \tag{2.36}$$

For the remainder of this section, we will not use the Einstein summation convention. The last integral is written as

$$\mathfrak{I}_{\dot{A}\dot{B}} = \int \frac{d\boldsymbol{x}}{|\boldsymbol{r}-\boldsymbol{x}|} \exp\left(-\alpha_{\dot{A}}|\boldsymbol{x}-\boldsymbol{R}_{\dot{A}}|^2 - \alpha_{\dot{B}}|\boldsymbol{x}-\boldsymbol{R}_{\dot{B}}|^2\right) \prod_i (x^i - R^i_{\dot{A}})^{a^i_{\dot{A}}} (x^i - R^i_{\dot{B}})^{a^i_{\dot{B}}}. \tag{2.37}$$

Let us define the following constants

$$\alpha_{\dot{A}\dot{B}} \equiv \alpha_{\dot{A}} + \alpha_{\dot{B}}, \tag{2.38}$$

$$\boldsymbol{R}_{\dot{A}\dot{B}} \equiv \frac{\alpha_{\dot{A}}\boldsymbol{R}_{\dot{A}} + \alpha_{\dot{B}}\boldsymbol{R}_{\dot{B}}}{\alpha_{\dot{A}\dot{B}}}, \text{ and} \tag{2.39}$$

$$E_{\dot{A}\dot{B}} \equiv \exp\left(-\frac{\alpha_{\dot{A}}\alpha_{\dot{B}}}{\alpha_{\dot{A}\dot{B}}}(\boldsymbol{R}_{\dot{A}} - \boldsymbol{R}_{\dot{B}})^2\right), \tag{2.40}$$

which have the following symmetry properties

$$\alpha_{\dot{A}\dot{B}} = \alpha_{\dot{B}\dot{A}}, \qquad \boldsymbol{R}_{\dot{A}\dot{B}} = \boldsymbol{R}_{\dot{B}\dot{A}}, \qquad \text{and} \qquad E_{\dot{A}\dot{B}} = E_{\dot{B}\dot{A}}. \tag{2.41}$$

With this notation, the use of the same letter for the exponents and vector positions does not introduce confusion since the new quantities carry two indices, as opposed to the single index carried by the original single centered Gaussian. Thus, the integral now can be expressed as

$$\mathfrak{I}_{\dot{A}\dot{B}} = E_{\dot{A}\dot{B}} \int \frac{d\boldsymbol{x}}{|\boldsymbol{x}-\boldsymbol{r}|} \exp\left(-\alpha_{\dot{A}\dot{B}}|\boldsymbol{x}-\boldsymbol{R}_{\dot{A}\dot{B}}|^2\right) \prod_i (x^i - R^i_{\dot{A}})^{a^i_{\dot{A}}} (x^i - R^i_{\dot{B}})^{a^i_{\dot{B}}}, \tag{2.42}$$

which is also a symmteric function in the indices $\dot{A}$ and $\dot{B}$:

$$\mathfrak{I}_{\dot{A}\dot{B}} = \mathfrak{I}_{\dot{B}\dot{A}}. \tag{2.43}$$

Applying the Laplace transform given by Eq. (2.44),

$$\frac{1}{|\boldsymbol{x}-\boldsymbol{r}|} = \frac{2}{\sqrt{\pi}} \int_0^\infty \exp\left(-u^2(\boldsymbol{x}-\boldsymbol{r})^2\right), \tag{2.44}$$

into Eq. (2.42) yields

$$\mathfrak{I}_{\dot{A}\dot{B}} = \frac{2E_{\dot{A}\dot{B}}}{\sqrt{\pi}} \int_0^\infty du \int d\boldsymbol{x} \exp\left(-\alpha_{\dot{A}\dot{B}}(\boldsymbol{x} - \boldsymbol{R}_{\dot{A}\dot{B}})^2 - u^2(\boldsymbol{x} - \boldsymbol{r})^2 \prod_i (x^i - R^i_{\dot{A}})^{a^i_{\dot{A}}} (x^i - R^i_{\dot{B}})^{a^i_{\dot{B}}} \right). \tag{2.45}$$

It is convenient to use the variable

$$t^2 = \frac{u^2}{\alpha_{\dot{A}\dot{B}} + u^2}, \tag{2.46}$$

hence

$$\begin{aligned} \mathfrak{I}_{\dot{A}\dot{B}} &= 2E_{\dot{A}\dot{B}} \sqrt{\frac{\alpha_{\dot{A}\dot{B}}}{\pi}} \int_0^1 \frac{1}{(1-t^2)^{3/2}} \Bigg\{ \\ &\quad \times \int \Bigg[\exp\left(-\alpha_{\dot{A}\dot{B}}(\boldsymbol{x} - \boldsymbol{R}_{\dot{A}\dot{B}})^2 - \frac{\alpha_{\dot{A}\dot{B}} t^2}{1-t^2} (\boldsymbol{x} - \boldsymbol{r})^2 \right) \\ &\quad \times \prod_i (x^i - R^i_{\dot{A}})^{a^i_{\dot{A}}} (x^i - R^i_{\dot{B}})^{a^i_{\dot{B}}} d\boldsymbol{x} \Bigg] \Bigg\} dt \end{aligned} \tag{2.47}$$

$$\begin{aligned} &= \frac{2E_{\dot{A}\dot{B}}\pi}{\alpha_{\dot{A}\dot{B}}} \int_0^1 dt \Bigg\{ \\ &\quad \prod_i \Bigg[\sqrt{\frac{\alpha_{\dot{A}\dot{B}}}{\pi(1-t^2)}} \int dx^i (x^i - R^i_{\dot{A}})^{a^i_{\dot{A}}} (x^i - R^i_{\dot{B}})^{a^i_{\dot{B}}} \\ &\quad \times \exp\left(-\alpha_{\dot{A}\dot{B}}(x^i - R^i_{\dot{A}\dot{B}})^2 - \frac{\alpha_{\dot{A}\dot{B}} t^2}{1-t^2} (x^i - r^i)^2 \right) \Bigg] \Bigg\}. \end{aligned} \tag{2.48}$$

The terms $(x^i - R^i_{\dot{A}})^{a^i_{\dot{A}}} (x^i - R^i_{\dot{B}})^{a^i_{\dot{B}}}$ can be expanded in a series of Hermite polynomials of $(x^i - R^i_{\dot{A}})$ and $(R^i_{\dot{A}} - R^i_{\dot{B}})$, and the resultant expansion can then be analytically integrated. In DensToolKit we coded an implementation that closely follows the method proposed by McMurchie *et. al.* [3]. However, we use our own implementation of the Boys function (denoted as $F_n(x)$), which offers relative accuracy of $F_n(x)$ to at least $O(10^{-12})$ for the range $0 \leq x \leq 1000$, and $0 \leq n \leq 6$. DensToolKit is coded in an *as-needed* basis, thus we are not concerned with higher order terms of n since 6 is the maximum value needed for the computations supported by DensToolKit.

For this field, we also exploit the symmetry of the product $c_{\dot{A}\dot{B}}\phi_{\dot{A}}\phi_{\dot{B}}$, hence

$$\begin{aligned} V(\boldsymbol{r}) &= \sum_a \frac{Z_a}{|\boldsymbol{r} - \boldsymbol{R}_a|} - \sum_{\dot{A}} \left(c_{\dot{A}\dot{A}} \int \frac{\phi^2_{\dot{A}}(\boldsymbol{x})}{|\boldsymbol{r} - \boldsymbol{x}|} d\boldsymbol{x} + 2 \sum_{\dot{B} > \dot{A}} c_{\dot{A}\dot{B}} \int \frac{\phi_{\dot{A}}(\boldsymbol{x})\phi_{\dot{B}}(\boldsymbol{x})}{|\boldsymbol{r} - \boldsymbol{x}|} d\boldsymbol{x} \right) \\ &= \sum_a \frac{Z_a}{|\boldsymbol{r} - \boldsymbol{R}_a|} - \sum_{\dot{A}} \left(c_{\dot{A}\dot{A}} \mathfrak{I}_{\dot{A}\dot{A}} + 2 \sum_{\dot{B} > \dot{A}} c_{\dot{A}\dot{B}} \mathfrak{I}_{\dot{A}\dot{B}} \right). \end{aligned} \tag{2.49}$$

Chapter 3

Topological analysis

In addition to evaluate density fields on different grids, one of the purposes of DensToolKit is to seek the critical points of the electron density of a molecule, as proposed in Quantum Theory of Atoms in Molecules (QTAIM). By critical point, we mean any point $\boldsymbol{r}_*$ that accomplishes

$$\nabla F(\boldsymbol{r}_*) = \mathbf{0}, \tag{3.1}$$

where F is any density field. The critical points can be classified according to the sign of the Hessian eigen-values at $\boldsymbol{r}_*$, as we explain below.

In version 1.0.0, we have implemented the seek of all electron density critical points, including bond paths, and some of the critical points of LOL.

The development of this part of DensToolKit will be active during the upcoming years, and it is expected to grow as the needs of our research group.

3.1 Classification of critical points

A critical point can be classified by means of the Hessian eigen-values at the critical point. Since we will be interested only in three dimensional spaces, and on non-degenerated Hessians, the Hessian will always have three eigenvalues. Therefore, we can find four combinations of the eigenvalues' signs (we will follow the QTAIM notation, wherein a critical point is denoted as a couple $(3,s)$; here "3" stands for the three eigenvalues of the Hessian, and $s = n_+ - n_-$, where n_+ (n_-) is the number of positive(negative) eigenvalues; s is known as the signature of the critical point):

- (3,-3): The three Hessian eigenvalues are all negative. A critical point of this nature represents a maximum. A critical point (3,-3) of the electron density is called

Type	Density Field	
	ρ	LOL (γ)
ACP	✓	✓
BCP	✓	✓
RCP	✓	×
CCP	✓	×

Table 3.1: Implemented functions for seeking critical points of different density fields in DensToolKit 1.0.0.

Attractor Critical Point (ACP).

- (3,-1): Two negative and one positive Hessian eigenvalues. This is a saddle point, maximum in two directions and minimum in the other. When the critical point is of the electron density, it is called *Bond Critical Point* (BCP).
- (3,+1): One negative and two positive Hessian eigenvalues. This is also a saddle point, but this time is a maximum in one direction and mininum in the rest. The electron density critical point is called *Ring Critical Point* (RCP).
- (3,+3): All the three Hessian eigenvalues are positive. This critical point corresponds to a minimum of the field. Critical points of the electron density with this properties are called *Cage Critical Points* (CCP).

As a convention, we will use the names ACP, BCP, RCP and CCP for the critical points regardless the density field they are.

Below we present the implemented critical points for different density fields in form of a Table.

For the search of critical points, we follow the general strategy proposed by Banerjee *et. al.* [4] and Popelier [5].

3.1.1 Electron density critical points

Below we enlist some details regarding the searching algorithms for the critical points (CP) of the electron density.

ACP We do not perform actual searches of ACPs. Instead, we assume that these CPs are located at every nucleus of the molecule.

BCP We start the search of BCPs for every couple of atoms that are "bonded" according to the criterion used by the bond-drawing algorithms (two atoms are considered bonded, in terms of drawing a line between them) if the distance between those

two atoms is less than the sum of the atomic Van der Walls radius. The seed (initial point to start the search of a CP) is set to be the middle point between the two bonded nuclei. After all these points have been used as seeds, we look for the largest distance of separation between the "bonded" atoms, and then we perform the search for every couple of atoms that are separated by up to two times the maximum bond distance with the seed located at the middle point of every one of the atoms under this criterion. In this manner, we allow for the Hydrogen bonds to be accounted for.

RCP For each pair of BCPs whose distance of separation is less than the maximum bond distance (as was found for the BCPs search) we set the seed for the search of a RCP to be the middle point of those BCPs.

CCP For each pair of RCPs in the molecule, if the distance of separation among them is less than two times the maximum bond distance (as found in the search of BCPs), then we set the seed to be the middle point between the two RCPs.

3.1.2 LOL critical points

The search for LOL critical points is not finished in version 1.0.0. However, we will keep developing the implementation of these critical points in the near future, and perhaps also for other fields.

ACP We look for LOL ACPs around every molecule's nucleus. In DensToolKit 1.0.0, we set twelve seeds around every nucleus. The seeds are set by drawing an imaginary icosahedron centered at the nucleus position, and each seed is one of the icosahedron vertices.

BCP We set the seed for looking LOL BCPs as the middle point between every pair of ACPs found in the molecule.

3.1.3 Gradient Paths

A gradient path is defined as a curve that follows the direction of the gradient of the field, and the points that form the gradient paths can be defined through

$$\boldsymbol{r}(s) = \boldsymbol{r}_0(s_1) + \int_{s_1}^{s_2} \nabla F(\boldsymbol{r}(t))dt. \tag{3.2}$$

Here F is any wee-defined field, and $\boldsymbol{r}(s)$ is the set of points that belong to the gradient path that passes through $\boldsymbol{r}_0$, and s and t are parametric variables (see Ref. [6]).

In QTAIM, a gradient path that passes through a BCP is called a bond gradient path or simply a bond path. In DensToolKit 1.0.0, we have implemented the search of bond gradient paths for the electron density. Other fields will be implemented in the future.

The bond paths are integrated using a fifth order Runge-Kutta method with a step size of 0.1 a.u.

3.2 Properties along bond paths

DensToolKit can evaluate any of the implemented scalar fields listed in section §2 on the bond paths. The basic algorithm consists of, provided two atoms in a molecule, first finding the BCP, and then integrating the gradient path taking as $\boldsymbol{r}_0$ the BCP. The set of points computed are stored in an array, and then the requested field is evaluated at each of the points belonging to the bond path.

Chapter 4

Programs

In this chapter, a description for each program of DensToolKit is provided. The description given here is a bit more detailed than the description offered in each individual help menu of the programs.

Whenever possible, we will provide an example of how to run the program, and after §4.2 it will be assumed that a file called `ciclopropano_pbe6311.wfx` (which should be included along with the source code) exists and is the main input of the program under review.

4.1 General behaviour

4.1.1 Input files

All the programs belonging to DensToolKit read wave function files. These files can be obtained by Computational Quantum Chemistry packages such as Gaussian [7], GAMESS [8], or NWCHEM [9]. DensToolKit can read either the old `wfn` files or the newer `wfx` files [10].[1]

The wave function file is *always* the first argument in the command line. The only exception is when requesting the displaying of the help menu (which actually does not evaluate anything). Thus, the usage of every program in DensToolKit obeys the following pattern:

```
$dtk*** inputname.wf?  [option [value(s)]] ...[option [value(s)]]
```

[1] Some incompatibilities may arise when mixing files between Windows and Unix-like operative systems. Please ensure that the files `wfx` or `wfn` use the same newline as the operative system where DensToolKit is running. For this, you can use the programs `dos2unix` and `unix2dos`.

Here, the stars represent any of the programs in DensToolKit, the question mark can be `n` or `x`, and the options (with possible values) are specific to each program.

4.1.2 Output names

Most of the programs included in DensToolKit produce some kind of output. The main output is numerical, and it is saved in different files. While the specific output depends on the particular program, there are a few conventions that we adopt.

- **`dat files:`** This type contains single or multi-column data and is used whenever a field is evaluated at a line of points (one-dimensional grid). In some output files of this kind, comments are added at the beginning of the file. These comments start with the character `#`.
- **`tsv files:`** This file contains the information for two-dimensional grids. It is the format known as *tab separated values*, and the files also may contain comments starting with the character `#`.
- **`cub files:`** This file contains the information for three-dimensional grids. The format of these files is the standard `cub` format of Gaussian [7].
- **`gnp files:`** These files are scripts to be parsed to gnuplot. They purpose is to provide basic scripts with non-trivial information about the system that is being plotted. For instance, the positions of labels in the plot, values for the parametrization when lines or planes are projected, colors for the atoms etc.
- **`pov files:`** These type of files are the input for the pov ray-tracer.
- **`cpx files:`** This type of file is a native file of DensToolKit. It contains the information of critical points. These files act both as output and as input. For more details about this format, see §4.7.1.

4.2 Common options

All programs in DensToolKit share the following options:

- **Help menu:** Usage and some details of the program at hand is displayed by means of

  ```
  $dtk*** -h
  ```

  or

  ```
  $dtk*** -help
  ```

- **Version:** The version number can be accessed by
  ```
  $dtk*** -h
  ```
  or
  ```
  $dtk*** -version
  ```

- **Output name:** By using this option, the user can give a name other than the automatic name generated by each program. The program at hand will invariably add the appropriate extension to the file-name, and any extension will be consdered part of the name. This option is particularly useful when one wishes to give a wfx(wfn) and an out files located at arbitrary directories. For instance, the command

  ```
  $dtk*** /arbitrary/path/input.wfx -o /other/path/othername ...
  ```

  will invoke the program `dtk***` with an input wave function file located at `/arbitrary/path` and the output will be saved in the directory `/other/path` using the name `othername` with different extensions for the output.

- **Plotting the data:** Most of the DensToolKit's programs have the capability to create scripts that can be passed to plotting/rendering programs such as gnuplot or povray. While the primary purpose of DensToolKit is to evaluate numerical functions, we are aware that the visualization of the data is also important. With this in mind, and recalling that it is quite common the use of severs dedicated to Molecular Quantum Mechanical calculations, we have found that it is really convenient to be able to produce the visualization of data in a simple manner. If the program has the capability of calling plotting/rendering programs, such a call is activated with
  ```
  $dtk*** ... -P
  ```
  The programs that produce scripts (and internal callings) are: `dtkline`, `dtkplane`, `dtkfindcp`, `dtkmomd`, `dtkdemat1`, `dtkbpdens`, `dtkqdmol`

- **Compressing data:** Some programs have the option to call `gzip`. If this option is available, it can be activated with the following command line:
  ```
  $dtk*** ... -z
  ```
  The programs that provide this option are: `dtkline`, `dtkplane`, `dtkcube`, `dtkmomd`

4.3 dtkpoint

This program calculates all the implemented properties that can be calculated at a single point (see §2). Additionally, the program will perform the integration of the electron density over the whole space (which should render the molecule's total number of electrons). The program will take the wave function file and calculate the fields at the point

given by the user. The point(s) for the fields can be given in three different forms:

- **Cartesian coordinates**. The fields are calculated at the point (x, y, z). Example:

```
$dtkpoint ciclopropano_pbe6311.wfx -c 0.75e0 -5.0e-01 1.2e0
```

- **Set of Cartesian points.** The fields are calculated at a set of points given in an input file. For example, if the file `coords.dat` contains the coordinates, then the syntax is:

```
$dtkpoint ciclopropano_pbe6311.wfx -i coords.dat
```

  The file `coords.dat` contains the follwing data:

```
#Comments accepted by dtkpoint
1.1e0 2.04e-2 1.45e+1
2.3e0 -4.4e-2 3.31543e+00
#Another comment between coordinates
2.3e0 4.4e-2 -3.31543e+00
```

  Every line started with the character "`#`" will be ignored, and comments can be at any line of the input file. This program does not perform any check on the coordinates input file, therefore the user is responsible for providing a file that has $3N$ numbers, where N is the number of non-commented rows in the file. Each row corresponds to a point in the three dimensional space.

- **At a nucleus.** The fields are calculated at the position of a nucleus. Example:

```
$dtkpoint ciclopropano_pbe6311.wfx -a 2
```

  This will calculate the fields at the position of the 2^{nd} nucleus listed in `ciclopropano_pbe6311.wfx`.

`dtkpoint` help menu.

```
Usage:
```

```
     dtkpoint wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:

  -a a         Calculate the properties at the coordinates ofthe a-th atom.
  -c x y z     Calculate the properties at the point (x,y,z).
  -i crdfname  Calculate the properties at all points contained
                 in the file crdfname. The file must be a list of three
                 space-separated real numbers per row.
                 Please ensure the file has always the format specified above,
                 otherwise unpredictable errors may occur.
  -o outname   Set the output file name.
                 (If not given the program will create one out of
                 the input name.)
  -V           Displays the version of this program.
  -h           Display the help menu.

  --help          Same as -h
  --version       Same as -V
```

4.4 dtkline

This program calculates one of the implemented scalar fields (see §2) along a line defined by two of the molecule's atoms. For instance, the ELF along the line that passes through the atoms labeled as 2 and 4 of the cyclopropane molecule (Fig. 4.1(a)) is evaluated by means of

```
$dtkline ciclopropano_pbe6311.wfx -a 2 4 -p E -P
```

Here, the option `-a` tells `dtkline` to use the atoms `2` and `4` to define the line whereat the property will be evaluated; the option `-P` tells `dtkline` to create a pdf image using gnuplot; and with option `-p E`, one chooses the field ELF (see the help menu of `dtkline` for a complete list of fields —below—). The final result, after some editing of the gnuplot script is presented in Fig. 4.1(b).[2]

Sometimes, it is convenient to increase/decrease the number of points to evaluate within

[2]Let us recall that DensToolKit facilitates the creation of quality-plots (for instance, by providing the projected coordinates of the atoms C_2 and H_4 along the line), but it cannot replace the human judgement for creating nice plots.

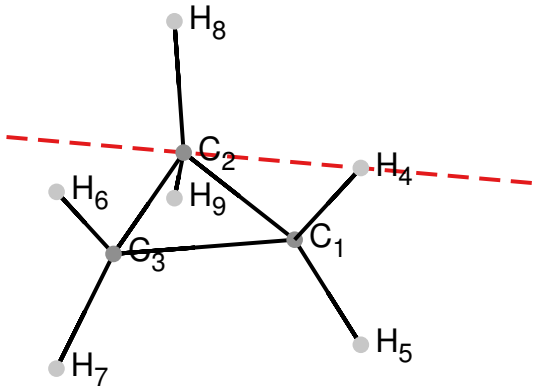
the line. To do so, one can use the option `-n dim`, where `dim` is the new number of points. If this option is not especified, `dtkline` uses 200 points.

Special cases:

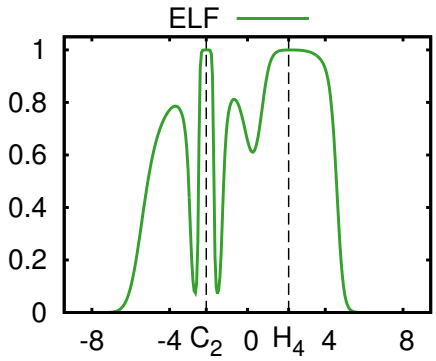
- **Wave function with one single atom:** In this case, `dtkline` will use the single atom present in the wave function file and then define a line that passes through this atom. The option `-a` is ignored, thus it is not necessary to use it, however if used the syntax must include two numbers. It is recommended, when you are interested in single atom wave functions, better not use the option `-a`.
- **Wave function with only two atoms:** For diatomic systems, `dtkline` will use these atoms to define the line, and if option `-a` is used, it will be ignored.

`dtkline` help menu.

```
Usage:

dtkline wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:

  -a a1 a2     Define the atoms  (a1,a2) used to define the line.
               If this option is not activated, the program will
               define the line using the first atom and the vector
               (1,1,1).
  -n  dim      Set the number of points for the dat file
  -o outname   Set the output file name.
```



(a) Cyclopropane molecule.



(b) ELF along the red line of Fig. 4.1(a).

Figure 4.1: Evaluating the ELF along the line that joins the atoms C_2 and H_4 with `dtkline`.

```
                (If not given the program will create one out of
                the input name; if given, the gnp file and the pdf will
                use this name as well --but different extension--).
  -p prop      Choose the property to be computed. prop is a character,
                which can be (d is the default value):
                  d (Density)
                  g (Magnitude of the Gradient of the Density)
                  l (Laplacian of density)
                  K (Kinetic Energy Density K)
                  G (Kinetic Energy Density G)
                  E (Electron Localization Function -ELF-)
                  L (Localized Orbital Locator -LOL-)
                  M (Magnitude of the Gradient of LOL)
                  S (Shannon Entropy Density)
                  V (Molecular Electrostatic Potential)
  -P        Create a plot using gnuplot.
  -k        Keeps the *.gnp file to be used later by gnuplot.
  -z        Compress the cube file using gzip (which must be intalled
              in your system).
  -V        Displays the version of this program.
  -h        Display the help menu.
  --help       Same as -h
  --version    Same as -V
*******************************************************************************
  Note that the following programs must be properly installed in your system:
                            gnuplot,epstopdf,gzip
*******************************************************************************
```

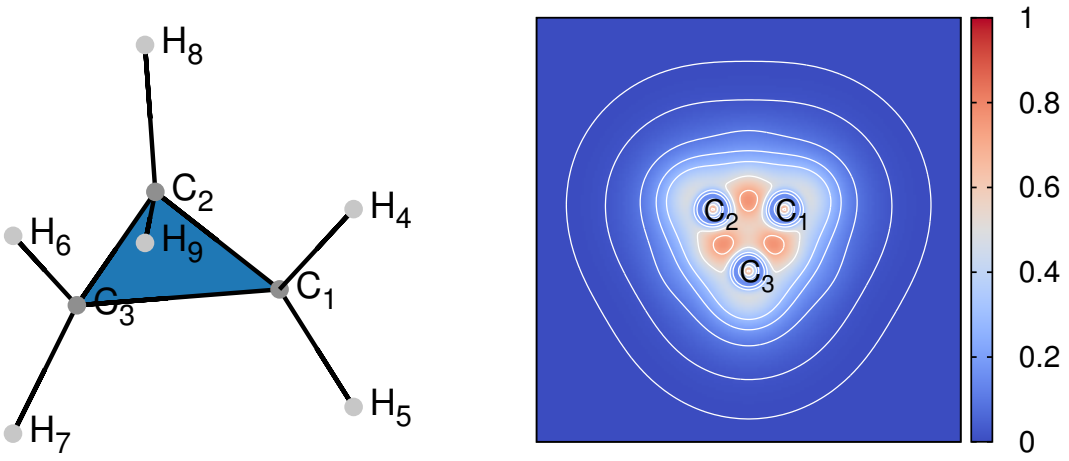
4.5 dtkplane

As the name suggests, this program evaluates a field upon a plane. `dtkplane` defines a plane based on the choosing of three atoms. Once the atoms are given, the program translates the plane until the centroid of the triangle that the three atoms form coincides with the center of a square. The size of the plane is calculated such that all the atoms in the molecule are contained in the plane, even if not all the atoms are within the plane. Since there is no restriction about the distance between the chosen atoms, it is assumed that the user is interested in studying the field around those atoms, therefore, the produced script only contains one parameter (called “dimparam”) to zoom in the plotted region.

As an example, the command line

```
$dtkplane ciclopropano_pbe6311.wfx -a 1 2 3 -p L -P -k -c -l
```

is the base for producing the figure 4.2(b).



(a) Cyclopropane molecule.

(b) LOL evaluated at the blue plane of Fig. 4.2(a).

Figure 4.2: Evaluating the LOL within the plane that contains the atoms C_1, C_2, and C_3 with `dtkplane`. The plot is rendered by gnuplot.

In the above command line, the option `-a 1 2 3` is used to provide the atoms that the plain will contain. `-p L` is the option for choosing the LOL field (see the menu help, below, for the complete list of fields); `-P` is for rendering a plot by calling gnuplot; `-k` for keeping the the gnuplot script on disk; `-c` activates the contours on the plane; and `-l` activates the labels of the atoms.

`-n`: For increasing/decreasing the number of points per direction, one can use the option `-n dim`. This will create a plane with `dim x dim` points.

`-c`: This option will print contour lines in the final plot.

`-l`: To add the labels of the atoms that were used to define the plane.

`-L`: This option will display the labels of every atom present in the wave function file. The positions are the projected position in the plane.

`-v`: This option will display information from third party programs such as gnuplot, gzip, etc.

Special cases:

- **Wave function with one single atom:** In this case, `dtkplane` will use the single atom present in the wave function file and then define a plane that passes through this atom. The option `-a` is ignored, thus it is not necessary to use it, however if

used the syntax must include three numbers. It is recommended, when you are interested in single atom wave functions, better not use the option `-a`.

- **Wave function with only two atoms:** For diatomic systems, `dtkplane` will use these atoms to define the plane, and if option `-a` is used, it will be ignored, however it must contain three numbers.

`dtkplane` help menu.

```
Usage:

     dtkplane wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:

  -a a1 a2 a3  Define the atoms  (a1,a2,a3) used to define the plane.
                 If this option is not activated, the program will
                 choose a default plane, but you may not like the view.
                 Note: if the *.wfn (*.wfx) file has only one or two atoms
                 this option must not be used. The program will define
                 a plane which includes that(those) one(two) atom(s).
  -n  dim      Set the number of points for the tsv file per direction
  -o outname   Set the output file name.
                 (If not given the program will create one out of
                 the input name; if given, the tsv, gnp and pdf files will
                 use this name as well --but different extension--).
  -p prop     Choose the property to be computed. prop is a character,
                 which can be (d is the default value):
                    d (Density)
                    g (Magnitude of the Gradient of the Density)
                    l (Laplacian of density)
                    K (Kinetic Energy Density K)
                    G (Kinetic Energy Density G)
                    E (Electron Localization Function -ELF-)
                    L (Localized Orbital Locator -LOL-)
                    M (Magnitude of the Gradient of LOL)
                    N (Gradient of LOL)
                    S (Shannon Entropy Density)
                    V (Molecular Electrostatic Potential)
  -P          Create a plot using gnuplot.
  -c          Show contour lines in the plot.
  -k          Keeps the *.gnp file to be used later by gnuplot.
  -l          Show labels of atoms (those set in option -a) in the plot.
  -L          Show the labels of ALL of the atoms in the wf? file.
  -v          Verbose (display extra information, usually output from third-
                 party sofware such as gnuplot, etc.)
  -z          Compress the tsv file using gzip (which must be intalled
```

```
                in your system).
  -V          Displays the version of this program.
  -h          Display the help menu.
  --help         Same as -h
  --version      Same as -V
*******************************************************************************
  Note that the following programs must be properly installed in your system:
                       gnuplot, epstool, epstopdf, gzip
*******************************************************************************
```

4.6 dtkcube

This program evaluates a scalar field within a three dimensional grid. The data is saved in a *.cub file, which is the standard cube file from the Gaussian [7] package. This is perhaps the most computationally expensive program in DensToolKit, and also one of the few programs that does not produce any visualization.

The purpose of this program is to create standard output that can be processed by advanced visualization programs such as VMD. For instance, with the command line

```
$dtkcube ciclopropano_pbe6311.wfx -p l -S 100
```

we produce the cub file that later is used in VMD to create the Figure 4.3. The option

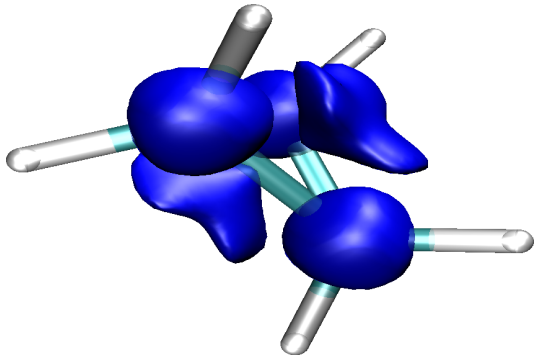


Figure 4.3: Evaluating $\nabla^2\rho$ within a three-dimensional grid (cube). The plot was generated using VMD, and later povray.

`-p l` tells `dtkcube` to evaluate the Laplacian of the electron density (see below for the complete list of available fields), and `-S 100` activates a *smart* cube. By smart cube, we mean that we try to minimize the number of trivial points, this is, we adjust the cube to be

a rectangular cuboid such that it encloses the molecule and a few extra space around the most outer atoms of the molecule. This allows DensToolKit to skip the calculation of spatial points that offer only trivial information (since the wave function and its derivatives decay rapidly as the point moves away from the molecule). For instance, the cuboid generated by the example command generates a grid of `96 x 100 x 90 = 864,000` points, which is 86.4 % of the points that would have been calculated if a normal cube had been evaluated.

`dtkcube` always creates the grid around the geometric center of the molecule. This also reduces the number of trivial points in the final cube/cuboid.

If you want to save the information about wfx input file name, CPU time (time taken to evaluate the whole grid), and some other extra-information, you may want to activate option `-l`.

4.6.1 Managing the dimensions of the grid

`dtkcube` offers several options for custom requests:

- **Default cube:** A default centered cube of `80 x 80 x 80` points is evaluated if no additional options are given.
- **`-n dim`**: A centered cube of `dim x dim x dim` points is evaluated.
- **`-N nx ny nz`**: A centered cube of `nx x ny x nz` points is evaluated.
- **`-s`**: A *smart* cuboid with the largest dimension being `80` is evaluated. The other dimensions are proportional to the molecule's dimensions.
- **`-S ldim`**: A *smart* cuboid with the largest dimension being `ldim` is evaluated. The other dimensions are proportional to the molecule's dimensions.

In the current version, `dtkcube` does not rotate the molecule to minimize the cuboid dimensions, but only looks for the highest and lower values of the atoms' coordinates in all of the three axis. These values are the ones used to adjust the cuboid.

`dtkcube` help menu.

```
Usage:

dtkcube wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:
```

```
-l           Write cpu time, input/output information etc. on a log file
-n  dim      Set the number of points per direction for the cube
                to be dim x dim x dim.
-N nx ny nz Set the individual points per direction for the cube
                to be nx x ny x nz.
-o outname   Set the output file name.
-s           Use a smart cuboid for the grid. The number of points for the
                largest direction will be 80.
-S ln        Use a smart cuboid for the grid. ln is the number of points
                the largest axis will have. The remaining axes will have
                a number of points proportional to its length.
-p prop      Choose the property to be computed. prop is a character,
                which can be (d is the default value):
                   d (Density)
                   g (Magnitude of the Gradient of the Density)
                   l (Laplacian of density)
                   K (Kinetic Energy Density K)
                   G (Kinetic Energy Density G)
                   E (Electron Localization Function -ELF-)
                   L (Localized Orbital Locator -LOL-)
                   M (Magnitude of the gradient of LOL)
                   S (Shannon Entropy Density)
                   V (Molecular Electrostatic Potential)
-z           Compress the cube file using gzip (which must be intalled
                in your system).
-V           Displays the version of this program.
-h          Display the help menu.
--help          Same as -h
--version       Same as -V
```

4.7 dtkfindcp

This program search critical points (CP) within a molecule. In version 1.0.0, the complete analysis is implemented for the electron density (ED). Partial implementation is offered for searching LOL critical points. In the future, additional fields will be implemented.

The complete search of critical points of a molecule can be found by the following command:

```
$dtkfindcp ciclopropano_pbe6311.wfx
```

This will find all the ED critical points (including ACPs, BCPs, RCPs, and CCPs), as well

as the bond gradient paths (paths that connect ACPs with BCPs). The information of the critical points are saved into two files. The `log` file contains the coordinates of all CPs found, and the field properties at such points. The `cpx` file shares the coordinates of the CPs, but additional information is stored in these files, such as the coordinates of the gradient bond paths, and the name of the wave function used to perform the search of CPs.

`dtkfindcp` is also capable of producing `pov` files (and internal callings to povray) for rendering 3D images of the critical points. For instance, the Figure was produced by typing

```
$dtkfindcp ciclopropano_pbe6311.wfx -P -g -T -k
```

This command produces a `pov` file and calls povray to render the image displayed in Fig. 4.4. The option `-g` tells `dtkfindcp` to display the bond gradient paths; and the option `-T` sets the style of the bond gradient paths to be tubes.

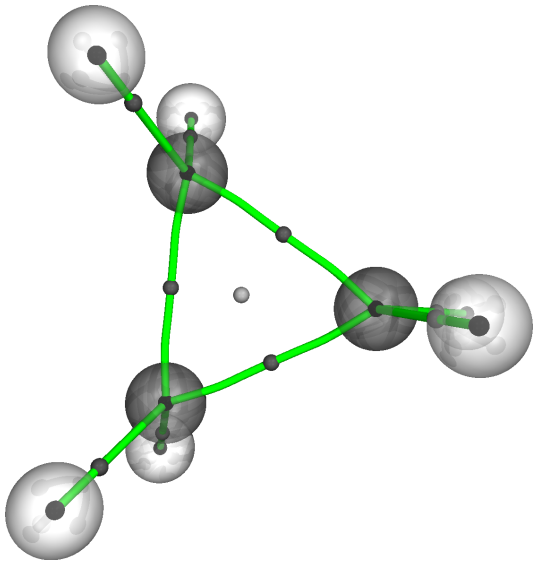


Figure 4.4: Electron density critical points of the cyclopropane molecule. The green lines are the bond paths joining the BCPs and the ACPs.

If you are interested only in finding the critical points, but you do not want to seek the gradient bond paths, you can deactivated the search of bond paths by using the option `-G`.

If you do not wish to create a `png` image, but you do want the `pov` file, using the option `-p` will accomplish that.

In addition to the cpx file, you can also save the coordinates of the critical points in a very simple format. Using the option `-m`, the coordinates of the critical points will be written into three different files. The names of these files will be created out of the `wfx(wfn)` file, or out of the output file name especified by using the option `-o outputname`. Different string will be added to the out names to identify which information they contain, and the files will obey the following formats:

- The file ending with "-ATCrds.dat" will contain the atomic coordinates, and the format of such file is
  `AtomicSymbol  AtomicNumber  X  Y  Z`
- The file ending with "-CPCrds.dat" will contain the coordinates of all the critical points that were found. The format of this file is
  `???  X  Y  Z`
  where `???` can be "`acp`", "`bcp`", "`rcp`" or "`ccp`" to identify the signature of the corresponding CP.
- The file ending with "-BPCrds.dat" will contain the coordinates of the points of all bond paths, without further identification. This is, the coordinates are points that belong to a bond path, but there is no way to identify to which specific bond path the point at hand belongs to. The format is simply
  `X  Y  Z`

Some of the options for calling povray can be set in the command line. For instance, the camera position can be handled by using the option `-c p`, where `p` is an integer that gives the normal vector components where the camera will be placed at. `p` can be `001,010,100,101,110,111`. Every value is a three digit number that represents the components of a vector, for instance `101` means that the camera will be placed at a position proportional to the vector (1,0,1), and so on. Warning: this option is not really compatible with the custom angle views that can be modified in the pov file (see §4.7.2 for more details), and perhaps will be deleted in future implementations.

If you wish to create a `png` image, and keep the `pov` file all at once, you need to especify the option `-k` for not deleting the pov file.

Below you can find a brief description of the cpx file format as well as a short overview of the custom options that can be easily modified in the pov files which, in junction with the script `dtkpov2png`, can be used to create custom images of the displayed properties.

Finally, in future implementations there will be the possibility to search critical points of different fields. In version 1.0.0, the LOL is partially implemented.

`dtkfindcp` help menu.

```
Usage:

dtkfindcp wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:

  -a           Draw transparent spheres around each nuclei in the wf? file.
  -o outname   Set the output file name (*.log).
                  (If not given the program will create one out of
                  the input name; if given, the pov file will
                  use this name as well --but different extension--).
  -m           Save the coordinates of atoms, critical points and gradient
                  paths (if calculated, see option -G) into *dat files.
  -p           Create a pov file. No image will be generated.
  -P           Create a png image using povray (calls povray),
                  deletes pov file.
  -c p         Set the camera vector direction for the povray scene.
                  p is an integer which can take the values
                  001,010,100,101,110,111; the camera will be located
                  somewhere in the direction (x,y,z), where xyz is one of
                  the above values.
  -g           Draw the bond paths (gradient paths) in the POV file.
  -T           Set the style for the gradient paths to be tubes.
  -G           Skip the calculation of the bond gradient paths.
  -k           Keeps the pov file if option -P is used.
  -t cpt       Set the type of critical point to be searched. cpt can be
                  one of the following fields (Density is the default):
                     d (Density)
                     L (Localized Orbital Locator -LOL-)
  -v           Verbose mode (displays additional information (for example the
                  output of povray, etc.
  -V           Display the version of this program.
  -h          Display the help menu.
  --help          Same as -h
  --version       Same as -V
*******************************************************************************
  Note that the following programs must be properly installed in your system:
                       povray, graphicsmagick, gzip
*******************************************************************************
```

4.7.1 The cpx file format

We follow the same conventions as for the wfx file format [10]. A cpx file is somewhat similar to an XML file, but with certain restrictions. We define opening and closing tabs, for instance

```
<WaveFunctionFileName>
 h2o.wfx
</WaveFunctionFileName>
```

or

```
<CriticalPointType>
 Electron Density
</CriticalPointType>
```

or

```
<NumberOfACPs>
 3
</NumberOfACPs>
```

However, any opening or closing tag MUST stand alone in a single line. The purpose of this restriction is to accelerate the reading of the cpx file. This format does not contain tags with spaces.

For the purposes of the data (information between tags) a new line is considered as equivalent to a space or a tab character. There can be comments between two different tags, but comments between the data is not allowed.

Below we provide an example of a cpx file, which is the result of a search of critical points of the water molecule.

#**WFX (WFN) file name** from which the the wave function of the molecule was obtained.

```
<WaveFunctionFileName>
 h2o.wfx
</WaveFunctionFileName>
```

#**Type of critical points:** This refers to the type of critical points, in terms of the scalar field. This is, this tag identifies whether the critical points are electron density critical points, LOL critical points, etc.

```
<CriticalPointType>
 Electron Density
</CriticalPointType>
```

#**Number of critical points:** The total numbers of critical points by signature identification. We follow the conventional naming:

- *Attractor Critical Point* **(ACP)** is a CP with signature (3,-3), and corresponds to a maximum in the topology of the Fields manifold.

- *Bond Critical Point* **(BCP)** is a CP with signature (3,-1), and corresponds to a saddle point where two of the Hessian eigenvalues are negative, and the third is positive.
- *Ring Critical Point* **(RCP)** is a CP with signature (3,+1), and corresponds to a saddle point where only one of the Hessian eigenvalues is negative and the other two are positive.
- *Cage Critical Point* **(CCP)** is a CP with signature (3+3), and corresponds to a local minimum in the field's manifold.

We are aware that the terms ACP, BCP, etc. loose their physical meaning when fields other than the electron density are analysed. However, the signature values remain the same, therefore this naming can be used for uniquely tagging a critical point.

```
<NumberOfCriticalPoints>
<NumberOfACPs>
 3
</NumberOfACPs>
<NumberOfBCPs>
 2
</NumberOfBCPs>
<NumberOfRCPs>
 0
</NumberOfRCPs>
<NumberOfCCPs>
 0
</NumberOfCCPs>
</NumberOfCriticalPoints>
```

#**CP Cartesian Coordinates**: The Cartesian coordinates of all critical points with a given signature. If there is no critical point of certain signature, the opening/closing tags will appear and there will be no data among the tags.

```
<ACPCartesianCoordinates>
  7.405253164682e-34 -1.481050632936e-33  2.403142046477e-01
  0.000000000000e+00  1.432892399165e+00 -9.612568185908e-01
 -1.754787090160e-16 -1.432892399165e+00 -9.612568185908e-01
</ACPCartesianCoordinates>
<BCPCartesianCoordinates>
  3.879598339682e-19  1.056340227248e+00 -6.465448941267e-01
 -1.302800283929e-16 -1.056340227248e+00 -6.465448941267e-01
</BCPCartesianCoordinates>
<RCPCartesianCoordinates>
</RCPCartesianCoordinates>
```

```
<CCPCartesianCoordinates>
</CCPCartesianCoordinates>
```

#**Bond critical points connectivity**. This block contains the information of how the network of BCPs are related to the ACPS. This makes more sense for the electron density CPs, since for this case, a BCP is usually found whenever two atoms are bonded. For other fields, the rule of starting the search of a BCP in between every two ACPs generally applies, thus this information saves the hystory of where the seed for looking the BCP at hand was located.

```
<BCPConnectivity>
1 2 1 2
2 2 1 3
</BCPConnectivity>
```

#**Labels**: This set of tags stores the labels of all the critical points of a certain signature. For the electron density ACPs, we use the atom labels provided in the `wfx(wfn)` file. For other fields, we use also the atom label plus an extra label after the numerical index. The BCPs are labeled according to the ACPs that were used to start the search (the midpoint between each pair of ACPs). The RCPs' labeling involvels all the atoms that were related to its search. And the same for the CCPs. Once again, if the number of CPs of a certain signature is zero, the opening/closing tags will be present, but there will be no data among these tags.

It is assumed that each label consists of a set of alphanumeric characters and hyphens. However, spaces in a label are NOT allowed. In fact for this tag, a space, a tab character, or an end line, all of them indicate the final of a label.

```
<ACPLabels>
 O1 H2 H3
</ACPLabels>
<BCPLabels>
 O1-H2 O1-H3
</BCPLabels>
<RCPLabels>
</RCPLabels>
<CCPLabels>
</CCPLabels>
```

#**Number of bond paths**: Self descriptive.

```
<NumberOfBondPaths>
 2
</NumberOfBondPaths>
```

#**Number of points per bond path:** There is no *a priori* way to know how many points will be per each bond path, since they will vary in length and curvature. Therefore, the number of points that belong to each bond path is set in this tag. Notice that we already know how many bond paths were found (see above, "NumberOfBondPaths").

```
<NumbersOfPointsPerBondPath>
 21 21
</NumbersOfPointsPerBondPath>
```

#**Bond paths' data**: This block contains the coordinates of each point that belongs to a given bond path. The data containts two child tags "BondPathIndex" which is a number to identify the bond path, and is given just as a redundant mechanism to check the integrity of the cpx file; and "CoordinatesOfBondPathPoints" which contains the actual coordinates of each point in the bond path identified by "BondPathIndex".

```
<BondPathsData>
<BondPathIndex>
 0
</BondPathIndex>
<CoordinatesOfBondPathPoints>
  1.283310240215e-18  1.325322904480e+00 -8.521158631320e-01
 -3.114719658371e-18  1.277560298804e+00 -8.844656336821e-01
  2.366717289600e-19  1.331886275170e+00 -8.749446714881e-01
  2.187505553771e-19  1.286998381318e+00 -8.383698088888e-01
  2.768727742932e-19  1.210147039589e+00 -7.743864478328e-01
  3.249714975667e-19  1.133255359299e+00 -7.104515640576e-01
  3.879598339682e-19  1.056340227248e+00 -6.465448941267e-01
  4.509481703697e-19  9.794250951975e-01 -5.826382241959e-01
  5.009611731128e-19  9.025741185897e-01 -5.186544264342e-01
  5.583774862697e-19  8.258317118277e-01 -4.545404760932e-01
  6.244561433914e-19  7.492696478497e-01 -3.902113108375e-01
  7.061472903938e-19  6.729180248149e-01 -3.256325051397e-01
  8.204186617202e-19  5.966597044907e-01 -2.609434996374e-01
  1.004820096952e-18  5.199999203721e-01 -1.967319120013e-01
  1.281126284848e-18  4.419521608625e-01 -1.342189650488e-01
  1.460104437383e-18  3.628958273391e-01 -7.298135325864e-02
  1.328768819774e-18  2.852253826867e-01 -1.000206373725e-02
  1.016984895739e-18  2.092544123305e-01  5.502203677181e-02
  6.596232316936e-19  1.340530676658e-01  1.209364863332e-01
  2.926802524295e-19  5.925397145290e-02  1.873072163197e-01
  7.405253164682e-34 -1.481050632936e-33  2.403142046477e-01
</CoordinatesOfBondPathPoints>
<BondPathIndex>
```

```
 1
</BondPathIndex>
<CoordinatesOfBondPathPoints>
  7.405253164682e-34 -1.481050632936e-33  2.403142046477e-01
 -2.771493234187e-17 -5.925397145290e-02  1.873072163197e-01
 -6.250711727050e-17 -1.340530676658e-01  1.209364863332e-01
 -9.650047009393e-17 -2.092544123305e-01  5.502203677181e-02
 -1.265815303299e-16 -2.852253826867e-01 -1.000206373725e-02
 -1.413403025249e-16 -3.628958273391e-01 -7.298135325864e-02
 -1.307349355991e-16 -4.419521608625e-01 -1.342189650488e-01
 -1.138199423562e-16 -5.199999203721e-01 -1.967319120013e-01
 -1.058905765908e-16 -5.966597044907e-01 -2.609434996374e-01
 -1.046062180849e-16 -6.729180248149e-01 -3.256325051397e-01
 -1.067544859534e-16 -7.492696478497e-01 -3.902113108375e-01
 -1.109011672163e-16 -8.258317118277e-01 -4.545404760932e-01
 -1.164060684268e-16 -9.025741185897e-01 -5.186544264342e-01
 -1.229057174906e-16 -9.794250951975e-01 -5.826382241959e-01
 -1.302800283929e-16 -1.056340227248e+00 -6.465448941267e-01
 -1.376543392952e-16 -1.133255359299e+00 -7.104515640576e-01
 -1.466480850572e-16 -1.210147039589e+00 -7.743864478328e-01
 -1.557516934400e-16 -1.286998381318e+00 -8.383698088888e-01
 -1.583008836928e-16 -1.331886275170e+00 -8.749446714881e-01
 -3.405682644852e-16 -1.277560298804e+00 -8.844656336821e-01
 -9.982455406799e-17 -1.325322904480e+00 -8.521158631320e-01
</CoordinatesOfBondPathPoints>
</BondPathsData>
```

4.7.2 Options in the `pov` file

Every `pov` file created by `dtkfindcp` has the following header:

```
#version 3.6; //Unless you know what you are doing, do not modify this line...
#include "colors.inc"
////////////////////////////////////////////////////////////////////////////
//
//Below you can find some options to be parsed to povray
//set your custom values.
//You can reconstruct the image using the script dtkpov2png
//
////////////////////////////////////////////////////////////////////////////
#declare GNUPlotAngle1=0;
#declare GNUPlotAngle2=0;
```

```
#declare YAngle=0;
#declare DrawAtomTranspSpheres=false;
#declare DrawStandardBonds=false;
#declare DrawAttractorCriticalPoints=true;
#declare DrawBondCriticalPoints=true;
#declare DrawRingCriticalPoints=true;
#declare DrawCageCriticalPoints=true;
#declare DrawGradientPathSpheres=false;
#declare DrawGradientPathTubes=true;
//  Activation of "DrawGradientPathSpheres" requires deactivation of
//  "DrawGradientPathTubes", and vice versa.
#declare RadiusAllCriticalPoints=0.1;
#declare ColorACP=rgb <0.0,0.0,0.0>;
#declare RadiusACP=RadiusAllCriticalPoints;
#declare ColorBCP=rgb <0.3,0.3,0.3>;
#declare RadiusBCP=RadiusAllCriticalPoints;
#declare ColorRCP=rgb <0.6,0.6,0.6>;
#declare RadiusRCP=RadiusAllCriticalPoints;
#declare ColorCCP=rgb <0.9,0.9,0.9>;
#declare RadiusCCP=RadiusAllCriticalPoints;
#declare ColorABGradPath=rgb <0.0,1.0,0.0>;
#default { finish { specular 0.3 roughness 0.03 phong .1 } }
////////////////////////////////////////////////////////////////////////
//For the colors, instead of rgb <...>, you may want to try Red, Yellow, ...
//  or any of the colors defined in "colors.inc"
//
////////////////////////////////////////////////////////////////////////
// END OF CUSTOM OPTIONS
////////////////////////////////////////////////////////////////////////
```

Getting the right position of the povray's camera is a bit tricky, and we do not attemt to solve this problem in an automatic way. However, we provide a relatively easy-to-use tool to find an adequate view. Using the program `dtkqdmol`, you can create a quick view of the molecule, and since it is rendered via gnuplot, two view angles are displayed in the window (see §4.11). You can adjust the view through the interactive gnuplot's window and then change the values of the defined variables `GNUPlotAngle1`, and `GNUPlotAngle1`. An extra angle can be specified in povray (which gnuplot does not use): the `YAngle` variable. For more information about this view angles, see the povray's documentation (under the "camera's rotate" section) or the gnuplot's help menu (`help set view`).

By default, all critical points are drawn with the same size. For setting different sizes, you can adjust the options "`RadiusACP=RadiusAllCriticalPoints;`" to you desired value. (Changing only the value of `RadiusAllCriticalPoints` will change the size of ALL critical points.)

The colors of the critical points are given by three numbers, correspoding to the red, green or blue saturation value, and they all must be numbers between 0 and 1.

The rest of the options are self descriptive, and the only note is to use `true` or `false`.

4.8 dtkmomd

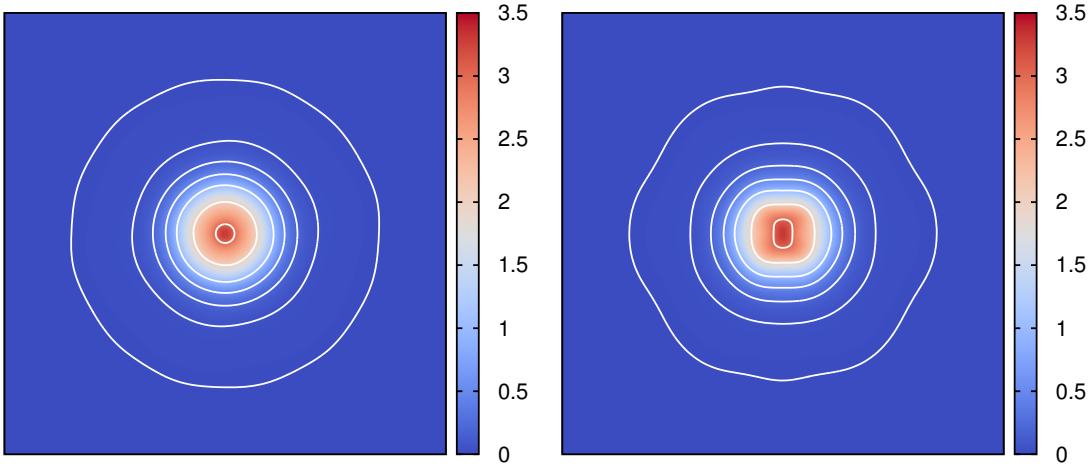
This program evaluates the electron momentum density (EMD), *i.e.,* the Fourier transform of the electron density. This single program may evaluate the momentum density for a single point; one, two and three dimensional grids. The output, if any, is written as explaneid below.

- **Point (0D)**: No output is created, instead the value is displayed at the screen. This grid is set by the option `-0 P_x P_y P_z`, where `P_i` are the values at which the EMD is evaluated.
- **Line (1D)**: The output is written to a `dat` file and, in the current version, only lines parallel to the axis are implemented. This grid is accessible through the option `-1 x`, where "x" indicates the axis to be evaluated and can take the values "x", "y" or "z". Creation of gnuplot scritps is enabled for this grid (option `-P`).
- Plane (2D): The data is saved into a `tsv` file, and in the present version, only planes normal to the axis are implemented (the planes contain the origin of the momentum space coordinates). To evaluate the EMD in a plane, one must use the option `-2 xy`, where `xy` are two non-spaced characters that indicates the plane whereat the EMD will be evaluated, and can be `xy`, `xz`, or `yz`. Creation of gnuplot scritps is also enabled for this grid (option `-P`).
- Cube (3D): The values of the EMD are stored in a `cub` file (standard Gaussian cube file), and this grid is set by using option `-3`. For this grid, there is no immediate visualization, but it needs to be produced through other programs such as VMD.

`dtkfindcp` help menu.

```
Usage:

dtkmomd wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:

  -0 x y z     Evaluate the momentum density at the point (x,y,z).
                  Here x,y, and z are numbers. No file is created.
  -1  x        Evaluate the momentum density on a line. "x" sets the coordinate
                  direction, and can take the values x, y or z.
                  Here x is a character. It creates a *.dat file.
```

```
  -2  xy       Evaluate the momentum density on a plane. "xy" sets the plane of
                  interest, and can take the values xy,xz,yz.
                  Here xy are two characters. It creates a *.tsv file.
  -3           Evaluate the momentum density on a cube.
                  It creates a *.cub file.
  -n  dim      Set the number of points for the cub/tsv/dat file per direction
  -o outname   Set the output file name.
                  (If not given the program will create one out of
                  the input name; if given, the dat/tsv/cub/gnp/pdf files will
                  use this name as well --but different extension--).
  -P           Create a plot using gnuplot. (Only works with options -1 or -2)
  -k           Keeps the *.gnp file to be used later by gnuplot.
  -v           Verbose (display extra information, usually output from third-
                  party sofware such as gnuplot, etc.)
  -z           Compress the tsv file using gzip (which must be intalled
                  in your system).
  -V           Displays the version of this program.
  -h           Display the help menu.
  --help          Same as -h
  --version       Same as -V
*******************************************************************************
  Note that the following programs must be properly installed in your system:
                    gnuplot, epstool, epstopdf, gzip
*******************************************************************************
```



(a) Plane P_x-P_y (b) Plane P_y-P_z

Figure 4.5: Evaluating the momentum density of the cyclopropane molecue on two-dimensional grids (planes).

4.9 dtkdemat1

This program evaluates the density matrix of order 1 (DM1) along a line, which can be the line that joins two atoms, or the bond path that connects those two ACPs. In this version, only the electron density bond path can be selected.

`dtkdemat1` can be called, for example, by

```
$dtkdemat1 ciclopropano_pbe6311.wfx -P -l -c -s 0.02
```

The above command was used for evaluating the data shown in Fig. 4.6. The output of the program consists on four files. The data for generating the three-dimensional and two-dimensional plots is saved in a `tsv` file. The data for the main and secondary diagonal (white solid and dashed line of figure 4.6(b)) are saved in two `dat` files. In addition, a `log` file is created, where some information about the position of the critical point (or the minimum of the ED) is located, etc.

As usual, the option `-P` requests the creation of plots. This program generates four of them. The names of the plots use the input wave function file name, adds the characters `BP` (if the bond path is selected), or `SL` if the straight line that joins two atoms (especified with the option `-a` a_1 a_2, see below), the labels of the selected atoms, and two or three characters to indicate the specific data displayed in the plot. The DM1 information is displayed as a surface (its name ends with `3D`), as a heat map (whose name ends with `2D`), and two simple plots that contains the DM1 values at the main diagonal (solid white line in Figure 4.6(b), and whose name ends with `1D1`) and at the secondary diagonal (dashed white line in Figure 4.6(b), and whose name ends with `1D2`).

The option `-l` makes `dtkdemat1` to draw the labels of the atoms for the two dimensional and one-dimensional plots, while the option `-c` draws the contour lines as shown, for example, in Figure 4.6(b).

By default, `dtkdemat1` uses the first and second atoms of the `wfx/wfn` file. For selecting different atoms, you should use the option `-a` a_1 a_2, where a_1, and a_2 are the desired atoms.

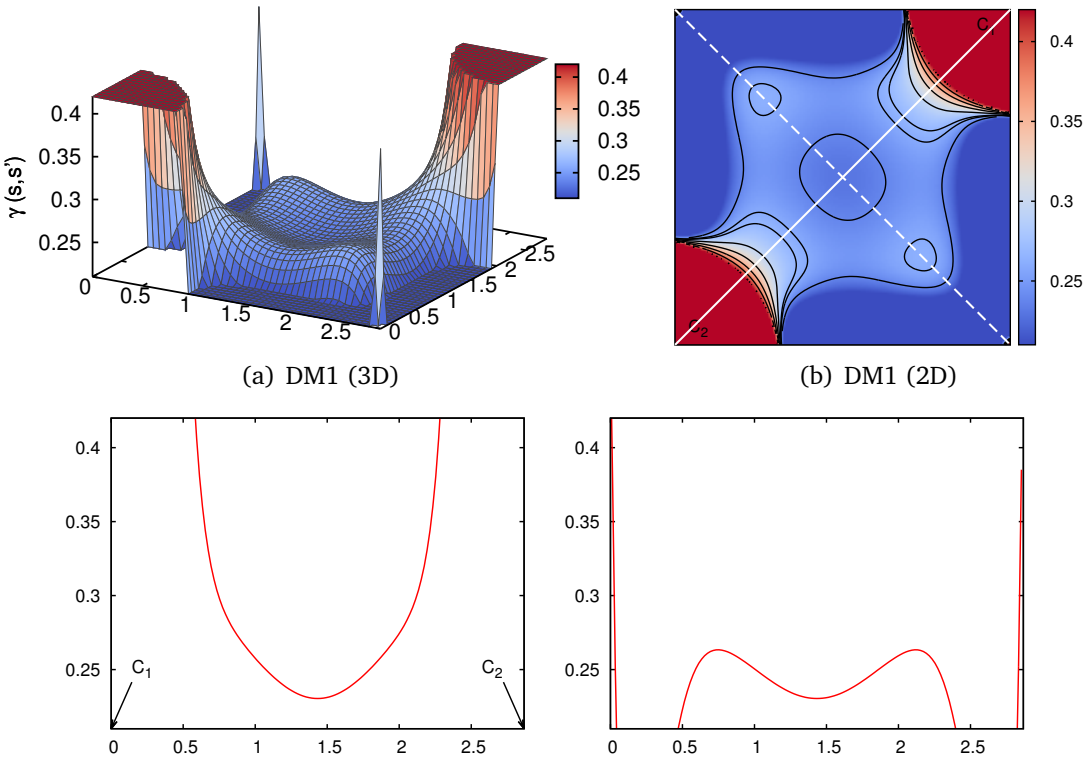
As well, by default, the program will calculate the bond path between the atoms, and then use this curve to evaluate the DM1. However, in some cases is needed to evaluate the DM1 along the straight line that joins the atoms, which can be requested with the switch `-L`.

For increasing the number of points within the bond path line, you may want to look for a combination of the option `-s step` and `-n dim`. Here `step` orders `dtkdemat` to use

the step `step` for the searching of the bond path (this is the step used in the Runge-Kutta integrator), while `dim` is the dimension of an array that contains the maximum number of points contained in the bond path. By default, `dtkdemat1` sets `step=0.03`, and `dim=200`. When requesting to evaluate DM1 over the line that joins the atoms, `step` is no longer significant, rather the line will contain `dim` points.

`dtkfindcp` help menu.

```
Usage:

dtkdemat1 wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:

  -a a1 a2     Define the atoms  (a1,a2) used to define bond path/line.
                  If this option is not activated, the program will
```



(a) DM1 (3D)

(b) DM1 (2D)

(c) DM1 (1D1) along the main diagonal (white solid line in (b)).

(d) DM1 (1D2) along the secondary diagonal (white dashed line in (b)).

Figure 4.6: Evaluating the density matrix of order 1 with `dtkdemat1`.

```
                set a1=1, a2=2.
                Note: if the *.wfn (*.wfx) file has only one atom
                the program will exit and no output will be generated.
  -L            Calculate DM1 upon the straight line that joins the atoms
                instead of upon the bond path.
  -n  dim       Set the number of points for the tsv file per direction.
                Note: for the bond path you may want to look for a good
                combination of n and the number "step" given in option -s,
                since the number of points in the bond path will be mainly
                governed by step.
  -o outname    Set the output file name.
                (If not given the program will create one out of
                the input name; if given, the tsv, gnp and pdf files will
                use this name as well --but different extension--).
  -s step       Set the stepsize for the bond path to be 'step'.
                Default value: 0.03
  -P            Create a plot using gnuplot.
  -c            Show contour lines in the plot.
  -l            Show labels of atoms (those set in option -a) in the plot.
  -v            Verbose (display extra information, usually output from third-
                party sofware such as gnuplot, etc.)
  -z            Compress the tsv file using gzip (which must be intalled
                in your system).
  -h            Display the help menu.
  -V            Displays the version of this program.
  --help        Same as -h
  --version     Same as -V

*******************************************************************************
  Note that the following programs must be properly installed in your system:
                    gnuplot, epstool, epstopdf, gzip
*******************************************************************************
```

4.10 dtkbpdens

This program seeks the bond path between two atoms and then evaluate the requested scalar/vector field at the points belonging to the bond path. This is specially meaningfull when the bond path is not the straight line that joins the atoms, and especially in cases where they differ considerable such as in hydrogen bonds.

The data for producing Figure 4.7 was obtained by typing the two following commands.

```
$dtkbpdens ciclopropano_pbe6311.wfx -P -l -s 0.02 -p M
```

```
$dtkbpdens ciclopropano_pbe6311.wfx -P -l -n 200 -p M -L
```

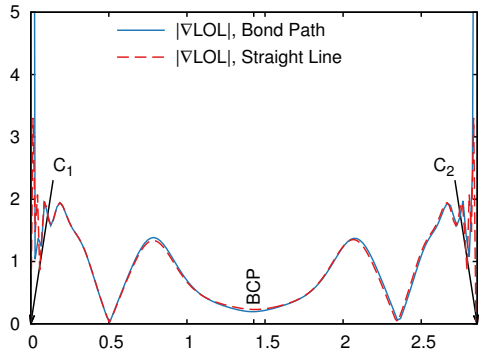


Figure 4.7: Evaluating the magnitude of LOL along the bond path

Plots will be generated by using option `-P`, and the labels of the atoms will be added activating the switch `-l`.

The step for the Runge-Kutta integrator can be set with the option `s step`, and the maximum number of points for the bond path is set with option `-n dim`. Here `dim` is the number of points requested for the arrays that contains the coordinates of the points belonging to the bond path.

If the property is evaluated at the line that join the atoms (activating the switch `-L`), then the variable `step` is no longer used (whose default value is `step=0.03`), and instead the value of `dim` is used to divide the line (the default value for this variable is `dim=200`). The output will differ from the data obtained with `dtkline` in the length of the line: `dtkbpdens` evaluates a field between the atoms, while `dtkline` evaluates the property upon a longer line that passes through the same atoms.

To choose the scalar field to plot, one simply uses the option `-p c`, where `c` is a character that can be one of those included in the list shown in the menu (below).

`dtkfindcp` help menu.

```
Usage:

dtkbpdens wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:
```

```
  -a a1 a2     Define the atoms  (a1,a2) used to define bond path/line.
                  If this option is not activated, the program will
                  set a1=1, a2=2.
                  Note: if the *.wfn (*.wfx) file has only one atom
                  the program will exit and no output will be generated.
  -L           Calculate the field upon the straight line that joins the atoms
                  instead of upon the bond path.
  -n  dim      Set the number of points for the dat file.
                  Note: for the bond path you may want to look for a good
                  combination of n and the number "step" given in option -s,
                  since the number of points in the bond path will be mainly
                  governed by step.
  -o outname   Set the output file name.
                  (If not given the program will create one out of
                  the input name; if given, the dat, gnp and (eps)pdf files will
                  use this name as well --but different extension--).
  -p prop      Choose the property to be computed. prop is a character,
                  which can be (d is the default value):
                     d (Density)
                     g (Magnitude of the Gradient of the Density)
                     l (Laplacian of density)
                     K (Kinetic Energy Density K)
                     G (Kinetic Energy Density G)
                     E (Electron Localization Function -ELF-)
                     L (Localized Orbital Locator -LOL-)
                     M (Magnitude of the Gradient of LOL)
                     S (Shannon Entropy Density)
                     V (Molecular Electrostatic Potential)
  -s step      Set the stepsize for the bond path to be 'step'.
                     Default value: 0.03
  -P           Create a plot using gnuplot.
  -l           Show labels of atoms (those set in option -a) in the plot.
  -v           Verbose (display extra information, usually output from third-
                  party sofware such as gnuplot, etc.)
  -z           Compress the dat file using gzip (which must be intalled
                  in your system).
  -h           Display the help menu.
  -V           Displays the version of this program.
  --help          Same as -h
  --version       Same as -V

-------------------------------------------------------------------------------
        The format of the dat file is:
           L  X  Y  Z  V
        where L is the value of the parameter that maps the bond
        path to a line; X, Y, and Z are the actual spatial coordinates
        of each point in the bond path; and V is the value of the
        chosen field at the point (X,Y,Z) ---see option -p.
-------------------------------------------------------------------------------
*******************************************************************************
```

```
  Note that the following programs must be properly installed in your system:
                          gnuplot, epstopdf, gzip
*******************************************************************************
```

As we can see in the help menu, the output file has a distinctive format. It contains the value of the variable used for the parametrization of the bond path (or straight line), the actual coordinates in the Cartesian system wherein the molecule is actually embedded, and finally the requested field.

4.11 dtkqdmol

The purpose of this program is to provide a quick view of the molecule through the gnuplot interactive terminal. This is particularly useful for viewing the labels of the atoms, and to produce simple diagrams, such as the Figures 4.1(a) and 4.2(a).

`dtkqdmol` is *not* intended to replace more powerful visualization tools such as JMol, or the viewers included in the *ab initio* calculation programs, but to serve as a command line program to improve the workflow of analysing and evaluating properties, and also to adjust the view angles to be parsed to povray (see `dtkfindcp` section). In fact the name is a mnemonic name from `dtk quick draw molecule`.

The visualization of the molecule is carried out using the stable and powerful interactive terminals of gnuplot. Usually the standard terminal under Unix-like and MacOSX systems is `x11` (chosen by default by `dtkqdmol`), and `windows` under Windows (also default when `dtkqdmol` is compiled with cygwin). However, one can especify a custom terminal by using the option `-t term`, where `term` is the desired terminal. We use quite regularly the terminals `x11` and `qt`; please ensure that `term` is working properly in your system and that is recognized by gnuplot.

The following command line should create a `gnp` file and immediately call gnuplot, which in turn opens an interactive window with the cyclopropane molecule. A snapshot of the opened window is shown in Fig. 4.8

```
$dtkqdmol ciclopropano_pbe6311.wfx -t qt -r
```

The option `-t qt` only works when the Qt terminal is installed in your system and propperly linked to gnuplot.

To immediately call gnuplot, use the option `-r`. If this switch is not activated, the program will create the `gnp` file, which can be run later with gnuplot.

To close the interactive window, select it (if it is not active), and press any key.

`dtkfindcp` help menu.

```
Usage:

dtkqdmol wf?name [option [value(s)]] ... [option [value(s)]]

Where wf?name is the input wfx(wfn) name, and options can be:

  -o outname   Set the output file name.
                  (If not given the program will create one out of
                  the input name; if given, the gnp file will
                  use this name as well --but different extension--).
  -r           Run the generated gnuplot file.
  -t term      Set the terminal for gnuplot (x11,qt,wxt,...)
                  (Default terminal: x11).
  -V           Displays the version of this program.
  -h           Display the help menu.
  --help          Same as -h
  --version       Same as -V
*******************************************************************************
  Note that the following programs must be properly installed in your system:
                                    gnuplot
*******************************************************************************
```

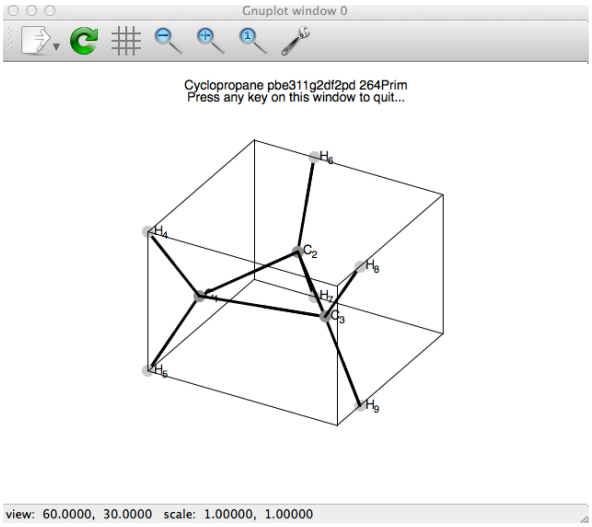


Figure 4.8: Snapshot of the interactive terminal qt under MacOSX.

Chapter 5

Scripts

There are several tasks which are highly repetitive during the processing of data. DensToolKit provides some scripts that perform very simple, but multicommand, tasks. For instance, it is often necessary to correct the bounding box of `eps` figures (a task done by `epstool`) and then produce a `pdf` file (which is done by `epstopdf`). Another often task is to render a `png` image from a `pov` file, especially when adjusting the view by setting the custom options (see §4.7). The latter task can be easily executed with the script `dtkpov2png`. Below is the sintax and some options that can be used.

For the scripts, please note that the options must be written *before* the name of the source file.

5.1 dtkeps2pdf

This script has only one option for setting the output name.

`dtkfindcp` help menu.

```
usage: dtkeps2pdf [option(s) [argument(s)]] [inputname.eps]

This script takes the file inputname.eps, calls epstool to correct the bounding
box of the eps, and finally calls epstopdf. The options can be:

   -o outname.pdf   Set the final name for the pdf to be "outname.pdf"
   -h               Display the help menu.

Notice that epstool and epstopdf must be installed in your system. Please, visit
```

```
http://pages.cs.wisc.edu/~ghost/gsview/epstool.htm
http://www.ctan.org/pkg/epstopdf

for more information about this excelent tools.
```

5.2 dtkpov2png

This script has two options, one for setting the output name, and one another to set the width of the final png image. The height of the image will be calculated from the width, always using a ration 4:3. Hence, it is recomended that the width is a multiple of 12.

`dtkfindcp` help menu.

```
usage: dtkpov2png [option(s) [argument(s)]] [inputname.pov]

This script takes the file inputname.pov, calls povray to create a png, then it
calls gm convert to trim the png file. The options can be:

   -o outname.png   Set the final name for the png to be "outname.png"
   -w width         Set the width of the image to be "width"
                      (Default value: 1200. Since the image will be trimmed,
                       the actual width will be smaller than "width".)
   -h               Display the help menu.

Notice that povray and graphics magic must be installed in your system.
Please, visit

http://www.povray.org
http://www.graphicsmagick.org

for more information about this fabulous programs.
```

Tip: Use a small value of width when testing the view, and use the final resolution once you have chosen the final visualization options.

Bibliography

[1] A. D. Becke and K. E. Edgecombe, "A simple measure of electron localization in atomic and molecular systems," *Jour. Chem. Phys.*, vol. 92, pp. 5397–5403, 1990.

[2] P. Kaijser and V. H. S. Jr., "Evaluation of momentum distributions and compton profiles for atomic and molecular systems," vol. 10 of *Advances in Quantum Chemistry*, pp. 37 – 76, Academic Press, 1977.

[3] L. E. McMurchie and E. R. Davidson, "One- and two-electron integrals over cartesian gaussian functions," *Journal of Computational Physics*, vol. 26, no. 2, pp. 218 – 231, 1978.

[4] A. Banerjee, N. Adams, J. Simons, and R. Shepard, "Search for stationary points on surfaces," *The Journal of Physical Chemistry*, vol. 89, no. 1, pp. 52–57, 1985.

[5] P. Popelier, "A robust algorithm to locate automatically all types of critical points in the charge density and its laplacian," *Chemical Physics Letters*, vol. 228, no. 1–3, pp. 160 – 164, 1994.

[6] R. F. W. Bader, *Atoms in Molecules: A Quantum Theory*. Clarendon Press, 1990.

[7] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Ö. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J. Fox, "Gaussian 09 Revision D.01." Gaussian Inc. Wallingford CT 2009.

[8] M.W.Schmidt, K.K.Baldridge, J.A.Boatz, S.T.Elbert, M.S.Gordon, J.J.Jensen, S.Koseki, N.Matsunaga, K.A.Nguyen, S.Su, T.L.Windus, M.Dupuis, and J.A.Montgomery *J. Comput. Chem.*, vol. 14, p. 1347, 1993.

[9] M. Valiev, E. Bylaska, N. Govind, K. Kowalski, T. Straatsma, H. V. Dam, D. Wang, J. Nieplocha, E. Apra, T. Windus, and W. de Jong, "Nwchem: A comprehensive and scalable open-source solution for large scale molecular simulations," *Computer Physics Communications*, vol. 181, no. 9, pp. 1477 – 1489, 2010.

[10] T. A. Keith, "wfx format specification." http://aim.tkgristmill.com/wfxformat.html, 2010.