DGrid 4.4

— User's Guide —

Miroslav Kohout

January 9, 2009

Preface

The format change in previous version was cursed with some incompatibilities. I hope that the version 4.4 is rather a cure than a curse (however, Murphy's law rules the universe).

At first sight the most obvious change is that the Basin part of the program package was included into the DGrid. This step is a consequence of the concept of mutual calls of routines of both parts of the package. However, separate input files for the grid and basin calculations, respectively, remain. Additionally, DGrid now includes all the utilities as well.

I have changed the procedure for the refinement of the property grids. The input parameter is now the desired precision of the integration. The refinement data are saved in format different from the one in the 4.3 version! Like in the previous version, high integration precision demands large number of additional points to be computed. This can extremely increase the time consumption, especially for molecules with heavier atoms. However, with this procedure it is now possible to evaluate for instance the source function of C. Gatti. Have a look in the 'Utilities' section where the evaluation of the source function is described in detail (as well as the refinement procedure).

There is a new possibility to define the grid region in DGrid 4.3. The corresponding command in the *control* file is '**mesh** *meshsize*'. Then a box with desired mesh size centered around the molecule is created by an internal DGrid routine.

The manual is mainly a description of the command syntax and the process 'flow' and dependencies. Although I have included some examples and few comments it cannot be (and was not the intention) to serve as a course for bonding analysis. The user should know what he is doing and how far he can trust the results. The program works with discrete property grids. Thus, for instance, the zero-flux surfaces are actually not given, but just the basin volumes formed from small cubes. The integration within the basins is done numerically. All this is affected by the mesh size of the property grid.

The info calls were extended. All the possible parameters can be seen with the command:

```
dgrid -h
```

which prints the following information:

```
-------------------------------------------------------------------------
dgrid -p                          print property keywords
dgrid -s                          print spin keywords
dgrid -t                          print type keywords
dgrid -u                          utilities
dgrid -v                          print DGrid version
-------------------------------------------------------------------------
```

Contents

List of Tables

List of Figures

Chapter 1

Property grids

1.1 Disclaimer

The program DGrid is distributed 'as is' without warranties of any kind. The author assume no responsibility of any kind from the use of DGrid.

Use the following citation for the program DGrid:

M. Kohout, DGrid, version 4.4, Radebeul, 2008.

1.2 Installation

The program DGrid is written in C++, i.e., an appropriate compiler must be available. The installation on a Unix system and on a Windows PC (using MS Visual Studio) is described.

1.2.1 Linux

Unpack the distribution tar file dgrid-4.4.tar.gz using the commands:

`gunzip` dgrid-4.4.tar.gz

`tar` xf dgbas-4.4.tar

This creates new directory *dgrid-4.4* (the file *dgrid.sln* is needed only for Windows installation). Change to the directory *dgrid-4.4*. Besides the directory *doc* with the documentation and examples there should be the directory *src* with all the source code files

and a makefile (the file *dgrid-4.4.vcproj* is needed only for Windows installation). View the makefile - you may have to change the name of the compiler (for instance xlC on the IBM-AIX) and the compiler options. DGrid wil be created with the command:

```
make
```

The executable `dgrid-4.4` will be written to the directory $HOME/bin (this can be changed in the makefile). For convenience, link this executable with the file dgrid:

```
ln -s dgrid-4.4 dgrid
```

Run the DGrid calculation with the command:

`dgrid` *controlfilename*

The *control* file name is given as a parameter (i.e., without '<'). The output goes to the console, unless the *output* file name is given in the *control* file, cf. the section 'Control file', page 6).

1.2.2 Windows

In this section the compilation of DGrid with Visual Studio 8 is described. Change to the directory where you wish to save the Visual Studio Project, for instance *C:\Mydir*. Create an empty directory named *dgrid* in which the distribution tar file dgrid-4.4.tar.gz has to be unpacked with an appropriate Windows tool (like Wintar or 7-zip). Change into the directory *dgrid*. You should find there the project file *dgrid.sln* and the directory *dgrid-4.4*. The directory *dgrid-4.4* contains, besides the directory *doc* with the documentation and examples, the directory *src* with all the source code files and the file *dgrid-4.4.vcproj*.

Start the Visual Studio 8 and open the project *dgrid* using the project file *dgrid.sln*. Build the project *dgrid*. This compiles all the files and creates the executable *dgrid-4.4.exe* which can be finally found in the directory *C:\Mydir\dgrid\Release*.

Run the DGrid calculation with the command:

`C:\Mydir\dgrid\Release\dgrid-4.4.exe` *controlfilename*

The output goes to the console, unless the *output* file name is given in the *control* file, cf. the section 'Control file', page 6).

1.3 Introduction

DGrid is a program for the generation of property values on an equidistant grid (or possibly at a single point only) and the analysis of those gridded fields, inclusive the basin search. It does **not** perform visualization of any kind of the computed value fields. For this a separate visualization tool (not included in the package) is required. The value fields from a DGrid calculation are written in a special format, which needs to be recognized by the visualization tool of your choice. There must be an interface provided by you (for instance a program reading the data in DGrid format and writing a file in the format suitable for the visualization tool). Some visualization tools understand the Cube format [1], supported by DGrid. For the visualization program Amira (respectively Avizo) there is a module for the DGrid data available (not included in the package, but free of charge).

For the calculation of property values two input files are essential:

1. *basis* file with the information about the basis set and the molecular orbital representation (for CI calculation additionally also the density matrices) for the evaluated atom or molecule. The atomic or molecular data files supplied by one of the quantum chemical packages GAUSSIAN [2], MOLPRO [3], MOLCAS [4], TURBOMOLE, and ADF [5], respectively, need to be converted into the DGrid *basis* file format. This conversion is described in the next section.
2. *control* file (written by the user) with keywords controlling the property value calculation.

Both *basis* and *control* file must be written in a special format. The informations are read in by the parsing routine `rdcard` that obeys few simple rules:

- Empty lines and lines starting with a colon are skipped.
- Strings and numbers terminated by a colon are skipped.
- Strings and numbers must be separated by one or more blanks.
- Two colons at the beginning mark the line as a text input – the line is read in without parsing (e.g., as a title).
- String followed by '=' is a variable. The string or number following the equal sign is assigned to this variable.

1.4 Basis file

Each of the quantum chemical packages GAUSSIAN, MOLPRO, MOLCAS, TURBOMOLE, and ADF provide an option to write the information about (among others) the molecular geometry, basis set, i.e., the atomic orbitals (AO), and the resulting molecular

orbitals (MO) to a separate output file. Those output files are written in different formats. Additionally, ADF is based on Slater-type orbitals, whereas the other programs use Gauss-type orbitals.

GAUSSIAN calculations supply the binary check-point file *name*.chk that firstly need to be converted into the formatted check-point file *name*.fchk using the GAUSSIAN utility `formchk`. Similarly, the density functional program ADF writes the required information to a binary file TAPE21, that needs to be converted into a formatted file using the ADF utility `dmpkf`. For MOLPRO, MOLCAS, and TURBOMOLE the necessary informations need to be written to a file in *molden* format. In case of TURBOMOLE include in the *molden* file as the second line the string:

[TURBOMOLE]

otherwise the normalization of d and f functions will be wrong.

DGrid reads the necessary information from the *basis* file written in a special format. Thus, the quantum chemical packages output files have to be converted into this format by the conversion routines included in the DGrid package:

`dgrid` *formatted-qm-file*

The resulting DGrid *basis* file includes all orbitals from the quantum chemical calculation (occupied as well as the virtual ones). Of course, in the DGrid *basis* file the unoccupied orbitals have occupations equal zero. To include these orbitals into the property calculation change the occupations manually in the DGrid *basis* file (or in the *control* file using the keyword **occupation**).

Find in the *examples* directory of the DGrid package the formatted GAUSSIAN03 checkpoint file C3H6_HF_6-31G.fchk. Using the command:

`dgrid` C3H6_HF_6-31G.fchk

yields the DGrid *basis* file file C3H6_HF_6-31G.g03 with 39 orbitals (6-31G basis set). Of course, only 12 orbitals are (doubly) occupied.

The first line of the *basis* file starts with the keyword 'BASISFILE'. Additionally, the version of DGrid creating the data is included. The *basis* file contains the information about the package performing the quantum chemical calculation as well as the date, time and the title of the calculation, followed by the coordinates of the involved atoms and the assignment of the basis sets to the atoms. After the header 'MO DATA' the descriptor and the atomic orbital expansion for each symmetry is given (sym: NONE means orbital output without symmetry descriptors). In the AO expansion the number and the descriptor point to the atom and the corresponding AO, respectively, in the basis set list. For example '1 pz_a' is the p_z AO on the atom number 1 (carbon), with the exponents and coefficients of the second p AO on this atom (the first one is without the character appended). The AO expansion descriptors are followed by the energy and the MO coefficients for each orbital in the symmetry. The *basis* file C3H6_HF_6-31G.g98 from the above example reads:

```
BASISFILE   created by DGrid version 4.4   28-11-2008

program: GAUSSIAN03        date: Fri Nov 28 16:57:10 2008

:title
::C3H6 HF 6-31G

:atom  No.         x          y          z          charge
:-------------------------------------------------------
  C     1:       0.0000     0.0000     1.6428         6.00
  ...
  H     9:      -1.6980    -2.3818    -1.3751         1.00
:-------------------------------------------------------

:                        +-------------------------------+
                         :   calculation:   RESTRICTED   :
:                        +-------------------------------+

Energy=   -117.00804 Hartree                        Electrons=  12 alpha  +   12 beta

:                           +--------------------------+
                            :  basis:  CARTESIAN  GTO  :
:                           +--------------------------+

: atom  No.       type            exponents and coefficients
:-------------------------------------------------------------------------------------
   C     1 -   3    s    exp:     3.04752488e+03    4.57369518e+02    1.03948685e+02
                         coe:     1.83473713e-03    1.40373228e-02    6.88426223e-02
         ...
   H     4 -   9    s    exp:     1.61277759e-01
:-------------------------------------------------------------------------------------

:                                  +------------+
:                                  |  MO  DATA  |
:                                  +------------+

sym: NONE               1  s            ...       1  py           1  pz
                           ...          ...          ...             ...

:orb       energy     coefficients
:-------------------------------------------------------------------------------------
   1     -11.22336    0.5748808770      ...     0.0000000000   -0.0002879503
                         ...            ...            ...       ...
...
  39       1.53997   -0.1018194710      ...     0.0000000000    0.1187452130
                         ...            ...            ...       ...
:-------------------------------------------------------------------------------------
:end of MO data
```

```
:                        +------------+
                         | OCCUPATION |
:                        +------------+

:-------------------------------------------------------------
:  #  symmetry         orb          ALPHA            BETA
:-------------------------------------------------------------
   1  NONE              1      1.00000000000    1.00000000000
      ...
  39  NONE             39      0.00000000000    0.00000000000
:-------------------------------------------------------------
```

Notice that the molecular orbital number 39 has positive orbital energy (like all other virtual orbitals in the *basis* file. The MO expansion coefficients are followed by the occupation section. Only the first 12 orbitals are occupied. For an actual DGrid calculation the *basis* file can be renamed to any name. The name of the particular *basis* file is known by the DGrid program through the assignment '**basis**=*basisfilename*' in the *control* file described in next section.

1.5 Control file

For the *control* file any name can be chosen. The *control* file supplies all the information concerning the calculation. The data must comply with the rules described in the introduction. Following is the example *rho.inp* for the *control* file (see the *examples* directory):

```
:TITLE
:-------------------------------------------------------------------|
::C3H6   HF    6-31G
:-------------------------------------------------------------------|

:KEYWORDS
:-------------------------------------------------------------------
 basis=C3H6_HF_6-31G.g03

 output=C3H6_run

:CHOOSE THE DESIRED PROPERTIES
:----------------------------------------
 compute=rho
 compute=rho      laplacian
:----------------------------------------

 mesh=0.1   3.0

END
```

The first readable information in the *control* file must be the title (remember that all empty lines as well as lines beginning with a single colon will be skipped). The title line starts with two colons, i.e., the line will not be parsed. The two colons are obligatory, whereas the title text can be omitted (i.e., empty title). If some title text is found in this line, it will be written into the header of each *property* file.

The keyword **basis** is followed by the name of the *basis* file (described in the previous section). This can also be a UNIX path to this file.

The keyword **output** is followed by the generic name of the *output* file. To this name the extension '.dg' will be appended (for the above example the output will be written to the file *C3H6_run.dg*). Using the assignment '**output=.**' the name of the *output* file would be generated from the *basis* file name (i.e., the file name *C3H6_HF_6-31G.g03.dg* would be generated from the *basis* file *C3H6_HF_6-31G.g03.g98*). Omitting the **output** keyword will send the output to the console. In the *output* file some setup information and the calculation progress is shown. For a single point calculation (see the keyword '**point**=' in section 1.7) all property data will be printed to the given output file.

At least one **compute** assignment, choosing the property to be computed, is obligatory for a grid calculation. With the keyword **compute** the desired property to be computed is chosen. Each property must have its own '**compute**=property' assignment. In the above example the electron density and its Laplacian will be computed. Let as have a closer look on the **compute** assignments. The general form is:

compute=property <type> <spin> <pair>

The descriptors in the brackets can be omitted. Depending on the combination of the descriptors the *result* file name will be affected. As already shown above for the density Laplacian, in the *result* file name the string 'lap_' will precede the property name. With the descriptors <spin> and <pair> additional information will be appended to the *result* file name. In the following the possible descriptors are given together with the strings that will be appended to the *result* file name (exemplary for the property *prop* and the generic name *filename* in coordinate space, i.e., '_r' appended).

The descriptor <type> can be one of (case independent):

- gradient-mag ↦ *filename*.grad-mag_*prop_r*
- gradient-vec ↦ *filename*.grad_*prop_r*
- ln-derivative ↦ *filename*.ln_deriv_*prop_r*
- hessian ↦ *filename*.hess_*prop_r*
- laplacian ↦ *filename*.lap_*prop_r*
- relative-laplacian ↦ *filename*.rel-lap_*prop_r*

If <type> is omitted, then the property value will be computed (and the *result* file named *filename.prop_r*). The <type> descriptors can listed with the command '`dgrid -t`'.

The descriptor <spin> can be one of (case independent):

- alpha ↦ *filename.prop_r_*a
- beta ↦ *filename.prop_r_*b
- both ↦ *filename.prop_r*
- singlet ↦ *filename.prop_r_*s
- triplet ↦ *filename.prop_r_*t

If <spin> is omitted, then total spin will be used (i.e., default is `both`).

The descriptor <pair> can be one of (case independent):

- alpha-alpha ↦ *filename.prop_r_*aa
- beta-beta ↦ *filename.prop_r_*bb
- alpha-beta ↦ *filename.prop_r_*ab
- beta-alpha ↦ *filename.prop_r_*ba
- singlet-pair ↦ *filename.prop_r_*sg
- triplet-pair ↦ *filename.prop_r_*tr
- spinless ↦ *filename.prop_r*

If <pair> is omitted, then the default `spinless` will be used. The <spin> and <pair> descriptors can listed with the command '`dgrid -s`'.

The *property* file names are generated from the generic file name (which is either the *basis* file name or the file name assigned by the keyword **result**, see below). For each property a unique file extension, given in Table 1.1, will be appended to the generic file name. Additionally, for properties computed in coordinate space the descriptor '*_r*' will be appended (for calculations performed in momentum space – keyword '**space**=momentum' – the descriptor '*_p*' would be appended). In the above *control* file the properties are given by the assignments '**compute**=rho' and '**compute**=rho laplacian', i.e., the total electron density (summing up both spins) and the total density Laplacian is chosen. The two *property* files will be named *C3H6_HF_6-31G.g03.rho_r* and *C3H6_HF_6-31G.g03.lap_rho_r*, respectively (because of the assignment **basis**=*C3H6_HF_6-31G.g03*).

As already mentioned, the generic file name can be changed using the keyword **result**. For instance, with the assignment '**result**=*C3H6*' the two *property* files would be named *C3H6.rho_r* and *C3H6.lap_rho_r*, respectively.

In any case, files written by DGrid will not overwrite already existing files. If a file exists, DGrid will create new file name by appending '_1' to it (and successively higher numbers). The *property* files will be written in the DGrid format (special header and 5 values per line). This can be changed with keyword **format** (cf. section 1.7).

Per default all properties are computed in the direct (coordinate) space. With the assignment **space**=momentum it is possible to perform the calculation of the desired properties in the momentum space. At the time the transformation into the momentum space is done only for Gauss-type orbitals.

The assignment '**mesh**=0.1 3.0' defines the grid region. The keyword '**mesh**=' is followed by the mesh-size, i.e., distance between neighboring grid points (the units from the *basis* file are used which is usually set to bohr). Additionally, a border around the initial box can be given. In the example a grid with the distance between neighboring points of 0.1 bohr and a border of 3.0 bohr will be created around the C_3H_6 molecule. The program works as follows (cf. Fig. 1.1 on page 21). From the atomic positions the average coordinate is computed. The 'initial box' (the smallest box around the molecule enclosing all atoms) is created. The box sides are parallel to principal axes of the molecular rotational ellipsoid. The 'initial box' is enlarged in all three directions by the border of 3.0 bohr (the grid region dimensions being multiple of the mesh-size of 0.1 bohr). This results in a grid of $95 \times 109 \times 103$ points (with the dimensions of 9.4 bohr $\times$ 10.8 bohr $\times$ 10.2 bohr, cf. the lower diagram in Fig. 1.1).

Next example for the *control* file showing another possibility to define the grid region is given with the file *elid.inp* (see the *examples* directory):

```
:TITLE
:-------------------------------------------------------------------|
::C3H6   HF    6-31G
:-------------------------------------------------------------------|

:KEYWORDS
:--------------------------------------------------------------------
 basis=C3H6_HF_6-31G.g03

 output=.


:CHOOSE THE DESIRED PROPERTIES
:-----------------------------------------
 compute=ELI-D    triplet-pair
:-----------------------------------------
```

```
 GRID_DEFINITION:   vectors
:----------------------------
:              X     Y     Z
:----------------------------
 origin:    -5.0  -5.0  -8.0
                               INTERVALS:
:--------------------------------------
 i-vector:  10.0   0.0   0.0     100
 j-vector:   0.0  10.0   0.0     100
 k-vector:   0.0   0.0  16.0     160

END
```

Here, the grid region is explicitly given by three vectors. The definition starts with the keyword **vectors**. It is followed by a line with three floating point numbers for the coordinates of the region origin. Next three lines contain the data for three vectors spanning the chosen volume (i.e., the shift with respect to the region origin, NOT the endpoints of the vectors) as well as the number of intervals along each vector. In the example an orthogonal grid of 101 × 101 × 161 points in a box of 10 bohr × 10 bohr × 16 bohr (i.e., 0.1 bohr mesh) with lower left vertex at the coordinates [-5.0, -5.0, -8.0] will be computed.

The assignment '**output=.**' in the above example directs the program to write the output data into the file *C3H6_HF_6-31G.g03.dg* as deduced from the *basis* file name. The '**compute**=ELI-D triplet-pair' assignment yields a grid written to the *result* file named *C3H6_test.elid_r_t_tr*, showing that in this case ELI-D is based on the charge of triplet-coupled electrons ('*_t*' part) that is needed to form a fixed fraction of triplet pair ('*_tr*' part) [7]. Additionally, it can be seen that in this case the <spin> descriptor is set automatically to 'triplet' (similarly, for the assignment '**compute**=ELI-D triplet' the <pair> descriptor would be set to 'triplet-pair').

The properties are computed at every grid point from all occupied orbitals. The occupations are given in the *basis* file for each orbital of particular symmetry. To compute a property for specified orbitals (for instance the electron density based on certain orbitals) the occupations must be specified after the keyword **occupation**. The data must be written in 1 line for each chosen orbital. First the name of the symmetry must be given, followed by the number of the desired orbital in this symmetry (the number is accessible from the *basis* file) together with the α-spin and β-spin occupations. In the *control* file *rho_orb.inp*, shown below, the orbitals number 11 and 12 of the symmetry 'NONE' (GAUSSIAN output without symmetry), both double occupied, are selected for the calculation. To compute the amplitude for 1 single orbital, use '**compute**=phi'. In this case only 1 orbital is allowed in the occupation section.

```
:TITLE
:-------------------------------------------------------------------|
::C3H6   HF    6-31G
:-------------------------------------------------------------------|

:KEYWORDS
:-------------------------------------------------------------------
 basis=C3H6_HF_6-31G.g03

 output=.

:CHOOSE THE DESIRED PROPERTIES
:----------------------------------------
 compute=rho
:----------------------------------------

 mesh=0.1  3.0


 USERS_OCCUPATION_INPUT:  occupation
:-----------------------------------

: SYM      #         ALPHA  BETA
:------------------------------
 NONE     11         1.0    1.0
 NONE     12         1.0    1.0

END
```

Table 1.1 summarizes the properties that can be calculated (the conditional properties are given in Table 1.2 on page 13). For almost all properties the derivatives can be computed using the <type> descriptors for the it compute keyword, cf. page 7. ELI-D in Table 1.1 is written as $\Upsilon_D = \rho\tilde{V}_D$, i.e., the electron density ρ multiplied by the pair volume function $\tilde{V}_D$. As described in Ref. [18] ELI-D can be exactly decomposed into orbital contributions, the same way as the electron density is given by the sum over orbital contributions.

The calculation of the ELI-D contributions is performed with the property keyword 'pELI-D', for instance with '**compute**=pELI-D alpha', cf. the *control* file *pelid.inp* in the *example* directory. In this case the Υ_D^α, i.e., ELI-D for the α-spin pairs is considered. The contributions of the two highest occupied orbitals to Υ_D^α are chosen with the keyword **occupation**. The pair-volume function $\tilde{V}_D$ for pELI-D is of course computed from the total pair density. The occupations of the chosen orbitals are highlighted in the *output* file. Observe, that although the occupations are given for both spins, only the α-spin part is taken into account. Summing up all the contributions of all orbitals recovers the total ELI-D. Beware! Using the assignment '**compute**=eli-d alpha' together with the keyword **occupation**, instead of '**compute**=peli-d alpha', yields another quantity, because in this case also the pair-volume function $\tilde{V}_D$ is calculated from the chosen orbitals. This quantity does NOT add up to total ELI-D.

Table 1.1: **Available properties**

Keyword	Property	Description	Ref.	File ext.
compute=ELF	$[1+\chi^2]^{-1}$	ELF: 'spin-polarized'	[8, 9, 10]	elf
compute=ELF-cs	$[1+\chi_{cs}^2]^{-1}$	ELF: 'closed-shel'	[8, 9, 10]	elf_cs
compute=ELF-Kirshnitz	$[1+\chi_{kirsh}^2]^{-1}$	ELF: Tsirelson	[11]	elf_kirsh
compute=ELI-D	$\rho\,\tilde{V_D}$	ELI-D	[12, 13]	elid
compute=ELI-q	$\rho_2^{(s)}\,\tilde{V_q}^2$	ELI-q	[7]	eliq
compute=ELIA	$\rho_2^{\alpha\beta}/(\rho_\alpha\rho_\beta)$	ELIA	[15]	elia
compute=hole-curvature	$\nabla^2\rho_2^{\alpha\alpha}$	Fermi-hole curvature		hole_curv
compute=LOL	$\frac{\tau^{LSDA}}{\tau+\tau^{LSDA}}$	localized orbital locator	[16]	lol
compute=LS	$-\frac{1}{4\pi}\frac{\nabla^2\rho(r)}{\lvert r-r'\rvert}$	local source	[20, 21]	ls
compute=OEP	$\frac{\nabla^2\sqrt{\rho}}{2\sqrt{\rho}}$	one-electron potential	[17]	oep
compute=pair-volume-function	$\tilde{V_D}$	pair volume function	[18]	pair_vol
compute=pELI-D	$\rho_{orb}\,\tilde{V_D}$	orbital resolved ELI-D	[18]	p_elid
compute=phi	ϕ_i	orbital amplitude		phi
compute=charge-volume-function	$\tilde{V_q}$	charge volume function	[7]	q_vol
compute=rho	ρ	electron density		rho
compute=on-top-density	$\rho_2(r,r)$	on-top density		rho_ontop
compute=rho-spin	ρ_s	spin density		rho_spn
compute=tau	τ	kin. energy density		tau
compute=tp	$\tau - t_w$	Pauli kin. energy dens.		tp
compute=tw	$\frac{1}{8}\frac{(\nabla\rho)^2}{\rho}$	Weizsäcker term		tw

$\rho_s = \rho_\alpha - \rho_\beta$

$G = \sum_{i<j}^{\sigma}\sum_{k<l}^{\sigma} P_{ij,kl}\,[\phi_i\nabla\phi_j - \phi_j\nabla\phi_i].[\phi_k^*\nabla\phi_l^* - \phi_l^*\nabla\phi_k^*]$ – Fermi hole curvature

$\tilde{V_D} = [12/G]^{3/8}$

$\tilde{V_q} = 1/\rho^{(s)}$ ($\rho^{(s)}$ density of singlet coupled electrons)

$c_F = \frac{3}{10}(3\pi^2)^{2/3}$ – Fermi constant

$\tau = \frac{1}{2}\sum_i |\nabla\phi_i|^2$ – kinetic energy density of noninteracting system

$\chi = \left[\tau - \frac{1}{8}\frac{(\nabla\rho_\alpha)^2}{\rho_\alpha} - \frac{1}{8}\frac{(\nabla\rho_\beta)^2}{\rho_\beta}\right] / \left[2^{2/3}c_F\,(\rho_\alpha^{5/3}+\rho_\beta^{5/3})\right]$

$\chi_{cs} = \left[\tau - \frac{1}{8}\frac{(\nabla\rho)^2}{\rho}\right] / \left[c_F\,\rho^{5/3}\right]$

$\chi_{kirsh} = \left[c_F\,\rho^{5/3} - \frac{1}{9}\frac{(\nabla\rho)^2}{\rho} + \frac{1}{6}\nabla^2\rho\right] / \left[c_F\,\rho^{5/3}\right]$

$\tau^{LSDA} = 2^{2/3}c_F\,\rho^{5/3}$ – originally defined for single spin channel, i.e., ρ_σ [16]

Table 1.2: **Conditional properties**

Keyword	Property	Description	File ext.
compute=cond-pair-density	$P_{cond}(1,2)$	conditional pair density	pd_cond
compute=hole-density	$\rho^{\sigma}_{ex}(1)$	hole density	hole_density
compute=pair-density	$\rho_2(1,2)$	pair density	pd

$\rho_2^{\sigma\sigma}(1,2) = \frac{1}{2}\sum_{i<j}^{\sigma}\sum_{k<l}^{\sigma} P_{ij,kl}\ |\phi_i(1)\phi_j(2)|\ |\phi_k^*(1)\phi_l^*(2)|$
$P_{cond}^{\sigma\sigma}(1,2) = \rho_2^{\sigma\sigma}(1,2)/\rho_{\sigma}(2)$
$\rho_2^{\alpha\beta}(1,2) = \frac{1}{2}\sum_{i,j}^{\alpha}\sum_{k,l}^{\beta} P_{ij,kl}\ |\phi_i(1)\phi_j(2)|\ |\phi_k^*(1)\phi_l^*(2)|$
$P_{cond}^{\sigma\sigma'}(1,2) = \rho_2^{\sigma\sigma'}(1,2)/\rho_{\sigma'}(2)$
$\rho_2(1,2) = \rho_2^{\alpha\alpha}(1,2) + \rho_2^{\beta\beta}(1,2) + \rho_2^{\alpha\beta}(1,2) + \rho_2^{\beta\alpha}(1,2) = \rho_2^{(s)}(1,2) + \rho_2^{(t)}(1,2)$
$P_{cond}(1,2) = \rho_2(1,2)/\rho(2)$
$\rho^{\sigma}_{ex}(1) = 2P^{\sigma\sigma}_{cond}(1,2) - \rho_{\sigma}(1)$

With the assignment '**point**= *x y z*' the calculation is performed at the cartesian position [*x y z*] only. The desired property value, its gradient, Hessian, eigenvectors and curvatures are printed into the *output* file, resp. on the console. In this case the grid definition given by the **vectors**, respectively **mesh** keyword are not used, cf. the *control* file *cond.inp* used below (with no **output** assignment, i.e., results printed to console):

```
:-------------------------------------------------------------------|
::C3H6   HF    6-31G
:-------------------------------------------------------------------|

:KEYWORDS
:-------------------------------------------------------------------
 basis=C3H6_HF_6-31G.g03

:CHOOSE THE DESIRED PROPERTIES
:----------------------------------------
 compute=rho                 alpha
 compute=cond-pair-density   alpha-alpha
:----------------------------------------

 ref_point= 0.0  0.0  0.5
 point    = 0.0  0.0  0.0

 mesh=0.1  3.0
END
```

With the assignment '**ref_point**= *x y z*' the position of the reference electron is fixed at cartesian coordinates [*x y z*]. This is necessary for the calculation of the electron pair density (6 dimensional property) and some related properties (for instance the conditional properties), cf. Table 1.2. Then, fixing the position of 1 electron yields a 3 dimensional property. Running the *control* file *cond.input* gives the electron density (together with the gradient and Hessian) at the position [0.0, 0.0, 0.0] (for the density the reference point is not taken into account). Observe, that the (practically) zero gradient and the signature of the curvatures mark this position as a ring critical point. Additionally, the conditional $\alpha\alpha$-pair density at [0.0, 0.0, 0.0] (together with the corresponding gradient and Hessian) for the reference electron fixed at the position [0.0, 0.0, 0.5] is computed.

The last information in the *control* file must be the keyword **end**. Data lines following the keyword **end** are not read. All keywords are read case insensitive. The keywords can be written in any order. Also, there is no need to write each keyword in a separate line.

All available keywords are described in detail in section Keywords).

1.6 Property files

Each property is written to separate file distinguished by a descriptor according to the **compute** assignment, see Tables 1.1 and 1.2. Every *property* file has a header followed by the property values. Following is an example of the *property* file header for the orbital density computed from data supplied by the GAUSSIAN03 package:

```
GRIDFILE    created by DGrid version 4.4   28-11-2008

:job was executed on:         Thu Dec  4 10:46:49 2008

basis_from_program:      GAUSSIAN03              basis=C3H6_HF_6-31G.g03

property:   rho[r]                               spin=both       pair=unknown

: property computed from orbitals
:--------------------------------------
:  symmetry         No.    occupation
:--------------------------------------
  NONE             11     1.000   1.000
  NONE             12     1.000   1.000
:--------------------------------------

::C3H6   HF    6-31G

:atom       x          y          z
:-----------------------------------
 C       0.0000     0.0000     1.6428
 ...
```

```
 H      -1.6980    -2.3818    -1.3751
:-----------------------------------

Energy=    -117.00804 Hartree                      Electrons=   12 alpha  +    12 beta

:             lattice vectors
:-------------------------------------
:         a           b           c
:-------------------------------------
x:    1.000000    0.000000    0.000000
y:    0.000000    1.000000    0.000000
z:    0.000000    0.000000    1.000000
:-------------------------------------

:        origin           v_i         v_j         v_k
:-----------------------------------------------------
x:      -4.7000        9.400000    0.000000    0.000000
y:      -5.4000        0.000000   10.800000    0.000000
z:      -4.4000        0.000000    0.000000   10.200000

points:                    95         109         103
:-----------------------------------------------------

: start of data
:------------------------------------------------------------------------------
 9.8558557e-08   1.2104934e-07   1.4784535e-07   1.7957637e-07   2.1692357e-07
...
```

The header starts with the keyword 'GRIDFILE' followed by the information about the DGrid version. Next lines contain the date and time of the calculation, the name of the quantum chemical package that produced the *basis* file and the name of the *basis* file. In certain cases DGrid needs the access to the *basis* file. Thus, do not rename the this file (or change the name in the respective grid files as well). If DGrid is not able to find the *basis* file it will possibly follow another evaluation route (the corresponding information will be given in the *output* file).

The next line contains the name of the calculated property followed by the chosen spin and pair-spin. This information is again necessary for DGrid to perform certain evaluations. Especially if old DGrid files are used, this information is missing. Then some new features of the program will not be used! For instance, in previous version 'ELI' was used instead of 'ELI-D'. The programs do not known the property 'ELI' in the 4.4 version. Thus, e.g., DGrid will not be able to find the exact attractor positions, or to evaluate the curvatures at attractor positions.

If the calculation was performed for selected orbitals, like in the above example, then the chosen orbitals as well as the occupations are written in lines following the property.

The title (from the DGrid *control* file) is followed by the atomic coordinates. Next line shows the energy of the system and the number of electrons. The remaining part of

the header contains the information about the grid region, which are the lattice vectors (meaningful only for solid state calculations) and the description of the computed region with the coordinates of the grid origin together with the three vectors spanning the region and the number of points in each direction. In the case of scalar field this is also the last line of the header (see latter for vector fields).

The values of the computed property are written in the format 14.7e. The grid points run in the x direction first (in contrast to the LMTO format where z runs first).

If the property *prop* was computed with the assignment '**compute**=*prop* gradient-vec' then all 3 vector components will be written to the *result* file. The descriptor '$vec = 3$' indicates that the data are not single valued. Instead, 3 values per point are written in each line, as exemplary shown below:

```
:        origin           v_i          v_j          v_k
:-------------------------------------------------------
x:      -4.7000        9.400000     0.000000     0.000000
y:      -5.4000        0.000000    10.800000     0.000000
z:      -4.4000        0.000000     0.000000    10.200000

points:                   95          109          103
:-------------------------------------------------------

vec=3

: start of data
:-------------------------------------------------------------------------------
:     vx              vy              vz
:-------------------------------------------------------------------------------
 1.1930383e-06   1.1807959e-06   1.1716233e-06
 1.4118954e-06   1.4427352e-06   1.4308966e-06
...
```

1.7 Keywords

In this section all keywords (for exceptions see below) available for the DGrid *control* file are described in alphabetical order. The keywords are read in case insensitive. The DGrid program is now inclusive the Basin part, i.e., the calculation and examination of basins is performed by DGrid. However, there is still a separate *input* file needed for such analysis. Therefore, the keywords associated with basin evaluation are described in section 2.5.

The keywords are given either as variables, e.g., like '**compute**=', or standing alone, like '**end**'. Some of them are followed by 1 or more numbers, e.g., '**mesh**=0.1 3.0'. If float number should be given then one must really input float numbers, otherwise the input routine will throw an error message. Thus, the command '**point**= 0.5 1.0 3' would not be valid, because '3' is an integer number.

Table 1.3: **Keywords and parameters available for the grid calculation**

Keyword	Description	Value
basis=	basis set	basis set filename
compute=	choose property	properties given in Table 1.1
contributions=	orbital contributions	property (rho, eli), x, y, z (float)
cp=	search for critical point	(max, saddle, ring), x, y, z (float)
end	end of input	
format=	result file (grid) format	cube, `dgrid`, grace, lmto
mesh=	define grid by mesh size	step and box border
occupation	choose orbital occupations	list of orbitals
output=	specify output file	output filename
point=	properties at given point	3 cartesian coordinates (float)
reduced_step=	step for trajectory	float number (default 0.04)
ref_point=	reference point	3 cart. coordinates (float)
result=	specify result filename	generic name of the property file
space=	space representation	momentum, `position`
values=	number of values in a row	integer number (default 5)
vectors	define grid by 3 vectors	origin, 3 vectors, and intervals
wf_part=	select wavefunction part	`both`, imaginary, real

Default values are given in `Typewriter` font

Often more keywords can be input in single line. But it is better to put every keyword separately in a line, because sometimes additional data are assumed per default. Some data even need to be written in rigorous sequence. For instance the definition of a grid using the **vectors** keyword or the choice of occupied orbitals. Table 1.3 summarizes all the keywords available for the grid and single point evaluation, respectively.

1.7.1 Basis

The keyword **basis** is followed by the name of the *basis* file (described in section 1.4). This can also be a UNIX path to this file:

basis=/home/test/molecule.adf

If the name of the *result* file is not explicitly given then it will be generated from the

name of the *basis* file. Consequently, in this case the *result* files will be written to the same directory where the *basis* file is located. Computing for the above *basis* file the electron density yields the *result* file named */home/test/molecule.adf.rho_r*. If you wish to avoid this, use the '**result**=*newname*' assignment, yielding the file *newname.rho_r*. The same applies to the *output* file (where the file */home/test/molecule.adf.dg* would be generated).

1.7.2 Compute

The keyword **compute** is followed by the desired property <prop>. All available properties are given in Tables 1.1 and 1.2 on page 12 and 13, respectively.

compute=<prop> <type> <spin> <pair>

The descriptors <type>, <spin>, and <pair>, described in the section 1.5 on page 7, can be written in any order. Each property must have its own **compute** assignment on separate line. Bear in mind that for some properties calculation based on set of chosen orbitals (see keyword **occupation**) may not be reasonable (for instance for one-electron potential or ELF).

1.7.3 Contributions

The keyword **contributions** is followed by the property <prop> together with the desired descriptors <type>, <spin> and <pair>. However, for the property only one of 'rho', 'rho laplacian', or 'ELI-D', respectively, is allowed by now.

contributions=<prop> <type> <spin> <pair> <x> <y> <z>

DGrid computes the property and its orbital contributions at the position given by the cartesian coordinates [*x y z*] (float numbers). Using the *control* file *elid_contrib.inp* yields the *output* file *C3H6_elid_contrib.dg*. Below is the result of the analysis:

```
------------------------------------------------------------------------------
contributions at point           [  0.000000,    1.128050,    0.651290]
------------------------------------------------------------------------------


                    spin :        alpha
                    pair :        alpha-alpha
                                 ----------
               ELI-D[r] :        1.78581
```

```
--------------------
      MO          %
--------------------
   11   11      38.9
    4    4      27.2
   12   12      13.0
    8    8      10.3
    5    5       8.0
    6    6       2.7
--------------------
               100.0
```

The orbital contributions are given at the position of an ELI-D bond attractor (found using the keyword **cp**, see below). The analysis shows that the ELI-D value 1.78581, which is proportional to the population of α-spin electrons needed to form a fixed fraction of an $\alpha\alpha$-spin pair, is a sum of mainly five contributions. The two highest contributions originate from the canonical orbitals $\phi_1 1\phi 11^*$ (38.9%) and $\phi_4\phi_4^*$ (27.2%). The orbital indexes correspond to the numbering in the *basis* file *C3H6_HF_6-31G.g03* set in the *control* file.

If the keyword **contributions** is used, any **compute** assignments will be ignored (actually, in this case the **compute** assignment can be omitted). Also, any grid definitions will be skipped, i.e., only the given position will be evaluated, and no *result* file will be written.

1.7.4 Cp

The keyword **cp** is followed by the descriptor <crit> for the critical point:

cp=<crit> <x> <y> <z>

The descriptor <crit> can be one of (case insensitive):

- min ↦ minimum (3, +3)
- max ↦ maximum (3, -3)
- saddle ↦ saddle point (3, -1)
- ring ↦ ring critical point (3, +1)

The search for the critical point starts from the position given by the cartesian coordinates $[x\ y\ z]$ (float numbers). The property for which the critical point will be searched must

be given in a **compute** assignment. The position of the critical point (if found) is printed to the *output* file, together with the property value at this point, the gradient (which will be close to zero) and the Laplacian. Additionally, the eigenvectors and curvatures are determined from the Hessian. If you see the information 'Using Taylor' then the Hessian needed for the search was not computed analytically.

The search path is saved in the file *search.path.str*. This file is written in the STR format (for more details see section 4.2 in the Appendix). In the *control* file *elid_cp.inp* for the search of an ELI-D attractor the starting point [0 2 1] was chosen. The resulting attractor (C-C bond descriptor) position is [0.000000, 1.128050, 0.651290], see the *output* file *C3H6_elid_cp.dg*. This position was used in the above example for the ELI-D contributions.

1.7.5 End

The keyword **end** is the last keyword in the *control* file that will be parsed. Any information after this keyword will be ignored.

1.7.6 Format

The keyword **format** is followed by the descriptor <fmt> for the format of the *result* file:

format=<fmt>

The format descriptor <fmt> can be one of (case insensitive):

- cube ↦ CUBE [1]
- dgrid ↦ *result* file
- grace ↦ Grace [6]
- lmto ↦ TB-LMTO-ASA [26]

In both the Cube and Grace formats the property values are given by float numbers. Routines which expect grids of integer values (like the basin evaluation) could fail.

1.7.7 Mesh

The computed grid region is defined with the keyword **mesh**. It is followed by the mesh-size, i.e., the distance between neighboring points. Optionally, a border around the molecule can be given.

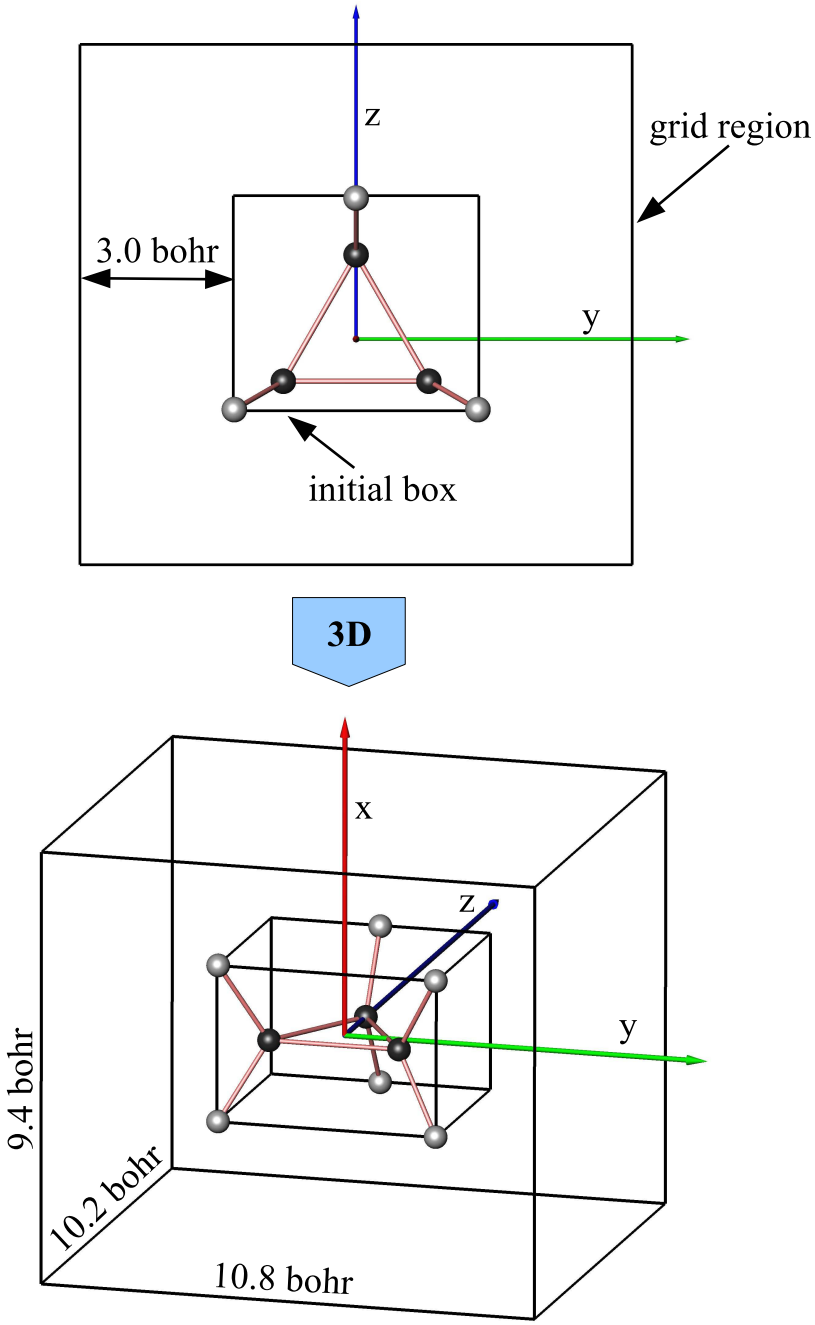

Figure 1.1: Grid region for C_3H_6 (C black, H gray) created by the assignment '**mesh**=0.1 3.0'

mesh=<size> <border>

The mesh-size must be given as a float number. The units for both the mesh-size and the border are identical with the units for the atomic positions given in the *basis* file (usually bohr).

The determination of the grid region can be shown with the example *rho.inp* from the *example* directory (see also page 6) where the assignment The assignment '**mesh**=0.1 3.0' is used. In this example a grid with the distance between neighboring points of 0.1 bohr and a border of 3.0 bohr is created around the C_3H_6 molecule. The routine works as follows (cf. Fig. 1.1 on page 21). From the atomic positions the average coordinate is computed. The 'initial box' (the smallest box around the molecule enclosing all atoms) is created. The box sides are parallel to the principal axes of the molecular rotational ellipsoid. The 'initial box' is enlarged in all three directions by the border of 3.0 bohr (the grid region dimensions being multiple of the mesh-size of 0.1 bohr). This results in a grid of $95 \times 109 \times 103$ points (with the dimensions of 9.4 bohr $\times$ 10.8 bohr $\times$ 10.2 bohr, cf. the lower diagram in Fig. 1.1).

1.7.8 Occupation

The keyword **occupation** marks a section in the *control* file, where orbitals can be chosen for the evaluation. It must be the only keyword in the line:

occupation

The next readable information after this line describes the orbital choice using the following syntax:

<label> <nr> <occa> <occb>

Where <label> is the symmetry descriptor from the *basis* file (cf. page 5, for instance 'NONE' for the Gaussian calculations, see also the example *control* file *rho_orb.inp*). The symmetry label is followed by the orbital number, which must comply with the numbering in the *basis* file (for ADF calculations for each symmetry the orbital numbering starts again with 1). Next two (float) numbers are the orbital occupations for each spin channel. Other possibility is:

<label> all

Here all orbitals with given symmetry label will be chosen (for instance all sigma orbitals). The orbital choices will be read until the data do not comply with the above syntax.

1.7.9 Output

The keyword **output** is followed by the generic name of the *output* file. This can also be a UNIX path to this file.

output=<XY_test>

The *output* file name will be generated by appending the string ''.dg' to the generic name (i.e., 'XY_test.dg' for the above example). If the name of the *output* file is not explicitly given by the **output** assignment then the output will be written to the console. Using the assignment:

output=.

generates the *output* file name from the name of the *basis* file by appending the string '.dg' to it.

1.7.10 Point

The keyword **point** is followed by the position given by the cartesian coordinates [x y z] (float numbers) at which the properties will be evaluated:

point= <x> <y> <z>

The properties must be given by corresponding **compute** assignments. For each property the value at the position [x y z] is printed to the *output* file (respectively console), together with the property gradient and the property Laplacian. Additionally, the eigenvectors and curvatures are determined from the Hessian. If you see the information 'using Taylor' then the data were not computed analytically.

If the keyword **point** is used any grid definitions will be ignored, i.e., only the given position will be evaluated, and no *result* file will be written.

An additional feature is the information about indexes connected with local virial [24]. The feature is invoked with the keyword *virial* following the [x y z] coordinates:

point= <x> <y> <z> `virial`

For all properties given by the **compute** assignments the corresponding data at the [x y z] position, like in the previous case. However, at the end of the output the the following line will be included (data for C_3H_6 at the position [1.0 0.5 0.5]):

```
-------------------------------------------------------------------------
Virial indexes:        L          V          G         |V|/G        H
-------------------------------------------------------------------------

                     0.2073     -0.1304     0.0911     1.4311     -0.0393
-------------------------------------------------------------------------
```

where L is the total electron density Laplacian, G is the positive definite kinetic energy density, $V = \frac{1}{4}\nabla^2\rho - 2\tau$ is the local potential energy density and $H = G + V$ is the energy density. $|V|/G$ is the corresponding local virial ratio at the position $[x\ y\ z]$.

1.7.11 Reduced_step

With the keyword **reduced_step** the search for critical points can be done with reduced maximal step (to avoid jumps between regions with different Hessian form):

reduced_step=<step>

The defaul value for <step> is 0.04 bohr.

1.7.12 Ref_point

The keyword **ref_point** is followed by the position given by the cartesian coordinates [x y z] (float numbers) for the reference electron:

ref_point= x y z

The assignment is needed only for 2-particle properties, see Table 1.2 on page 13. In this case the coordinate of one electron is fixed at the position [x y z], whereas the position of the other electron runs through the grid (as defined by the **vectors** or **mesh** assignments). Additionally, it is used for the reference position when calculating the local source (with '**compute**=LS').

1.7.13 Result

The keyword **result** is followed by the generic name of the *result* file. This can also be a UNIX path to this file.

result=<XY_test>

The *result* file names for each computed property are generated from this generic name

appending the corresponding strings (see Tables 1.1 and 1.2 on page 12 and 13, respectively). If the name of the *result* file is not explicitly given by the **result** assignment then DGrid generates the *result* file name from the name of the *basis* file by appending the strings corresponding to the respective properties.

1.7.14 Space

The keyword **space** is followed by the descriptor <spc> determining in which space representation the properties will be computed:

space=<spc>

The descriptor <spc> can be one of (case insensitive):

- position $\mapsto$ coordinate space $[x\ y\ z]$
- momentum $\mapsto$ momentum space $[p_x\ p_y\ p_z]$

The default is the evaluation in position space representation. In case of momentum space representation the grid definition is using the momentum coordinates. Not all properties have reasonable counterparts in the momentum space. DGrid computes the properties in momentum space using the same formulas as in the coordinate space. The only difference is that it utilizes Fourier-transformed orbitals (momentals). Thus, the charge has similar meaning in momentum space, however not the kinetic energy density, if computed from squared orbital gradients (in momentum space the kinetic energy is proportional to the expectation value of p^2).

1.7.15 Values

The keyword **values** is followed by the number <num> of values per line in the *result* file:

values=<num>

<num> must be given by an integer number.

1.7.16 Vectors

The keyword **vectors** marks a section in the *control* file, where the grid region is defined. It must be the only readable information in the line:

vectors

The next readable information after this line defines the origin of the grid using 3 cartesian coordinates [x y z] (float numbers):

<x> <y> <z>

In the *control* file the strings 'GRID_DEFINITION:', 'origin:', 'i-vector:', etc. – which are not parsed due to the colon at the end of the strings – are used just for clarity:

```
 GRID_DEFINITION:   vectors
:              X     Y     Z
:---------------------------
 origin:     0.0  -4.0   0.0
:                              INTERVALS
:-------------------------------------
 i-vector:   3.0   4.0   0.0     100
 j-vector:  -4.0   3.0   0.0     100
 k-vector:   0.0   0.0   5.0     100
```

The next 3 lines define the vectors spanning the grid mesh. Each line contains 3 float numbers for the vector length along the x, y, z axes, and an integer number for the number of intervals the vector will be divided in. Thus, the above assignment will generate a grid mesh starting at the position [0 -4 0]. The 3 vectors (i,j,k) are 5 bohrs long. The direction of vector k coincides with z axis. The vectors i, j are perpendicular to each other. The endpoint of vector i is at [3 0 0] (this information is not explicitly given in the input, but can easily be deducted from the data – from the origin position [0 -4 0] one moves 3 bohrs in x direction and 4 bohrs in y direction). The grid mesh has points spaced by 0.05 bohr along each vector (i.e., $101 \times 101 \times 101$ points in total). To avoid numerical instabilities the mesh point distance should not exceed 0.1 bohr.

1.7.17 Wf_part

The keyword **wf_part** is followed by the descriptor <img> determining which part of the wavefunction is used for the calculation of the properties:

wf_part=<img>

The descriptor <img> can be one of (case insensitive):

- real ↦ use real part only
- imaginary ↦ use imaginary part only

- both ↦ use both parts

The default value is `both`.

Chapter 2

Basin evaluation

2.1 Introduction

On an equidistant grid of property values regions surrounded by the surfaces of zero-flux of the property gradient (i.e., basins [24]) are determined. Alternatively, localization domains [25], i.e., regions surrounded by an isosurface, resp. separate regions surrounded by two isosurfaces can be determined.

DGrid computes the volumes of the basins, resp. the localization domains. If there is a second grid for another property (with the same grid parameters as the first one) the program integrates this property over each basin. An output file can be created, where each grid point is assigned to its basin. For molecules, if the *basis* file is present, the attractors and saddle points are determined, together with the corresponding curvatures and Hessian eigenvectors. Additionally, the interconnection graph (graph with attractors and saddle points connected by gradient field lines; in case of the electron density called the molecular graph) can be produced (using the STR format, see the Appendix).

The basin evaluation is performed by DGrid using a *control* file with keywords specific for this analysis. The control file must comply with the rules given in the introduction to the DGrid program.

2.2 Control file

For the *control* file any name can be chosen. The *control* file supplies all the information concerning the calculation. The data must comply with the rules described in the introduction to chapter 1, page 3. A simple basin calculation can be performed with very few command. Following is the *control* file *rho_basins.inp* for the calculation of electron density basins for the C_3H_6 molecule (see the *examples* directory):

```
:TITLE
:-------------------------------------------------------------------|
::C3H6 density basins
:-------------------------------------------------------------------|

:KEYWORDS
:--------------------------------------------------------------------
 property =C3H6_HF_6-31G.g03.rho_r
 integrate=C3H6_HF_6-31G.g03.rho_r

 output=.

end
```

The *control* file could be even further simplified by omitting the charge integration invoked by the **integrate** keyword. The *property* grid file *C3H6_HF_6-31G.g03.rho_r* must be computed in advance by a separate DGrid job (cf. the *control* file *rho.inp* from the *example* directory).

The first readable information in the *control* file must be the title (whereby all empty lines as well as lines beginning with a single colon will be skipped). The title line starts with two colons, i.e., the line will not be parsed. The two colons are obligatory, whereas the title text can be omitted (i.e., empty title). If some title text is found in this line, it will be written into the header of the *basin* file, see section 2.3 below.

The keyword **property** is obligatory for the basin calculation. It is followed by the name of the *property* file (i.e., the file with the scalar grid values for the desired property). This can also be a UNIX path to this file. The program determines the basins, i.e., regions enclosed by the surfaces of zero-flux of the property gradient. Alternatively, a *basin* file (see below) with basin data from previous run can be read in instead of the *property* file. Of course, in this case the basins will not be recalculated, thus saving time.

If the keyword **integrate** is given, then the property from the file named after the keyword will be integrated over each basin. In case of electron density use a grid with the grid point distance less than 0.1 bohr (better 0.05 bohr) to get reasonable charges. Alternatively, the file from a DGrid refinement run can be used.

The keyword **output** is followed either by the name of the *output* file or by the '.' character (as in the above example). In this case the *output* file name will be generated from the *property* file name. In both cases the string '.bas' will be appended to the generic file name. Thus, for the above example the output will be written to the file *C3H6_HF_6-31G.g03.rho_r.bas*.

After the basins are determined a grid file will be created with each grid point assigned to its basin by an integer number. The name of the resulting *basin* file will be generated from the *property* file name. Alternatively, the name for the *basin* file can be chosen using the keyword **result** followed the generic file name. In both cases the string '.bsn' will be appended to the generic file name. The *basin* file is written in the DGrid format (**x** coordinate runs first). The format can be changed with the keyword **format**.

All available keywords are summarized in section 2.5.

The attractors, the prerequisite for a basin, are searched on the discrete grid. This often yields much more discrete local maxima than actually given by the attractors of the vector field. There are two possibilities for the DGrid program how to proceed, depending on the presence of the *basis* file (this information is given in the *property* file – bear in mind that the *basis* file must be present at the given location!):

1) Unknown *basis* file

This is usually the case for solid state calculations or if the *basis* file was not found by DGrid (in which case a warning will be printed into the output. Using the keyword **attractors** prints the actual list of discrete local maxima on the grid (the list length depends on the additional parameters, see keyword **attractors** in section 2.5). It supplies the following information:

Loc.max	basin	i	j	k	value	min.dist	value diff.	x	y	z	atom	dist
1	1	94	108	121	57.3203	1.98	57.1037	0.000	0.000	1.650	C1	0.01
2	2	94	80	72	43.2548	1.98	43.0383	0.000	-1.400	-0.800	C3	0.03
3	3	94	136	72	43.2548	1.98	43.0383	0.000	1.400	-0.800	C2	0.03
4	4	61	108	143	0.2167	1.98	-57.1037	-1.650	0.000	2.750	H5	0.05
5	5	127	108	143	0.2167	1.98	-57.1037	1.650	0.000	2.750	H4	0.05
6	6	61	61	61	0.2165	1.98	-43.0383	-1.650	-2.350	-1.350	H9	0.06
7	7	127	155	61	0.2165	1.98	-43.0383	1.650	2.350	-1.350	H6	0.06
8	8	61	155	61	0.2165	1.98	-43.0383	-1.650	2.350	-1.350	H7	0.06
9	9	127	61	61	0.2165	1.98	-43.0383	1.650	-2.350	-1.350	H8	0.06

The above result is for a *control* file including the keyword **attractors** and the *basis* file removed form the directory (thus unknown for DGrid). *Loc.max* is the running number of the local maximum. The value *basin* indicates to which basin the local maximum belongs (notice that after the trajectory search the basins will be renumbered in ascending order of the volume size). The indexes *i j k* are the grid coordinates of the local maximum, whereas *x y z* are the corresponding cartesian coordinates. *value* shows the function value (here the density) at the grid position. The local maxima are given in descending order with respect to the function values. *min.dist* is the distance to the closest local maximum (given in the same units as the grid parameters). *value diff.* is the value difference between the local maximum and the closest local maximum. *atom* shows the symbol of the closest atom (at the distance *dist*).

Using the keyword **attractors** creates the file *locmax.str* with the coordinates of the local maxima written in the STR format (for visualization, cf. the Appendix). In this case, no search for the basins is performed. It is a good idea first to start with the keyword **attractors** and check the total number of basins found on the discrete grid. Then, with the keyword **attractors** still active, the number of basin can be reduced (without actually performing the time consuming trajectory search – for the basin search the keyword **attractors** needs to be commented out). The local maxima will be written to the output as well as into the file *locmax.str*. The number of attractors written out can

be changed appending additional parameters to the keyword **attractors**, cf. the section 2.5.

To start the basin search the keyword **attractors** must not be active!

2) Using *basis* file

In this case all local maxima on the discrete *property* grid will be tested by Hessian search and assigned to the corresponding attractor. Using the keyword **attractors** disables the basin search. Instead, the topology section appears in the output (here corresponding to the local maxima on previous page, now for a run with the *basis* file included):

```
PROPERTY:    rho[r]                  spin = alpha          pair = unknown


* * *  A T T R A C T O R S    I N    R E G I O N  * * *

-------------------------------------------------------------------------------------
Basin      x          y          z           value        Laplacian      curvature
-------------------------------------------------------------------------------------
  1     0.0000     0.0000     1.6428         59.1020          -              C1
  2    -0.0000    -1.4227    -0.8214         59.1020          -              C3
  3     0.0000     1.4227    -0.8214         59.1020          -              C2
  4    -1.6980     0.0000     2.7503          0.2163          -              H5
  5     1.6980    -0.0000     2.7503          0.2163          -              H4
  6    -1.6980    -2.3818    -1.3751          0.2163          -              H9
  7     1.6980     2.3818    -1.3751          0.2163          -              H6
  8    -1.6980     2.3818    -1.3751          0.2163          -              H7
  9     1.6980    -2.3818    -1.3751          0.2163          -              H8
-------------------------------------------------------------------------------------

Connection graph written to the file C3H6_HF_6-31G.g03.rho_r_a.graph.str
```

The string 'rho[r]' and following data show that the attractors were searched in the α-spin electron density field. For each attractor the cartesian coordinates are printed, followed by the property value at the attractor position. In the above example the value of the property Laplacian is not given, because all attractors correspond to atomic positions. Instead of the three curvatures only the atomic symbols are shown. The file *C3H6_HF_6-31G.g03.rho_r_a.graph.str* contains only the attractor positions. The bond paths are not included (for this a basin search would be necessary).

To start the basin search the keyword **attractors** must not be active!

In both cases, with or without the presence of the *basis* file, the number of basins can be larger then requested (for instance large number of core ELI-D basins for heavier elements). Then, the total number of basins can be reduced in three ways (also simultaneously):

- using the keyword '**compactify** *dist vdiff*'

In this case all local maxima closer then the distance *dist* and differing in the function values by less then *vdiff* will be reduced to single attractor. If omitted *vdiff* is set to the property value range.

- using the keyword '**top**=*iso*'

  Then, all local maxima inside *iso*-localization domains will be merged into single attractor.

- using the keyword **eli_core**

  Then, all local maxima inside the respective atomic core defined by ELI-D (internal table) will be merged into single basin.

In DGrid the integration of a property is done numerically on an equidistant grid. Thus, the precision depends on the mesh size and can be subject to large errors. In case of electron density – the standard charge determination – especially in the atomic core regions. There are two possibilities to compensate the errors:

- using the keyword **core_charge**

  In a small sphere around each atom the electronic charge is replaced by a value from an internal table (data computed from the basis sets of Clementi and Roetti for the atoms N - Xe, as well as from relativistic ADF [5] calculations for the atoms Cs-Lu). Of course, the atomic coordinates need to be specified in this case (either by the **pic_file** directive or from the property file in the DGrid format.

- using the refinement procedure (see the section 3.5, page 51, in chapter 3). The file name from the refinement procedure is given after the keyword **integrate**.

The grid from a solid state calculation often runs over the full unit cell. Then, using the assignment '**coordinates**=relative' gives all positions in the output relative to the grid parameters. Otherwise the positions are given in units used in the *property* file (i.e., absolute positions). If translation or mirror symmetry is given at the borders of the grid, then using the assignment:

symmetry=translation `i j k`

or

symmetry=mirror `i j k`

utilizes the respective symmetry in the `i`, `j`, and `k` direction (i.e., the first, second and third dimension). At least 1 symbol for the direction must be present. After the basin search all symmetry equivalent basins are added together and shown as single basin.

The *control* file must end with the keyword **end**. All data after this keyword will be ignored.

2.3 Basin file

After the basin search each grid point is assigned to its basin by an integer number. The data are written on separate *basin* file. It has a header followed by the basin values. The format of the header is similar to the one for the *property* file. Following is the example file *C3H6_HF_6-31G.g03.rho_r.bsn* created by the *control* file *rho_basins.inp* mentioned in the introduction:

```
BASINFILE   created by DGrid version 4.4   16-12-2008

:job was executed on:         Wed Dec 17 11:04:50 2008

property_grid=C3H6_HF_6-31G.g03.rho_r

basis_from_program:      GAUSSIAN03              basis=C3H6_HF_6-31G.g03

basins_for: rho[r]                               spin=both       pair=unknown

::C3H6   HF    6-31G


:atom       x          y          z
:-------------------------------------
 C       0.0000     0.0000     1.6428
 C       0.0000     1.4227    -0.8214
 C      -0.0000    -1.4227    -0.8214
 H       1.6980    -0.0000     2.7503
 H      -1.6980     0.0000     2.7503
 H       1.6980     2.3818    -1.3751
 H      -1.6980     2.3818    -1.3751
 H       1.6980    -2.3818    -1.3751
 H      -1.6980    -2.3818    -1.3751
:-------------------------------------


Energy=    -117.00804 Hartree               Electrons=   12 alpha  +    12 beta


:             lattice vectors
:-------------------------------------
:         a           b           c
:-------------------------------------
x:    1.000000    0.000000    0.000000
y:    0.000000    1.000000    0.000000
z:    0.000000    0.000000    1.000000
:-------------------------------------
```

```
:        origin           v_i         v_j         v_k
:-------------------------------------------------------
x:      -4.7000        9.400000    0.000000    0.000000
y:      -5.4000        0.000000   10.800000    0.000000
z:      -4.4000        0.000000    0.000000   10.200000

points:                   95          109         103
:-------------------------------------------------------
```

The first readable information is a line beginning with the string 'BASINFILE' and the DGrid version. Next lines contain the date and time of the calculation, followed by the name of the *property* file used to determine the basins, the name of the quantum chemical package that produced the *basis* file followed by the name of the *basis* file. In the next line is the name of the calculated property and the chosen spin and pair-spin.

The title (line starting with a double colon) is followed by the atomic coordinates aa well as well as the energy of the system and number of electrons. The next part contains the information about the grid region. The lattice vectors are meaningful only for solid state calculation. The lattice data are followed by the coordinates of the grid origin together with the three vectors describing the computed region and the number of points in each direction.

The last part of the header gives some information about the basins. It shows the volumes and the maximal grid value in the respective basins as well as a descriptor indicating to which atoms the respective basin can be attributed (this information was not given in version 4.3):

```
:basin    volume    maximum        x        y        z        i    j    k   descriptor
:-------------------------------------------------------------------------------------
    1    101.164     0.4259     -1.700    2.400   -1.400     30   78   30           H7
    2    101.166     0.4259      1.700   -2.400   -1.400     64   30   30           H8
    3    101.258     0.4259     -1.700   -2.400   -1.400     30   30   30           H9
    4    101.258     0.4259      1.700    2.400   -1.400     64   78   30           H6
    5    102.476    86.5096      0.000   -1.400   -0.800     47   40   36           C3
    6    102.716    86.5096      0.000    1.400   -0.800     47   68   36           C2
    7    132.494     0.4297     -1.700    0.000    2.700     30   54   71           H5
    8    132.516     0.4297      1.700    0.000    2.700     64   54   71           H4
    9    160.457    75.6824      0.000    0.000    1.600     47   54   60           C1
:-------------------------------------------------------------------------------------
```

This section is followed by integer values according to the basin the given point corresponds to. The grid points run in the x direction first:

```
: start of data
:-----------------------------------------------------------------------------------------
  3     3     3     3     3     3     3     3     3     3     3     3     3     3     3
  3     3     3     3     3     3     3     3     3     3     3     3     3     3     3
  3     3     3     3     3     3     3     3     3     3     3     3     3     3     5
  5     5     5     5     5     5     2     2     2     2     2     2     2     2     2
```

It is always a good idea to visualize the basins to see whether there are some unusual artifacts. Special visualization techniques should be utilized (this creates surfaces around blocks of constant data values). If the only possibility is to create isosurfaces then bear in mind that, for instance, the isosurface with value 3 is not only found around the basin 3, but also between the basins 1 and 4, 2 and 4, etc.

2.4 Output file

The output of a basin evaluation includes the information about the files read in, the calculation progress and some statistics. The progress during the basin search is shown by a scale, giving the number of points which are tested. This check can be repeated few times, till the zero-flux surfaces do not move anymore. After the basins are determined the integration section starts. If the **integrate** keyword was not given, then just the volumes of each basin will be determined.

If there is a property file assigned by the '**integrate**=' command, like the electron density grid *C3H6_HF_6-31G.g03.rho_r* in the *control* file *rho_basins.inp*, the corresponding property will be integrated over each basin. In the *output* file first the structure data are given, followed by the information about replaced charge if the **core_charge** keyword was used. For each basin its volume, the integral of the property over this volume (usually the charge), the value of the attractor (maximum of the scalar property that was used to define the basins) as well as its position are given:

```
                  rho[r]      rho[r]
BASIN   VOLUME   INTEGRAL    MAXIMUM     <X>      <Y>      <Z>      ATOMS      DIST
-------------------------------------------------------------------------------------
   1   101.164     0.9965      0.4326  -1.698    2.382   -1.375     H7         0.00
   2   101.166     0.9965      0.4326   1.698   -2.382   -1.375     H8         0.00
   3   101.258     0.9965      0.4326  -1.698   -2.382   -1.375     H9         0.00
   4   101.258     0.9965      0.4326   1.698    2.382   -1.375     H6         0.00
   5   102.476     6.0006    118.2040  -0.000   -1.423   -0.821     C3         0.00
   6   102.716     6.0025    118.2040   0.000    1.423   -0.821     C2         0.00
   7   132.494     0.9960      0.4325  -1.698    0.000    2.750     H5         0.00
   8   132.516     0.9964      0.4325   1.698    0.000    2.750     H4         0.00
   9   160.457     6.0017    118.2040   0.000    0.000    1.643     C1         0.00
-------------------------------------------------------------------------------------
TOT   1035.504    23.9833

Basin data written on file C3H6_HF_6-31G.g03.rho_r.bsn
```

If the *basis* file is not present then the attractor positions and values correspond to the closest grid point. For each attractor the closest atom (or atoms, see below) as well as the distance to this atom are given (in the above example the distance is always zero, because the attractors of electron density coincide with the atomic positions).

Following is the output example for the ELI-D basins of the C_3H_6 molecule:

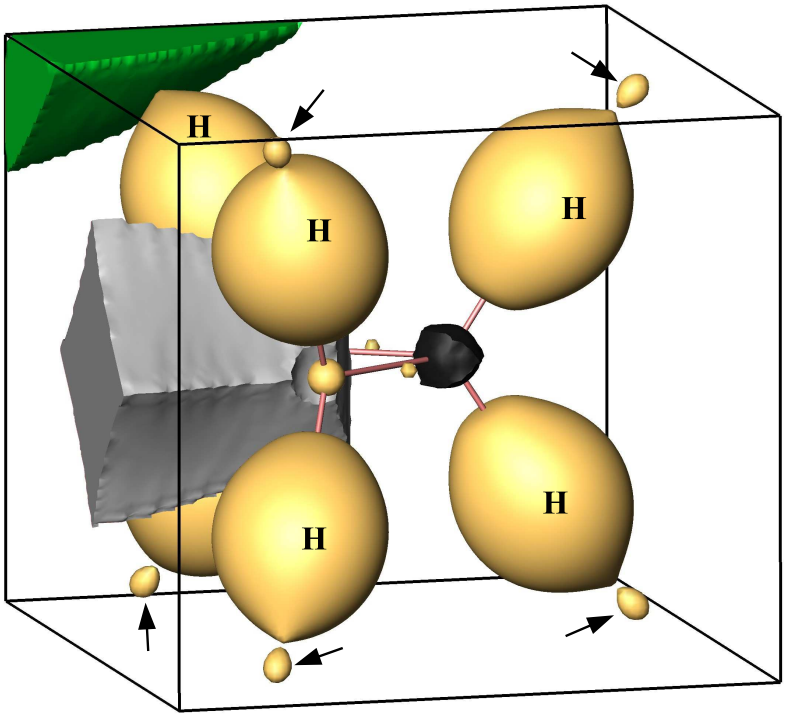

Figure 2.1: ELI-D for C_3H_6. Gold colored 1.675-localization domain (irreducible) with the spurious parts marked by arrows. Green: one of the spurious basins; grey: basin for the C-C bond; black: carbon core basin.

BASIN	VOLUME	rho[r] INTEGRAL	ELI-D[r] MAXIMUM	<X>	<Y>	<Z>	ATOMS	DIST	ECCNT
1	0.808	2.0926	16.7818	0.000	0.000	1.643	C1	0.00	
2	0.818	2.0999	16.7818	-0.000	-1.423	-0.821	C3	0.00	
3	0.818	2.0999	16.7818	0.000	1.423	-0.821	C2	0.00	
4	9.760	0.0008	1.6859	-4.288	-4.158	-2.400	H9	3.30	
5	9.760	0.0008	1.6859	4.288	-4.158	-2.400	H8	3.30	
6	9.760	0.0008	1.6859	-4.288	4.158	-2.400	H7	3.30	
7	9.760	0.0008	1.6859	4.288	4.158	-2.400	H6	3.30	
8	15.111	0.0008	1.6859	-4.288	-0.000	4.801	H5	3.30	
9	15.111	0.0008	1.6859	4.288	-0.000	4.801	H4	3.30	
10	27.914	1.7081	1.6971	0.000	0.000	-1.303	C3-C2	1.50-1.50	0.481
11	44.389	1.7196	1.6971	-0.000	1.128	0.651	C1-C2	1.50-1.50	0.481
12	44.392	1.7204	1.6971	0.000	-1.128	0.651	C1-C3	1.50-1.50	0.481
13	123.056	2.0841	7.0484	1.784	-2.443	-1.410	H8	0.11	
14	123.291	2.0902	7.0484	-1.784	-2.443	-1.410	H9	0.11	
15	124.153	2.0852	7.0484	-1.784	2.443	-1.410	H7	0.11	
16	124.310	2.0933	7.0484	1.784	2.443	-1.410	H6	0.11	
17	175.776	2.0889	7.0485	-1.784	0.000	2.821	H5	0.11	
18	176.517	2.0962	7.0485	1.784	0.000	2.821	H4	0.11	
TOT	1035.504	23.9833							

Basin data written on file C3H6_HF_6-31G.g03.elid_r_t_tr.bsn

Now there are much more basins then in the previous example using the electron density field (where each ρ-basin can be attributed to one of the 9 atoms). The first 3 basins show the carbon cores, with ELI-D attractors centered at the corresponding atoms, i.e., *DIST=0.00*, cf. also the black colored basin in Fig. 2.1. The next 6 basins (number 4-9) are due to the pure basis set, creating spurious ELI-D attractors behind the hydrogen atoms (at a distance of 3.30 bohr from the corresponding H atom), see Fig. 2.1. The ρ-integrals show that the corresponding basins (for instance the green one in Fig. 2.1) include almost no charge (0.0008 electrons).

In case of the basin number 10 (cf. the grey basin in Fig. 2.1) it can be seen from the above table that the corresponding attractor is located between the atoms C2 and C3. Two distances from the attractor to the respective atoms are given. In the above example the two distances (1.50-1.50) show that the ELI-D attractor has the same distance to both atoms. However, it is not located at the midpoint between the atoms, because the number following the distances – the eccentricity (perpendicular distance to the C2-C3 line) – is 0.481, i.e., nonzero. The last 6 basins (number 13-18) are attributed to the C-H bond, because of the absence of hydrogen cores in the ELI-D representation (thus, from a simple ELI-D analysis one cannot distinguish between the part that could be attributed separately to the hydrogen atom).

The distance of the property maximum to closest atoms together with the eccentricity is used to set up the basin descriptors (respectively the 'ATOMS' column in the output). It works in the following way. If the basin property maximum (i.e., the basin attractor) lies within the tabulated ELI-D core radius, then the corresponding atomic symbol is assigned to the respective basin. If the basin attractor is positioned between two atoms and the angle between the atom-atom interconnection line and the attractor-atom line is less than 40° then the basin descriptors is formed by the corresponding atomic symbols with a period in between. Otherwise, in the *basin* file, the symbol 'LP' (for lone-pair) is used. The basin descriptors should serve for a hint or help. It is not intended to be used as a rigid classification.

The integration section in the output is closed with the information about the file to which the basin data are written. This *basin* file can be used for further evaluations. For instance, the *basin* file can be used with the keyword **property**, like in the *control* file below (modified *rho_basins.inp* file):

```
::C3H6 density basins

:KEYWORDS
:---------------------------------------------------------------------
 property =C3H6_HF_6-31G.g03.rho_r.bsn
 integrate=C3H6_HF_6-31G.g03.rho_r

 output=.

 topology

end
```

Now, the basins are not recomputed by DGrid. Instead, they are directly read from the *basin* file *C3H6_HF_6-31G.g03.rho_r.bsn* created in previous run. The integration section will be the same as already discussed. Additionally, the 'Topology' section will appear in the *output* file. The first part of this section shows all attractors (see the corresponding output part on page 32 for the α-spin). This part is followed by the information about the saddle points:

```
* * *  (3,-1)   S A D D L E    P O I N T S    I N    R E G I O N  * * *

-------------------------------------------------------------------------------------------------------------
Basins       x       y       z       value   Laplacian          curvature           ellip   |V|/G      H
-------------------------------------------------------------------------------------------------------------
C3-H9    -1.0551 -2.0096 -1.1602      0.2688    -0.8053    0.5087  -0.6639  -0.6501    0.02   5.9015  -0.2529
H8-C3     1.0551 -2.0096 -1.1602      0.2688    -0.8053    0.5087  -0.6639  -0.6501    0.02   5.9015  -0.2529
H6-C2     1.0551  2.0096 -1.1602      0.2688    -0.8053    0.5087  -0.6639  -0.6501    0.02   5.9015  -0.2529
C2-H7    -1.0551  2.0096 -1.1602      0.2688    -0.8053    0.5087  -0.6639  -0.6501    0.02   5.9015  -0.2529
C1-H5    -1.0551  0.0000  2.3205      0.2688    -0.8053    0.5087  -0.6501  -0.6639    0.02   5.9015  -0.2529
H4-C1     1.0551  0.0000  2.3205      0.2688    -0.8053    0.5087  -0.6501  -0.6639    0.02   5.9015  -0.2529
C2-C3     0.0000 -0.0000 -0.9437      0.2205    -0.3137   -0.3762   0.3544  -0.2919    0.29   2.9991  -0.1569
C1-C2     0.0000  0.8173  0.4719      0.2205    -0.3137   -0.3762  -0.2919   0.3544    0.29   2.9991  -0.1569
C1-C3     0.0000 -0.8173  0.4719      0.2205    -0.3137   -0.3762  -0.2919   0.3544    0.29   2.9991  -0.1569
-------------------------------------------------------------------------------------------------------------

Connection graph written to the file C3H6_HF_6-31G.g03.rho_r.bsn.graph.str
```

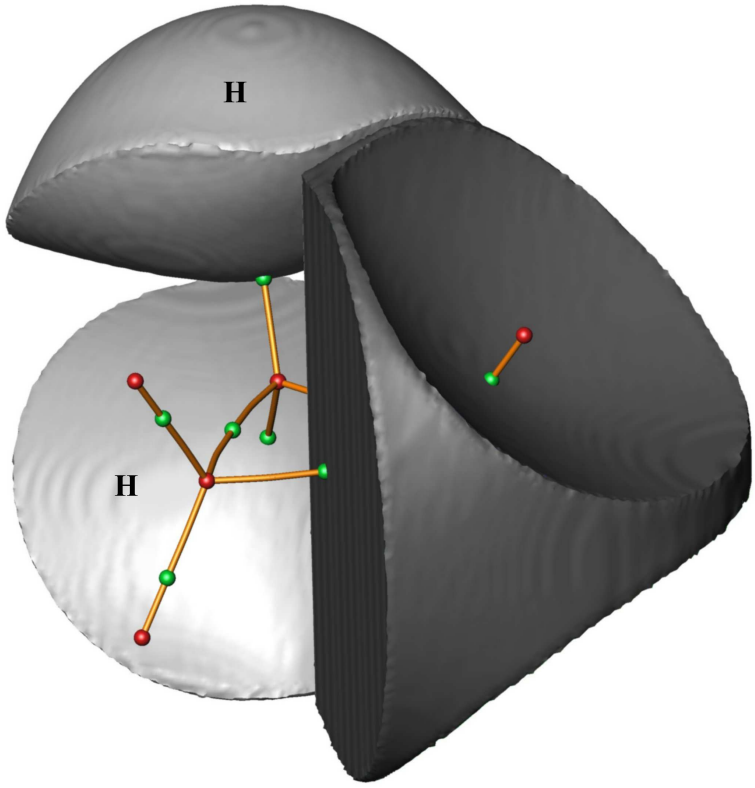

Figure 2.2: ρ-basins (cropped by the 0.001 bohr^{-3} isosurface of electron density) and the molecular graph for C_3H_6. Hydrogen basins are in grey, the carbon basin is black colored. Red spheres: attractors; green spheres: saddle points.

The first column shows between which basins the saddle point is located (here conveniently using the atomic assignment, otherwise the basin numbers would be used, e.g., for the ELI-D bond basins). The cartesian position of the saddle point as well as the property value (here the electron density) and its Laplacian at this position together with the corresponding principal curvatures are given in next columns. The ellipticity at the saddle point is given as $c_1/c_2 - 1$, with the two curvatures c_1 and c_2 perpendicular to the interaction path, whereby $|c_1| \geq |c_2|$. The virial ratio $|V|/G$ [27] is computed in DGrid from the kinetic energy density of non-interacting electrons (i.e., with $G = \tau$) and the local potential energy density is set to $V = \frac{1}{4}\nabla^2\rho - 2\tau$. Similarly, the energy density at the saddle point is given by $H = G + V$. If using the two last indexes for analysis be sure that the mentioned definitions apply to your wavefunction.

The connection graph *C3H6_HF_6-31G.g03.rho_r.bsn.graph.str* contains the information about the attractors and the saddle points as well as the interaction paths (here, because the electron density was analyzed, creating the molecular graph [24]). The data are written in the STR format described in the Appendix. The visualization of two hydrogen and one carbon basins (cropped by the 0.001 bohr^{-3} isosurface of electron density, see section 2.5, page 42) together with the molecular graph with the program Avizo are shown in Fig. 2.2. It can be seen that in the C-C bond paths are slightly bent lines connecting the carbon atoms. The saddle points (small green spheres) are located in the zero-flux surfaces (basin borders).

2.5 Keywords

In this section the keywords for the DGrid *control* file concerning the basin evaluation are described in alphabetical order. The keywords are read in case insensitive. The DGrid program is now inclusive the Basin part, i.e., the calculation and examination of basins is performed by DGrid. However, there is still a separate *input* file needed for such analysis.

The keywords are given either as variables, e.g., like '**property**=', or standing alone, like '**end**'. Some of them are followed by 1 or more numbers, e.g., '**top**=0.8'. If float number should be given then one must really input float numbers, otherwise the input routine will throw an error message. Thus, the command '**top**=1' would not be valid, because '1' is an integer number.

Often more keywords can be input in single line. But it is better to put every keyword separately in a line, because sometimes additional data are assumed per default. Some data even need to be written rigorous sequence. For instance the symmetry operation during the basin search using the **symmetry** keyword. Table 2.1 summarizes all the available keywords.

2.5.1 Attractors

With the keyword **attractors** the search for the basins is disabled. Instead, only the attractors will be searched and printed to the output. A separate file named *locmax.str*

Table 2.1: **Keywords and parameters available for the basin evaluation**

Keyword	Description	Value
attractors	write local maxima to file	first, last (integers, default 1-50)
compactify	unify local maxima	distance, value-diff (floats)
coordinates		relative, `cell`
core_charge	include core charge	
crop	crop basins with isosurface	file name for the property field
eli_core	reduce basins inside core	
end	end of input	
extrapolate_border	extrapolate border points	
format=	result file format (basins)	cube, `dgrid`, grace, lmto
gravity	basin center of gravity	
integrate=	integrate property	grid filename
localization_domains	search for localiz. domains	two isovalues (floats)
output=	specify output file	output filename
property=	scalar field defining basins	grid filename
reduced_step=	step for trajectories	step (float), default 0.04 bohr
result=	specify result filename	generic name of the property file
structure=	specify coordinates filename	filename with atomic coordinates
symmetry=	using symmetry	mirror, translation
top=	first cut	field value (`maximum`) (float)
topology	analyze critical points	

Default values are given in `Typewriter` font

will be created, containing the attractor positions written in STR format. The *basis* file must be present to perform the search, otherwise only the local maxima on the discrete grid will be printed, cf. page 31. Only first 50 local maxima will be printed (this is true also for the output into the *locmax.str* file). To output chosen number of local maxima use the command:

attractors <from> <to>

where <from> and <to> are integers giving the first and last maximum to be printed.

2.5.2 Compactify

With the keyword **compactify** all attractors (respectively the local maxima) closer than <dist> (in bohr) and differing in property value by less than <vdiff> will be assigned to the same basin, cf. page 32:

compactify <dist> <vdiff>

Especially if the *basis* file is not present, as is the case for solid state calculations, there can be large number of discrete local maxima. The keyword **compactify** enables a grouping of otherwise large number of (possibly artificial) basins. The defaul value for <dist> is twice the grid point distance. The default value for <vdiff> is the property range.

2.5.3 Core_charge

The keyword **core_charge** is used to invoke the inclusion of precomputed core charge into the integration procedure, see also page 33. The achieved integration precision is less then using the grid refinement (however, it is the only possibility in DGrid to enhance the integration precision if the *basis* file is not present, like for solid state calculations).

2.5.4 Crop

The keyword **crop** is followed by the name of the *property* which is used to crop the basins. This can also be a UNIX path to this file.

crop=<XY_prop> <iso>

The basins created by DGrid using the field given by the **property** keyword will be cropped by the isosurface of the field from <XY_prop> with the isovalue <iso> (default is 0.001). The basins must be created in the run (not read in). The ρ-basins in Fig. 2.2 on page 39 are cropped by the 0.001 bohr^{-3} isosurface of the electron density. If the cropping

procedure were not used then the basins would extend to the borders of the computed region.

Of course, with the cropping procedure part of the volume (outside the cropping isosurface) will not be considered for the integration. Also, the sum of the basin volumes will not recover the total volume of the computed region, cf. the line 'Volume difference' for a run with the *control* file *crop_basins.inp* (from the *example* directory) in the corresponding output:

```
TOT    505.776     23.8515

Volume difference:   -529.728
```

which shows the missing volume (the total volume of the grid-box is 1035.504 bohr^3). The charge in the excluded region of 529.728 bohr^3 volume will not be considered.

If a large grid-box around the molecule is used and electron density basins are searched, then due to the very low density (and density gradient) the search takes much more time because of corrections. In this case it is better to cut off the regions of low density with the **crop** command. If the default value removes more volume than demanded, use lower isovalue, for instance 1e-5.

2.5.5 Eli_core

The keyword **eli_core** is used to force a unification of core basins with attractors closer to the nucleus than the tabulated radius (given by the ELI-D radius of an inner core atomic shell, cf. page 32.

2.5.6 End

The keyword **end** is the last keyword in the *control* file that will be parsed. Any information after this keyword will be ignored.

2.5.7 Extrapolate_border

Keyword needed only for *property* files generated by older DGrid versions.

If the *basis* file is present then the keyword **extrapolate_border** disables the calculation of points outside the grid-box region (needed for more reliable basin search and property integration). The only reason for this keyword is to speed up the evaluation. It should be used only if the influence of the border parts is small, i.e., low gradients at the grid border and small contributions to the integrals.

2.5.8 Format

The keyword **format** is followed by the descriptor <fmt> for the format of the *result* file:

format=<fmt>

The format descriptor <fmt> can be one of (case insensitive):

- cube ↦ CUBE [1]
- dgrid ↦ *basin* file
- grace ↦ Grace [6]
- lmto ↦ TB-LMTO-ASA [26]

In the Cube and Grace formats the property values are given by float numbers. Routines which expect Basin values to be integers could fail!

2.5.9 Gravity

Using the keyword **gravity** changes the output section, where the volumes of basins are given, cf. page 36. Then, the property values are given for the center of gravity of the corresponding basin.

2.5.10 Integrate

The keyword **integrate** is followed by the name of the *property* file with a grid of values to be integrated over the basins. This can also be a UNIX path to this file.

integrate=<XY_prop>

The data in the *property* file must be written in one of the formats known to Basin (dgrid, lmto, cube). If the **integrate** keyword is omitted, then only the volumes of basins are determined and printed out. If the **integrate** keyword is given, but the **property** keyword is omitted then only the total integral within the grid region will be determined.

2.5.11 Localization_domains

With the keyword **localization_domains** the search for basins is disabled. Instead, regions enclosed by isosurfaces (of field given by the **property** keyword) will be determined. It has the following syntax:

localization_domains <iso1> <iso2>

If the value <iso2> is not given, then the command yields isosurfaces enclosing separate regions of values higher or equal <iso1>, i.e., the <iso1>-localization domains will be determined. Setting both values <iso1> and <iso2> restricts the (separate) regions to function values between <iso1> and <iso2>.

2.5.12 Output

The keyword **output** is followed by the generic name of the *output* file. This can also be a UNIX path to this file.

output=<XY_test>

The *output* file name will be generated by appending the string '.bas' to the generic name (i.e., 'XY_test.bas' for the above example). If the name of the *output* file is not explicitly given by the **output** assignment then the output will be written to the console. Using the assignment:

output=.

generates the *output* file name from the name of the *property* file by appending the string '.bas' to it.

2.5.13 Property

The keyword **property** is followed by the name of the *property* file with a grid of scalar data. The corresponding gradient field defines the basins. The name can also be a UNIX path to this file.

property=<XY_prop>

The data in the *property* file must be written in one of the formats known to Basin (dgrid, lmto, cube). If the **property** keyword is omitted then only the total integral of the integrated property (given by the **integrate** keyword) within the grid-box region will be determined.

If the **property** keyword is followed by the name of a *basin* file then the basin search will be disabled. Instead, the basins will be directly read from the *basin* file (thus saving time).

2.5.14 Reduced_step

With the keyword **reduced_step** the search for critical points (and the interaction paths) can be done with reduced maximal step, avoiding jumps between regions:

reduced_step=<step>

The defaul value for <step> is 0.04 bohr.

2.5.15 Result

The keyword **result** is followed by the generic name of the *basin* file. This can also be a UNIX path to this file.

result=<XY_test>

The *basin* file name will be generated by appending the string '.bsn' to the generic name (i.e., 'XY_test.bsn' for the above example). If the name of the *basin* file is not explicitly given by the **result** assignment then the *basin* file will be generated from the name of the *property* file by appending the string '.bsn' to it.

2.5.16 Structure

The keyword **structure** is followed by the name of the *structure* file. This can also be a UNIX path to this file.

structure=<XY_struct>

Using the *dgrid* format (and *cube* format as well) the atomic positions are already included in the *property* file. However, using the *lmto* format this information is missing. Then, the keyword **structure** reads this information from external file. This can also be an LMTO-PIC file (atomic symbol, atomic radius and the xyz coordinates in 1 line for each atom). Otherwise the data must be given in the STR format (described in the Appendix).

2.5.17 Symmetry

The keyword **symmetry** is followed by the descriptor <sym> for the symmetry operation and the direction in which the operation is acting:

symmetry=<sym> i j k

The symmetry descriptor <sym> can be one of (case insensitive):

- translation ↦ translate in given direction
- mirror ↦ mirror at the start and end of grid region

There must be at least one of the ijk descriptors for the directions. The vectors corresponding to the directions are defined in the *property* file).

2.5.18 Top

The keyword **top** is used to merge basins into larger sets:

top=<val>

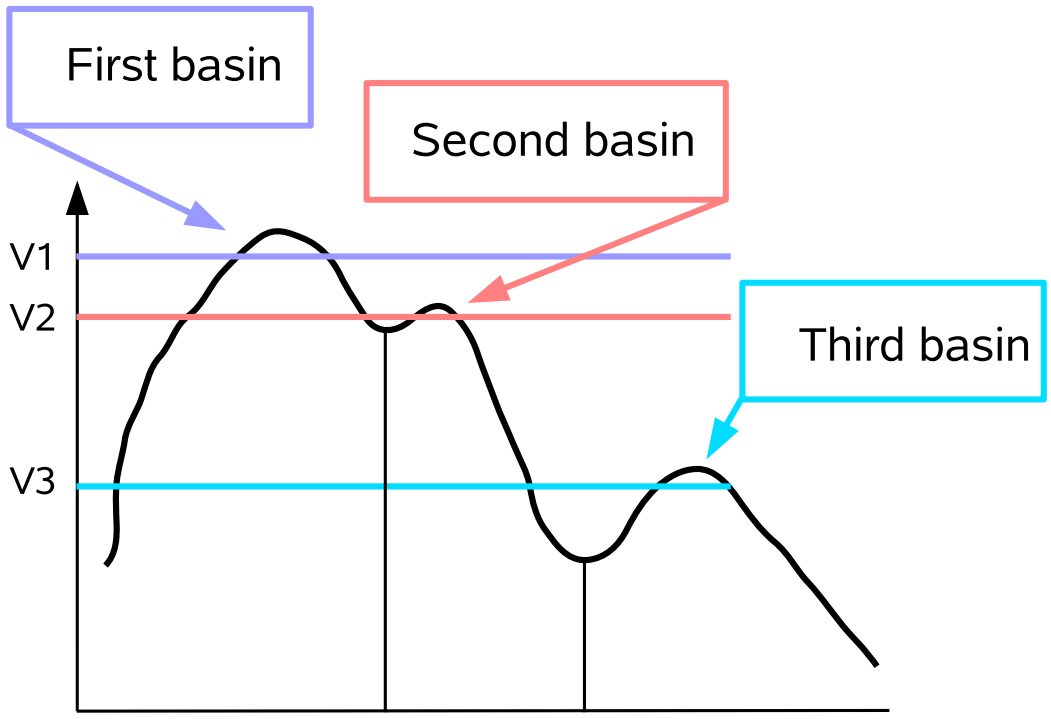

Figure 2.3: Usage of the **top** keyword. The vertical lines mark the basin borders (saddle points).

Each attractor has its corresponding basin. However, if the *basis* file is absent (solid state calculation) often larger number of discrete local maxima are found around the attractor position. With the *top* value <val> the program 'cuts' out all grid points with property

values higher than the *top* value and assign a basin to each separate closed region. Thus, choosing the value $V1$ in the Fig. 2.3 will mark all values above the $V1$ line and assign the points to the first basin. After that the basin search proceed the usual way to determine the remaining basins. Choosing as the top value $V2$ would first mark two separate regions and assign basins to it, thus avoiding possible spurious basins around the second attractor (if there were no spurious basins then the result would be the same as for the *top* value $V1$).

Sometimes it is desirable to merge few basins into a single one (creating a basin set). With $V3$ as the top value the third basin would form a separate region. However, the first and second basin would merge together (the values above the $V3$ line yield a single region). In this way it is possible conveniently merge together lot of basins and analyze such sets of basin as autonomous parts. In this case, see also how the merging of basins influences the results of the **topology** keyword.

2.5.19 Topology

Using the keyword **topology** invokes the topological analysis. It can be included in the *control* file already at the stage of basin search, respectively after the search, when the keyword **property** is followed by a *basin* file from previous run.

To perform the topological analysis the program needs to access the basins to determine the saddle points (bond critical points in case of the electron density). If some basins are merged together (either the core basins by the keyword **eli_core** or using the **top** keyword) then the topological analysis will find only the saddle points between the merged basin sets. For a detailed description of the results see page 39. The topology section creates a file (with the ending 'graph.str') in which the critical points as well as the interaction paths (in case of electron density – bond paths) are saved in the STR format (see Appendix). The connection graph can be visualized with suitable interface (available from author for the program Amira/Avizo). In case of the electron density evaluation the corresponding connection graph is called a molecular graph, cf. Fig 2.2.

Chapter 3

Utilities

The DGrid utilities are called from the command line with the following general syntax:

`dgrid` *filename* <kwr> <par>

The possible choices of keywords <kwr> and one or more parameters <par> depend on the input file (given by the *filename*), which can be an output file from quantum chemical program, a *basis* file, *grid* file, or *basin* file, respectively. The DGrid utilities work without a *control* file.

Quick reference to the syntax is given by the command:

`dgrid -u`

3.1 Conversion of QM package output

As described in previous sections, the DGrid program needs a *basis* file to evaluate the chosen properties. The *basis* file is written in a specific format, cf. section 1.4, and must be created in advance from data supplied by the quantum mechanical packages.

3.1.1 ADF

In case of the program ADF [5] (using the 'dmpkf' utility of ADF) the formatted TAPE21 *filename.kf* must be convert into the *basis* file using the DGrid command:

`dgrid` *filename.kf*

yielding the *basis* file named *filename.adf*. For localized orbitals computed with ADF the command must be extended as follows:

`dgrid` *filename.kf* `locorb`

3.1.2 Gaussian

In case of Gaussian [2] (using the 'formchk' utility of Gaussian) the formatted Checkpoint file *filename.fchk* must be convert into the *basis* file using the DGrid command:

`dgrid` *filename.fchk*

yielding the *basis* file named *filename.g98*, respectively *filename.g03*.

3.1.3 Molpro, Molcas, Turbomole

For the programs Molpro [3], Molcas [4], and Turbomole the basis data output must be written in Molden [19] format. Such output file *filename.molden* in Molden format must be convert into the *basis* file using the DGrid command:

`dgrid` *filename.molden*

producing the *basis* file named *filename.md*. In case of the program Turbomole include in the Molden output file manually as the second line the string:

[TURBOMOLE]

3.2 Basis file conversion

An older *basis* file can be converted into the new format using the DGrid command:

`dgrid` *basisfilename* `wrt`

This yields a copy of the *basis* file named *basisfilename_1* written in new format. If the system energy or the number of electron were not present in the older file then the corresponding data will be set to zero.

3.3 Structure file

With the DGrid command:

`dgrid` *basisfilename* `str`

a structure file will be created from the *basis* file. It contains the information about the atomic positions, cf. section 4.2 in the Appendix (however, without the color code and connections part). The resulting file named *basisfilename.str* can be used by an external program to visualize the molecular structure.

3.4 AO contributions

The atomic orbital expansions of the molecular orbitals are printed by the DGrid command:

`dgrid` *basisfilename* <thres> <occ>

where <thres> (float number) is the threshold for the AO coefficient and <occ> the threshold for the MO occupation. For instance:

`dgrid` C3H6_HF_6-31G.g03 0.3

prints out each molecular orbital from the *basis* file C3H6_HF_6-31G.g03 as a linear combination of atomic orbitals with coefficients of absolute value larger or equal 0.3 (the default value for the threshold is 0.1) and MO occupation larger than 0.0 (which is the default value for <occ>). Thus, the virtual orbitals will not be printed for the default <occ> value. The printing of virtual orbitals can forced by a negative value for <occ> (for instance -1.0).

3.5 Grid refinement

The integration done by the DGrid program is performed numerically on the property grid from a previous run. The precision of the integration crucially depends on the grid mesh size, which should not exceed 0.1 bohr, better is to use a 0.05 bohr mesh. Of course, the fine the mesh the better the integration result will be. However, decreasing the grid point distance by 1/2 increases the total number of grid points by the factor 8.

Higher mesh resolution is needed only in regions of highly non-linear behavior. DGrid can perform a refinement of the grid mesh by computing additional points in such regions. This is done by the DGrid command:

`dgrid` *gridfilename* `refine` <prec>

where <prec> is a float number for the precision of the integration. It is connected with the maximal change for the integral contribution in a box (determined by the mesh distances) around each of points chosen by the program.

For instance, the integration of the electron density grid (0.1 bohr mesh) saved in the file *C3H6_HF_6-31G.g03.rho_r* (created in a DGrid job using the *control* file *rho.inp*) yields the total charge of 23.9833 electrons in the grid region (the total number of electrons for C_3H_6 is 24). The result for this light molecule is close to the expected number, but the number of significant figures is definitely to large. Now, the electron density grid can be refined using the DGrid command:

`dgrid` C3H6_HF_6-31G.g03.rho_r `refine` 0.1

which performs the refinement with the precision of 0.1 electrons. With the above command the corresponding output will be written to the console. Following is the information concerning the refinement:

```
Refine grid for inhomogeneity >    0.100000

Sum of absolute inhomog. =   0.17093

0                                                  744
|----*----|----*----|----*----|----*----|----*----|
|>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>|

12320 values computed. Integrals written to file  C3H6_HF_6-31G.g03.rho_r.rfn

Sum of absolute changes =             0.0899256
```

It can be seen, that the DGrid refined the cells around 744 points computing 12320 additional values. The refinement data (refined integrals for grid points chosen by program) are saved in the file *C3H6_HF_6-31G.g03.rho_r.rfn* (i.e., the original file will not be changed). The subsequent integration (using the *control* file *ref_itg.inp* from the *example* directory) yields the total charge of 23.9686 electrons in the grid region. This is a small decrease of 0.0147 electron with respect to the original grid mesh.

Performing a deeper refinement using the precision parameter 0.01 results in much higher number of additional points to be computed. In this case the cells around 398 228 points are refined by additional 12 704 040 points. Of course, the time demand increases dramatically. The integration using the refined data gives the same result as in the coarse 0.1 refinement. This means that the previous integrals were already precise enough.

The impact of the refinement procedure is much higher for functions with highly nonlinear behavior, for instance for the electron density of heavier atoms or for the density Laplacian. With the example *control* file *lap.inp* the grid *C3H6_HF_6-31G.g03.lap_rho_r* of density Laplacian values can be computed (0.1 bohr mesh size). The integral of density Laplacian over the whole space equals zero. The integral over the computed region (ca. 1780 bohr^3 around the C_3H_6 molecule) should only slightly deviate from zero. However, the numerical integration yields -23.4476 bohr^{-5}. An refinement of the density Laplacian grid with the precision parameter 1.0 decreases the integral to -0.0141 bohr^{-5} (computing additional 427 704 points). Choosing the precision parameter 0.1 changes the integral to the value of -0.0160 bohr^{-5} (additional 3 320 296 points). While the previous result was closer to zero, it was not the proper value, as there is an contribution from the space outside the computed region. Decreasing the precision parameter further to 0.01 yields the total integral of the density Laplacian of -0.0177 bohr^{-5} over the computed region (at the expense of computing the huge amount of additional 15 908 254 points).

It is clear that the refinement procedure can be used this way with reasonable evaluation time for higher precision only for molecules composed of light atoms. Another possibility is to preform the refinement only over separate basins, thus saving evaluation time. This will be shown in next section on the example of integrals of the local source function.

3.5.1 Evaluation of the Source Function

The source function contribution $S(r, \Omega)$ is defined as the integral of the local source $LS(r, r')$ [21]

$$LS(r, r') = -\frac{1}{4\pi} \frac{\nabla^2 \rho(r')}{|r - r'|} \tag{3.1}$$

over given region Ω

$$S(r, \Omega) = -\frac{1}{4\pi} \int_{\Omega} \frac{\nabla^2 \rho(r')}{|r - r'|} \, dr' \tag{3.2}$$

It shows, how the region participate on the reconstruction of the electron density at the chosen reference position r. Positive values of $S(r, \Omega)$ mark the region Ω as a source, negative values as a sink.

To compute the local source field, the reference position needs to be specified in the *control* file. The specification is done with the keyword **ref_point** followed by the cartesian position. Following is the example *local_source.inp* form the *example* directory:

```
:-------------------------------------------------------------------|
::C3H6   HF    6-31G
:-------------------------------------------------------------------|

:-------------------------------------------------------------------
 basis=C3H6_HF_6-31G.g03

 output=.

:CHOOSE THE DESIRED PROPERTIES
:---------------------------------------
 compute=rho
 compute=LS
:---------------------------------------

 ref_point= 0.0  0.0 -0.9437

 mesh=0.05   4.0

END
```

The reference position is usually a saddle point in the electron density. The above position is the electron density saddle point between the atoms C2-C3, cf. the output of the *topology* run on page 39. Together with the local source the electron density is computed (later used for integration over density basins). The mesh size is set to 0.05 bohr. This is

the recommended value to get better basin borders (zero-flux surfaces). In the resulting *grid* file for the local source contains the following lines:

```
property:   Local-source[r]                    spin=both       pair=unknown

Ref_point=    0.0000    0.0000   -0.9437        ref_density=    0.22054
```

Thus, the *grid* file bears not only the information about the reference position, but also the density at that position. The (exact) integral of the local source over the whole space recovers this value. As the computed region is just a part of total space and the integration is done numerically, the integral will deviate from the density at the reference position. As proposed by C. Gatti [22], the source function contribution $S(r, \Omega)$, i.e., the local source integrals over the basins, are given by DGrid as percentage source contribution to the density at the reference point.

Integrating the above local source field from the file *C3H6_HF_6-31G.g03.ls_r* yields value much larger than the density $\rho(r_b) = 0.22054$ bohr^{-3} at the saddle point. It gives 259% of the expected value. Refining the local source grid by the precision parameter 0.1 (computing additional 203 706 points) yields by integration about 73% of the reference density. Increasing the precision with the refinement parameter 0.01 (evaluating 1 907 200 additional points) recovers by integration 100.12% of the reference density. Further increase of the precision (refinement parameter 0.001) does not (for the whole grid-box region of this molecule; however, see below for a basin choice) significantly change the recovering of the density at the reference position (100.10% – at large distances there is a charge depletion, i.e., positive electron density Laplacian; those remaining contributions from the missing volume will reduce the recovering value to 100%).

The above procedure just shows a test of the quality of the source function. In the analysis of the source function contribution one is interested, how particular basins contribute to the density at the reference (saddle) point. One possibility is to refine the total local source grid and then perform an integration of that grid over the basins. The basins can be created by a separate run with the above *C3H6_HF_6-31G.g03.rho_r* grid, cf. input on page 30 (possibly omiting the **integrate** keyword), yielding the *basin* file *C3H6_HF_6-31G.g03.rho_r.bsn*. The *control* file for the source function contributions with the refined local source grid and the mentioned *basin* file reads as follows:

```
:-------------------------------------------------------------------|
::C3H6 source function contribution over density basins
:-------------------------------------------------------------------|

:--------------------------------------------------------------------
 property =C3H6_HF_6-31G.g03.rho_r.bsn
 integrate=C3H6_HF_6-31G.g03.ls_r.rfn

 output=.

END
```

DGrid reads the basin data directly from the *basin* file without the time consuming search. The *C3H6_HF_6-31G.g03.ls_r.rfn* file contains not only the refinement data but also the name of the density grid file, which will be read in by DGrid. Following is a part of the *output* file:

```
          Local-source[r]       rho[r]
BASIN    VOLUME    INTEGRAL      MAXIMUM      <X>       <Y>       <Z>      ATOMS           DIST       ECCNT
-------------------------------------------------------------------------------------------------------------
   1 R 121.801     36.7963     118.2040    0.000     1.423    -0.821     C2              0.00         -
   2 R 121.925     36.5707     118.2040   -0.000    -1.423    -0.821     C3              0.00         -
   3 R 130.440      7.5788     118.2040    0.000     0.000     1.643     C1              0.00         -
   4 R 148.137      3.8607       0.4326    1.698    -2.382    -1.375     H8              0.00         -
   5 R 148.139      3.8607       0.4326   -1.698     2.382    -1.375     H7              0.00         -
   6 R 148.235      3.8603       0.4326    1.698     2.382    -1.375     H6              0.00         -
   7 R 148.237      3.8603       0.4326   -1.698    -2.382    -1.375     H9              0.00         -
   8 R 150.357      1.8895       0.4325    1.698     0.000     2.750     H4              0.00         -
   9 R 150.399      1.8898       0.4325   -1.698     0.000     2.750     H5              0.00         -
-------------------------------------------------------------------------------------------------------------
TOT   1267.669    100.1673

Volume difference:   -505.259
```

Each basin number is followed by an 'R' showing that the integrated data were refined. The volume difference at the end is a hint that the basins were cropped. Thus, part of the volume of the computed region was not considered. The influence of the cropping is acceptable as the recovering of the electron density at the reference position amounts to ca. 100.2%.

Due to the symmetry of the molecule the source function contributions of the two carbon atoms C2 and C3, respectively, should be the same (the reference position is exactly at the midpoint between the atoms). The small deviation (36.8% against 36.6%) is due to differing volumes of basin number 1 and 2. And the differing volumes are due to the odd number of grid points along the y axis (257, as the consequence of the keyword '**mesh**=0.05 4.0'). In this case one cannot, of course, separate the distance into two parts with equal number of points (this can be achieved by the keyword **vectors**). Additionally, it can be seen that the carbon C1 is contributing as a source of the density at the saddle point to much less extent (ca. 7.6%) than the two other carbons. The hydrogens (H6-H9) connected to the carbons C2 and C3 contribute with their basins approximately as half as much (ca. 3.9%) as the C1 basin. The two hydrogens (H4 and H5) on carbon C1 contribute again just as half as much (ca. 1.9%) as the other hydrogens to the density at the saddle point.

In the above example the evaluation of the source function was not too expensive. With this advantage it was possible to refine the whole local source grid. This is not the case when heavy atoms and large basis sets are involved. Such refinement could take hours! Then it is possible to perform the refinement for local source in the desired basins only. The command for the refinement now includes the *basin* filename and reads as follows:

`dgrid` *gridfilename* `refine` <prec> *basinfilename* <bas1> <bas2> etc.

where <bas1> <bas2> etc. is a sequence of integer numbers for the chosen basins inside which the refinement should be performed.

As an example let us take the same data as before, but perform the source function evaluation for the basins number 1, 3, 4, and 8 (which corresponds to basins with non-equivalent contribution, i.e., for the carbons C1 and C3, and the hydrogens H8 and H4, respectively):

`dgrid` *C3H6_HF_6-31G.g03.ls_r* `refine` 0.01 *C3H6_HF_6-31G.g03.rho_r.bsn* 1 3 4 8

The information about the basins is included in the resulting refinement file *C3H6_HF_6-31G.g03.ls_r.rfn* (respectively *.rfn_1* if there was a file with that name from a previous run). For this refinement 276 756 additional points were computed (cf. the 1 907 200 points for the refinement with the same precision parameter in the total box-region as given above). However, the participation of basin number 1 on the recovering of the saddle point density is not satisfactory, cf. the output on next page where the refined basins are marked with 'R'. It gives only 22.9%, which is too low compared to the value of 36.8% for this basin in the case of the total refinement.

```
          Local-source[r]       rho[r]
BASIN   VOLUME    INTEGRAL      MAXIMUM      <X>       <Y>       <Z>      ATOMS           DIST       ECCNT
-------------------------------------------------------------------------------------------------------------
   1 R 121.801     22.9303     118.2040    0.000     1.423    -0.821     C2              0.00         -
   2   121.925    -89.6793     118.2040   -0.000    -1.423    -0.821     C3              0.00         -
   3 R 130.440      7.6039     118.2040    0.000     0.000     1.643     C1              0.00         -
   4 R 148.137      3.8610       0.4326    1.698    -2.382    -1.375     H8              0.00         -
   5   148.139      3.8633       0.4326   -1.698     2.382    -1.375     H7              0.00         -
   6   148.235      3.8629       0.4326    1.698     2.382    -1.375     H6              0.00         -
   7   148.237      3.8629       0.4326   -1.698    -2.382    -1.375     H9              0.00         -
   8 R 150.357      1.8899       0.4325    1.698     0.000     2.750     H4              0.00         -
   9   150.399      1.8915       0.4325   -1.698     0.000     2.750     H5              0.00         -
-------------------------------------------------------------------------------------------------------------
TOT   1267.669    -39.9134

Volume difference:   -505.259
```

The reason for this discrepancy is that now the precision parameter is valid for the integral of just few basins, i.e., the relative precision is lower. For the integral of the whole box-region the precision is roughly 0.01 of 0.22 (saddle point density), i.e., ca. 5% error, whereas for the chosen basins (ca. 1/2 of the box-region volume) it corresponds to 0.01 of 0.11, i.e., around 10% error. The results are safe by using the precision parameter of 0.001 for both routes (actually 0.03 would be enough). Of course, this increases the amount of computed points.

The stability of the integration should be checked with increased precision of the refinement. Especially, when heavier atoms are involved.

3.6 Operations on single grid

The utility operates on data of a single *property* grid. It is invoked by the DGrid command:

`dgrid` *gridfilename* `op1`

DGrid reads in the *grid* file and asks for the desired operation which can be one of:

```
Available operations:
-------------------------------------------------------------------------------------
  +           -               x             ->  number
  /           ^               over          ->  number

  mirror      translation     inversion     ->  {i, j, k, in, jn, kn}
                              convert       ->  {cube, dgrid, grace, lmto}
                              spin          ->  {alpha, beta, both, singlet, triplet}
                              pair          ->  {alpha-alpha, beta-beta, triplet-pair}

  sqr         crop            reduce
  elf2eli     eli2elf
  trp2aa      aa2trp

  save        quit
-------------------------------------------------------------------------------------
```

The chosen operation is performed at each grid position. Some of the operations need additional parameters. After the operation is done, DGrid awaits the next operation. The transformed grid data are held in the memory. The procedure is finished either by exiting the program without saving the data (command '**quit**') or by saving the data (command '**save**') in which case the resulting grid will be written to the file with the same name as the original file with an appendix reflecting the chosen operation. Following is the description of the operations.

The operations '+, −, ×, /, ^, **over**' need a float number (integer will be converted to float) as a parameter in the input line. For instance the command '× 5' will multiply the property value at each grid point by 5. With '**over** 2' the value 2 will be divided by the property value at each grid point. The operation **sqr** takes the square root of each grid value and need no additional parameters.

The operations '**mirror**, **translation**, **inversion**' perform the given symmetry operation of the grid. An additional parameter, which is one of '`i j k`', respectively '`in jn kn`'is needed, determining at which position the operation is acting. Thus, the command '**mirror** `j`' will mirror the grid in the `j`-direction at the origin of the `j`-axis, whereas '**mirror** `jn`' will perform the mirroring at the end of the `j`-axis. The resulting grid will be twice as large as the original one. The command '**inversion** `k`' will perform an inversion of the grid in the `k`-direction around the grid midpoint of the `ij`-plane. This utility can be used for molecules or solid with the mentioned symmetry to save computational time, for instance, to compute just one octant for a homonuclear dimer and enlarge the grid by mirroring.

The operations **spin** and **pair** are used to set the appropriate value for the spin and pair-spin variable in a grid file.

The last operation which needs an additional parameter is **convert**. It is followed by the string determining to which format the grid file should be converted. The possible formats are given on page 20. It is also possible to use this utility to convert files written with older DGrid versions into the new 4.4 format (in some cases not all data will be available, like e.g. the energy of the system, which will be set to a default value).

The **crop** operation will ask for the name of the file containing the molecular structure (in STR or PIC format). Then all grid values within the atomic radii (given in the structure file) will be set to zero (cropped).

The **reduce** operation can be used only for grid files with odd number of points in each direction (i.e., even number of intervals). The number of grid points will be reduced in such way, that every second point in each direction will be removed without changing the size of the region-box (doubling the mesh size).

With the operations **elf2eli** and **eli2elf** the ELF grid files can under specific conditions formally be converted into ELI-D grid files (and vice versa). The conditions are that the property must stem from a single-determinantal wavefunction, the occupations must not be fractional and in case of triplet-coupling only restricted basis is allowed.

The operations **trp2aa** and **aa2trp** convert ELI-D files for closed-shell calculation, in which case there is a simple conversion factor between the single spin-channel and the triplet-coupled form of ELI-D [7].

3.7 Operations on two grids

Sometimes it is useful to add, subtract, multiply, or divide the values in two grids, for instance, to create a grid with promolecular electron density or compute a density difference maps. The operation can be accomplished by the DGrid command:

`dgrid` *gridfilename1* <oper> *gridfilename2*

DGrid reads in the two *grid* files (which must have identical mesh size and grid dimensions) and performs the required operation at each grid position. The operation <oper> can be one of:

- $+$ $\mapsto$ add the grid values $f_1(x,y,z) + f_2(x,y,z)$
- $-$ $\mapsto$ subtract the grid values $f_1(x,y,z) - f_2(x,y,z)$
- x $\mapsto$ multiply the grid values $f_1(x,y,z) * f_2(x,y,z)$
- $/$ $\mapsto$ divide the grid values $f_1(x,y,z)/f_2(x,y,z)$

The resulting grid values, with the same grid dimensions as the original ones, will be written to the file named *grid.add*, *grid.sub*, *grid.mult*, or *grid.div*, respectively. The header of the resulting *grid* file is identical with the header of the first *grid* file on the input line (only the title is changed). This means that, for instance the spin labels will be the ones from the first *grid* file. Such data, if desired, must be changed manually.

3.8 Cropped basins

This utility works on basin files created by the DGrid program, i.e., grids of integer values yielding the basins.

For a molecule the outer basins extend to infinity, i.e., such basins are bounded by the region-box chosen for the grid calculation. However, it might be useful to crop the basins by an isosurface of another property field, for instance the electron density. Using the 0.001 $e^-/bohr^3$ isosurface would correspond to basins cropped by the 'molecular envelope'. The procedure is accomplished by the DGrid command:

`dgrid` *basinfilename* `crop` <val> *gridfilename*

DGrid reads in both the *basin* file and the *property* file (the files must have the identical grid dimensions and mesh size). Each grid point outside the <val>-isosurfaces (i.e., outside the <val>-localization domains) will be set to zero (i.e., marked as no basin). The resulting cropped basin grid, together about the information about the isovalue and the *property* file name, will be written to the file with the same name as the *basin* file with the string '*.crop*' appended.

The procedure yields the same result as a basin search with the command '**crop**=' included in the *control* file, cf. page 42. The utility can be used if one do not want to perform a new basin search for different cropping isovalues.

3.9 Basin intersections

This utility works on basin files created by the DGrid program, i.e., grids of integer values yielding the basins written in the DGrid format.

Basins computed for two different scalar fields yield two different sets of space partitioning. It can be of interest to examine, how a particular basin from the first set is intersected by the basins of the second set [23]. Of course, not all basins of the second set will intersect the chosen basins (but at least one will do). The intersection utility has the following syntax:

`dgrid` *basinfilename1* `intersect` <bas> *basinfilename2*

DGrid reads in the two *basin* files (which must have identical mesh dimensions). In the first *basin* file the basin number <bas> is chosen. Then, the intersections of this basin by the basins of the second set is determined. The resulting basin grid containing only the intersections of basin number <bas> is written to the file with the same name as the first *basin* file with the string '.<*bas*>*_isect*' appended. The extent (in percent) to which the basins of the second set intersect the chosen basin is given in the output (written to the console).

For example, for the C_3H_6 molecule two grids for the electron density and ELI-D, respective, will be computed using the following *control* file (with the Gaussian03 *basis* file *C3H6_HF_6-31G.g03* from the *example* directory):

```
:TITLE
:-------------------------------------------------------------------|
::C3H6   HF    6-31G
:-------------------------------------------------------------------|

:KEYWORDS
:-------------------------------------------------------------------
 basis=C3H6_HF_6-31G.g03

 compute=rho
 compute=ELI-D     triplet

 mesh=0.05  4.0

END
```

Two grid files will be written, namely *C3H6_HF_6-31G.g03.rho_r* for the electron density and *C3H6_HF_6-31G.g03.elid_r_t_tr* for the ELI-D. The corresponding *basin* grids will be created with the following *control* file (for the ELI-D grid by changing the property file name accordingly):

```
:TITLE
:-------------------------------------------------------------------|
::C3H6 basins
:-------------------------------------------------------------------|

:KEYWORDS
:-------------------------------------------------------------------
 property=C3H6_HF_6-31G.g03.rho_r
 crop    =C3H6_HF_6-31G.g03.rho_r            0.001

END
```

Note that the basins will be cropped by the 0.001 isosurface of the electron density. The basin search yields the file *C3H6_HF_6-31G.g03.rho_r.bsn* with 9 density basins, respectively the file *C3H6_HF_6-31G.g03.elid_r_t_tr.bsn* with 12 ELI-D basins.

According to the QTAIM approach the electron density basins, cf. Fig. 2.2 on page 39, describe the atoms in molecule [24]. Because non-nuclear maxima are not present in case of the C_3H_6 molecule, the number of the density basins corresponds to the number of atoms. In case of ELI-D the number of basins is larger than for the electron density, because there are separate basins corresponding to the ELI-D attractors between the carbon atoms (which can be assumed as descriptors for the C-C bonds), cf. Fig. 2.1 on page 37.

There are always two possibilities for the intersection analysis depending on the choice of the intersecting set. In the above case either an ELI-D basins is intersected by the electron density basins or vice versa.

With the command:

`dgrid` *C3H6_HF_6-31G.g03.elid_r_t_tr.bsn* `intersect 4` *C3H6_HF_6-31G.g03.rho_r.bsn*

the ELI-D basin number 4, which corresponds to the basin between two carbon atoms, will be intersected by the electron density basins. The intersected ELI-D basin will be written to the file *C3H6_HF_6-31G.g03.elid_r_t_tr.bsn.4_isect*. The output will be written to the console. It prints the following information concerning the intersection of the ELI-D basin:

```
Intersections of ELI-D[r] basin No. 4 (C3-C2):

 rho[r]
 basin  descriptor      volume           %
-------------------------------------------
   7            C2      13.682        49.82
   9            C3      13.779        50.18
-------------------------------------------

Basin intersections written to file C3H6_HF_6-31G.g03.elid_r_t_tr.bsn.4_isect
```

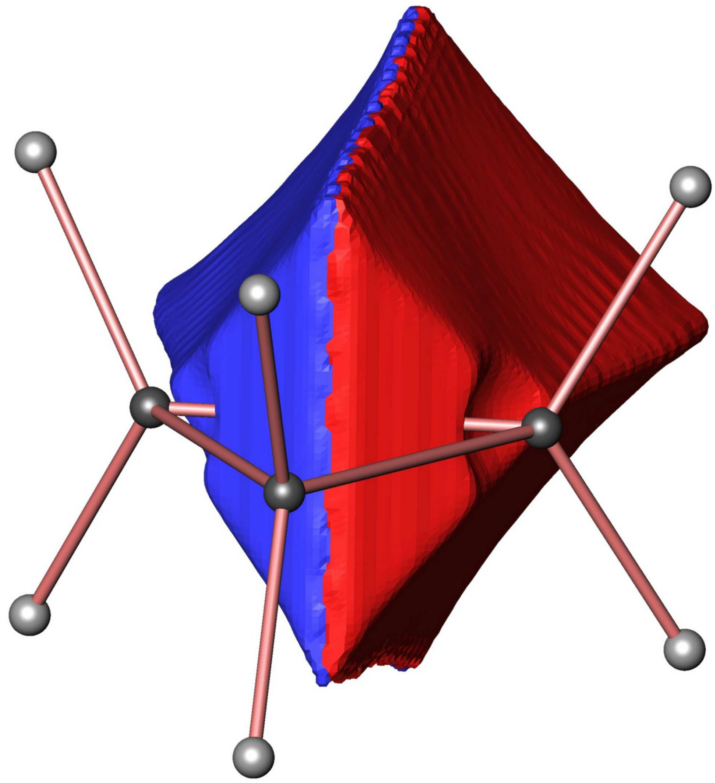

Figure 3.1: ELI-D basin (cropped by the 0.001 bohr^{-3} isosurface of electron density) of C_3H_6 intersected by electron density basins.

The result shows that the ELI-D basin number 4 (between the atoms C2 and C3) is intersected by two density basins, namely the ones corresponding to the atoms C2

and C3, respectively. The density basins participate to equal extent on the ELI-D basin volume (as given by the symmetry). The small deviation from 50% is due to the odd number of grid points along the y direction (257). The resulting file *C3H6_HF_6-31G.g03.elid_r_t_tr.bsn.4_isect* can be used either for visualization, cf. Fig. 3.1 or for the integration of the electron density to yield the charges within each intersection. The ration of the intersection charges can be connected with the bond polarity [23]. In the above case, due to the symmetry, the intersection charges must be again identical, yielding a non-polar bond.

Other possibility is to intersect chosen electron density basin by the ELI-D basins. With the command:

`dgrid` *C3H6_HF_6-31G.g03.rho_r.bsn* `intersect 8` *C3H6_HF_6-31G.g03.elid_r_t_tr.bsn*

the electron density basin number 8, which corresponds to the basin of the C1 atom, will be intersected by the ELI-D basins. The basin intersections will be written to the file *C3H6_HF_6-31G.g03.rho_r.bsn.8_isect*. On the console the following output will appear:

```
Intersections of rho[r] basin No. 8 (C1):

 ELI-D[r]
 basin  descriptor      volume           %
-------------------------------------------
   3            C1       0.819        1.09
   5         C1-C3      13.772       18.33
   6         C1-C2      13.772       18.33
   7            H6       0.003        0.00
   8            H4      23.329       31.05
   9            H7       0.002        0.00
  10            H5      23.423       31.18
  11            H9       0.006        0.01
  12            H8       0.006        0.01
-------------------------------------------

Basin intersections written to file C3H6_HF_6-31G.g03.rho_r.bsn.8_isect
```

The C1 electron density basin is intersected by many ELI-D basins. The first intersection with the ELI-D descriptor C1 is hidden solely inside the C1 density basin, because it corresponds to the ELI-D core basin of the C1 carbon atom. The ELI-D core basin take up just a small fraction (ca. 1%) of the atomic (density) basin volume.

The next two intersections are due to the ELI-D bond basins (C1-C3 and C1-C2 bond). In Fig. 3.2 only one of the intersections (green colored) is visible, due to the orientation of the molecule. The other two large intersections of the density basin are the ones with the ELI-D basins attributed to the hydrogen atoms H4 and H5 connected to the carbon atom C1. In Fig. 3.2 those intersections are visualized as the red and blue parts of the density basin. Again, due to the symmetry of the molecule, the red and blue intersections are of same size. The ELI-D basins of the remaining hydrogen atoms are just 'touching' the C1 atomic basin and have almost no contribution to the volume of that basin.

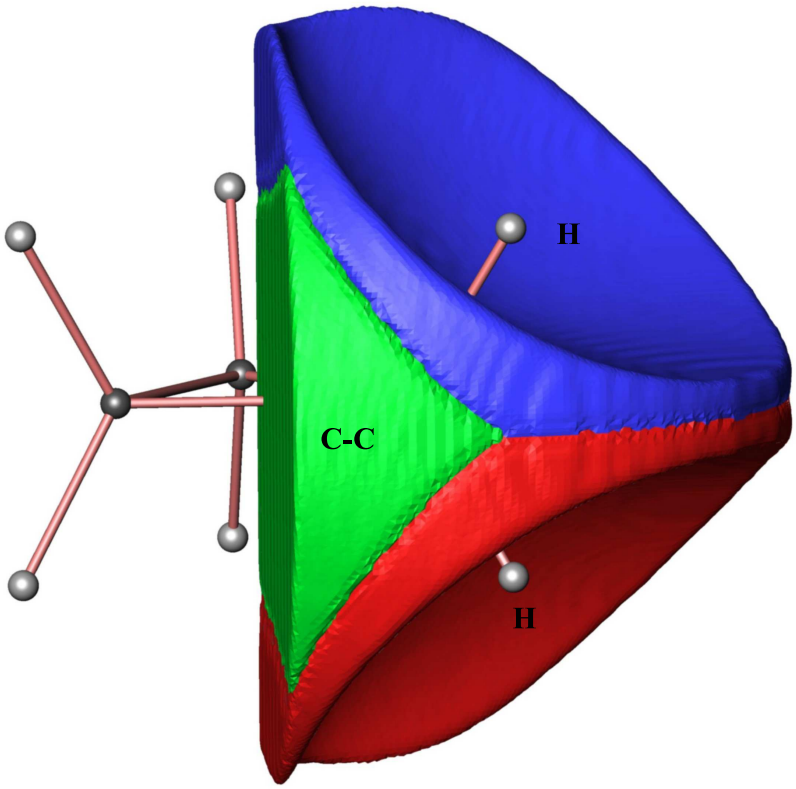

Figure 3.2: Electron density basin (cropped by the 0.001 bohr^{-3} isosurface of electron density) of C_3H_6 intersected by ELI-D basins.

The intersection basin file can be used (with keyword **property**=) to integrate the charge within the intersections. For the intersections of the atomic basin C1 such integration shows, that each ELI-D bond (C-C) basin participates with 0.86 electrons on the total charge of the atomic basin, whereas each of the closest ELI-D hydrogen basins participates with 1.09 electrons (the C1 atomic population equals 5.99 electrons).

Especially in case of the ELI-D hydrogen basins (there are no separate ELI-D basins for the C-H bond, because of the absence of a ELI-D hydrogen core region) the intersections are a good tool to estimate the bond polarity. With the command:

`dgrid` *C3H6_HF_6-31G.g03.elid_r_t_tr.bsn* `intersect 10` *C3H6_HF_6-31G.g03.rho_r.bsn*

the ELI-D basin for the H5 atom (respectively the C1-H5 bond) is intersected by the electron density basins. The output shows that the ELI-D basin volume of the H5 atom is divided between two density basins, cf. Fig. 3.3:

```
Intersections of ELI-D[r] basin No. 10 (H5):

 rho[r]
 basin  descriptor      volume           %
-------------------------------------------
   4            H5      46.757        66.62
   8            C1      23.423        33.38
-------------------------------------------
```

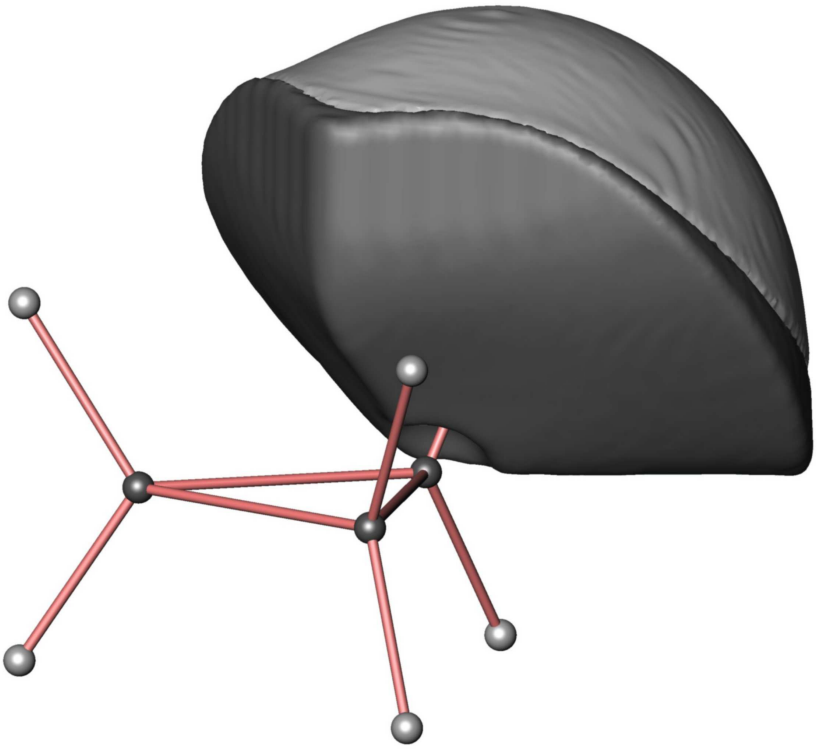

Figure 3.3: ELI-D basin (cropped by the 0.001 bohr^{-3} isosurface of electron density) for the hydrogen atom H5 of C_3H_6 intersected by the electron density basins. C-intersection is colored dark, H-intersection is in gray.

Note that the volumes of the outer atoms of a molecule depend on the choice of the cropping isosurface (respectively on the box-region size). Thus, the relative volume participation of the intersections on the basin volume should be used with caution in the analysis. More reliable is to integrate the electron density within the intersections, because the charge contributions outside the cropped volume rapidly decreases. For the above case it yields:

```
Intersected  ELI-D[r]  basin nr. 10

                  rho[r]
BASIN   VOLUME    INTEGRAL      <Qx>     <Qy>     <Qz>      ATOMS     DIST        ECCNT
-------------------------------------------------------------------------------------
   4 R  46.757      0.9802     -1.787   -0.000    2.806     H5        0.11          -
   8 R  23.423      1.0910     -0.833    0.000    2.116     C1-H5     0.96-1.07  0.059
-------------------------------------------------------------------------------------
TOT     70.181      2.0711     -1.285    0.000    2.443
```

Despite the larger volume contribution (66.6%) of the H5 density basin on the ELI-D H5-basin, the ration of the charges within the two intersections is almost 1:1, i.e., indicating a non-polar C-H bond.

Chapter 4

Appendix

4.1 Comments to property calculations

4.1.1 Grid mesh

For the visualization (with an external program) a distance of 0.1 bohr between the grid points (0.1 bohr mesh size) is sufficient. However, only for light elements (Li – F) the precision of charge integration is satisfactory for such mesh. For a steep property with lot of critical points (like ELI-D or density Laplacian) a 0.05 bohr mesh (or better) is recommended. For heavier elements (metals) even much more dense mesh would be necessary, however, creating huge grid files. In this case, use a 0.05 bohr mesh with additional refinement performed with the `refine` utility, cf. section 3.5. This approach is possible only if the *basis* file is present (i.e., not possible yet for solid state calculations).

4.1.2 Orbitals

The orbital amplitude is computed with the assignment **compute**=phi <spin>. Only 1 orbital (keyword **occupation**) and particular spin channel must be chosen. In momentum space (keyword **space**=momentum) the orbitals have in general a real and an imaginary part. The respective part can be chosen with the assignment **wf_part**=real, respectively **wf_part**=imaginary. The default value (i.e., without the appropriate choice) is `real`. This differs from the calculation of orbital density, where the default value for the orbital part is to use both, real and imaginary part.

4.1.3 Spin density

For the spin density, computed with the assignment **compute**=rho-spin, cf. Table 2 on page 12, both spin parts are needed. Thus, the spin density will be not computed if a particular spin part is chosen.

4.1.4 ELI

In case of ELI-D, computed with the assignment **compute**=eli-d <pair>, three results are possible, depending on the choice of the pair-spin component. Choosing the keyword 'alpha-alpha' respectively 'beta-beta' for the pair-spin yields the ELI-D for the corresponding same-spin electron pairs [13]. In each respective cases the electron density of corresponding spin will be sampled.

ELI-D for triplet coupled electron pairs [7] is computed with the keyword 'triplet-pair' for the pair-spin. In this case the charge of triplet-coupled electrons is sampled. Note that the value of ELI-D for triplet pairs is influenced by the total number of electrons of the system, see Ref. [7].

In ELI-D for triplet-coupled electrons both spin parts are included. The formula used in DGrid is derived for restricted basis only. Bear in mind, that for unrestricted basis (like for spin-polarized system computed with ADF), the formula is not the correct one (as in this case the expectation value of S^2 yields an improper value, contaminated by higher multiplets). In case of unrestricted basis ELI-D for triplet-coupled electrons will be evaluated, **however**, the density and the pair-volume function of triplet-coupled electrons is computed according to Eqs. 5 and 37 of Ref [7], which is just a pure prop to be able to show an ELI-D value.

The other variant of ELI, namely ELI-q (cf. Ref. [7]), is derived from the space partitioning based on fixed charge. If both spins should be included it must be specified that the partitioning is based on the fixed charge of singlet-coupled electrons, i.e., with the command '**compute**=ELI-q singlet', in which case the singlet-coupled electron pairs are sampled. It is still possible to compute the variant ELIA, which is derived from space partitioning based on fixed product of α-spin and β-spin charge, cf. Ref. [15].

4.1.5 ELF

The assignments '**compute**=ELF alpha', respectively '**compute**=ELF beta', result in the calculation of ELF according the formula of Becke and Edgecombe [8]. If both spin channels are chosen (i.e., '**compute**=ELF'), then ELF for the total density will be computed according the 'spin-polarized' formula, cf. Table 2 as well as Eq. 9 in Ref. [10]. The assignment '**compute**=ELF-cs' yields for the total density the ELF computed according the 'closed-shell' formula, cf. Table 2 as well as Eq. 7 in Ref. [10]. The 'spin-polarized' formulation follows the idea that ELF is based on kinetic energy densities, whereas the 'closed-shell' formula is more related to the electron pair densities and can be, in certain sense, rationalized from ELI for triplet-coupled electron pairs [7].

With the assignment '**compute**=ELF-Kirshnitz' (with or without the spin descriptor) the ELF variant of Tsirelson [11] is computed. It is based on kinetic energy density approximated by the Kirshnitz formula.

In case of LOL, computed with the assignment '**compute**=LOL', single spin channel should be chosen to comply with the definition of Schmider and Becke [16].

4.2 STR format

The STR format is used to support structure information for visualization. The data are written in sections into a file starting with the following header sections, termed 'UNITS' and 'SCALE':

```
UNITS
BOHR            1

SCALE
ATOMIC_RADIUS   0.5
BOND_DIAMETER   0.5
BOND_MIN        0.1
BOND_MAX        5.0
PATH_DIAMETER   0.5
```

The data can be evaluated by chosen visualization tool (and are actually used by the module written for the program Amira/Avizo). This header is followed by one or more sections called:

- ATOMS ↦ spheres
- CONNECTIONS ↦ cylinders
- UNIT_CELL ↦ unit cell
- POLYHEDRA ↦ polyhedra and polygons
- PATHS ↦ cylinders connected to form a path

Each section is followed by a separate part defining the colors used in the corresponding section (always using the same scheme, e.g., the 'ATOMS' section is followed by the 'ATOM_BUTTONS' part, the 'CONNECTIONS' section is followed by the 'CONNECTION_BUTTONS' part, etc.).

The last keyword in the file must be the string 'END'.

ATOMS

In this section the data for spheres representing atoms, maxima, saddle points, etc. are given. Following is an example for the C_3H_6 molecule, written out by the Avizo visualization program (similar data file can be created with the command '`dgrid` *basisfile* `str`', see the section 3):

```
ATOMS
:atom      radius      x         y         z      vis
:-------------------------------------------------
 C_1        0.451    0.0000    0.0000    1.6428   1
 C_2        0.451    0.0000    1.4227   -0.8214   1
 C_3        0.451   -0.0000   -1.4227   -0.8214   1
 H_4        0.095    1.6980   -0.0000    2.7503   1
 H_5        0.095   -1.6980    0.0000    2.7503   1
 H_6        0.095    1.6980    2.3818   -1.3751   1
 H_7        0.095   -1.6980    2.3818   -1.3751   1
 H_8        0.095    1.6980   -2.3818   -1.3751   1
 H_9        0.095   -1.6980   -2.3818   -1.3751   1

ATOM_BUTTONS
:atom      r       g       b
:-----------------------------
 C      0.3300  0.3300  0.3300
 H      0.6700  0.6700  0.6700
```

The (atomic) symbol is a 5 character string possibly followed by '_<num>' digit differentiating between atoms with identical symbols. It is followed by the radius and the cartesian position of the sphere. The last number shows the visibility (0 - not rendered by the visualization tool; 1 - visible). The 'ATOM_BUTTONS' section gives the RGB colors for each atomic symbol. If the 'ATOM_BUTTONS' part is not present then the visualization tool must choose its own default values.

CONNECTIONS

In this section the data for the cylinders (usually representing the bonds) are given:

```
CONNECTIONS
:connection   diameter      x1       y1       z1         x2       y2       z2     vis c
:-------------------------------------------------------------------------------------
 C-C_1-2        0.100     0.0000   0.0000   1.6428     0.0000   1.4227  -0.8214   1  1
 C-C_1-3        0.100     0.0000   0.0000   1.6428    -0.0000  -1.4227  -0.8214   1  1
 C-H_1-4        0.100     0.0000   0.0000   1.6428     1.6980  -0.0000   2.7503   1  1
 C-H_1-5        0.100     0.0000   0.0000   1.6428    -1.6980   0.0000   2.7503   1  1
 C-C_2-3        0.100     0.0000   1.4227  -0.8214    -0.0000  -1.4227  -0.8214   1  1
 C-H_2-6        0.100     0.0000   1.4227  -0.8214     1.6980   2.3818  -1.3751   1  1
 C-H_2-7        0.100     0.0000   1.4227  -0.8214    -1.6980   2.3818  -1.3751   1  1
 C-H_3-8        0.100    -0.0000  -1.4227  -0.8214     1.6980  -2.3818  -1.3751   1  1
 C-H_3-9        0.100    -0.0000  -1.4227  -0.8214    -1.6980  -2.3818  -1.3751   1  1

CONNECTION_BUTTONS
:col      r       g       b
:----------------------------
  1:   1.0000  0.4900  0.4900
```

The connection symbol shows which atoms are connected (can be also some chosen label).

It is followed by the diameter as well as the positions of the start and end points of the cylinder (bond). The last numbers show the visibility (0 - not visible in the visualization tool; 1 - visible) and color number. The 'CONNECTION_BUTTONS' part gives the corresponding the RGB colors.

UNIT_CELL

In this section the data for the unit cell are given (from which the unit cell can be constructed):

```
UNIT_CELL
:   diameter       x1        y1        z1          x2        y2        z2    col
:-------------------------------------------------------------------------------
A:   0.100       0.0000    0.0000    0.0000       1.0000    0.0000    0.0000   1
B:   0.100       0.0000    0.0000    0.0000       0.0000    1.0000    0.0000   1
C:   0.100       0.0000    0.0000    0.0000       0.0000    0.0000    1.0000   1

UNIT_CELL_BUTTONS
:col      r       g       b
:---------------------------
  1:   0.8600  0.0900  0.0900
```

For the three vectors A, B, C the diameter as well as the positions of the start and end points of the cylinders are given. The last number shows the color of the cylinder, given by the RGB values in the 'UNIT_CELL_BUTTONS' section.

POLYHEDRA

In this section the data for polyhedra build up from polygons (faces) are given :

```
POLYHEDRA
:           faces     vis col     spec      shin     transp
:-----------------------------------------------------------
Polyhedron    1         1   1     0.0000    0.0000    0.4681

  FACE vertices:   3
      0.0000    0.0000    1.6428
      0.0000    1.4227   -0.8214
     -0.0000   -1.4227   -0.8214

POLYHEDRA_BUTTONS
:col      r       g       b
:---------------------------
  1:   1.0000  0.4941  0.4941
```

For each polyhedron the number of faces is given, followed by the visibility and color number (see the 'POLYHEDRA_BUTTONS' section) together with the data concerning the shinnines and transparency of the faces. Each face has its own subsection giving the number of vertices and the corresponding cartesian coordinates (the face in the example is a triangle formed by the three carbon atoms of the C_3H_6 molecule).

PATHS

In this section the data for each trajectory running from the saddle point to the corresponding attractor are given (cf. the file *C3H6_HF_6-31G.g03.rho_r.bsn.graph.str* discussed on page 39):

```
PATHS
:        name     points   vis col  diameter
:-----------------------------------------
Path     c6-a6     15      1   1      0.02

:         x         y         z
:---------------------------------
        1.0551  2.009599 -1.160225
        1.0717  2.019214 -1.165777
                ...
        1.6869  2.375301 -1.371348

:        name     points   vis col  diameter
:-----------------------------------------
Path     c6-a7      23      1   1      0.02

:         x         y         z
:---------------------------------
        1.0551  2.009599 -1.160225
        1.0384  1.999983 -1.154674
                ...
        0.0121  1.429095 -0.825092

PATH_BUTTONS
:col      r       g       b
:---------------------------
  1:   1.0000  0.5000  0.0000
```

For each path a separate subsection is written. It contains the name of the path and the number of points, which connected by cylinders form the path. The label 'c6-a6' stands for the path from the critical point number 6 (the saddle point between the ρ-basins H6 and C2, cf. the table on page 39) to the attractor number 6 (internal assignment). This is followed by the visibility, color, and diameter of the path. Then the cartesian coordinates of each point of the path are given, 1 per line. The 'PATH_BUTTONS' section gives

the RGB colors of the paths. Fig. 4.1 shows the molecular graph (from the electron density) together with the interconnection graph (from ELI-D) for the C_3H_6 molecule. It can be seen that each carbon atom is connected by four interaction lines based on the electron density gradient (bond paths) with two hydrogens and two carbons. In case of ELI-D the carbons have again four interconnection lines, now connection the carbon with two hydrogen attractors and two ELI-D C-C bond attractors. Note that in the ELI-D interconnection graph each hydrogen attractor has four interconnection lines, connecting it with the closest (another) hydrogen attractor, the carbon core attractor and two lines reaches the ELI-D C-C bond attractors. Each ELI-D C-C bond attractor is connected to eight surrounding attractors.

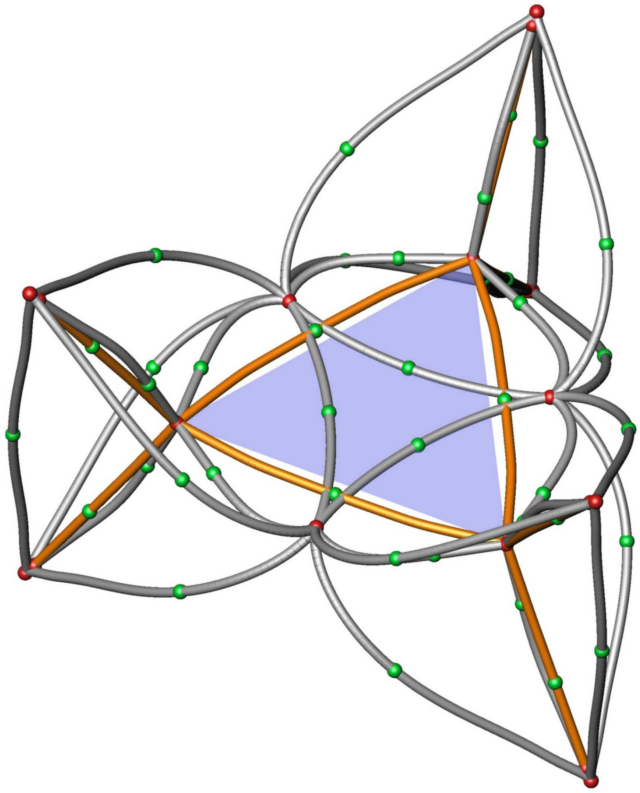

Figure 4.1: Molecular graph (brown) and ELI-D interconnection graph (gray) for C_3H_6. Red spheres: attractors; green spheres: saddle points. The carbon atoms are at the edges of the blue triangle.

END

Do not forget the string 'END' at the end of file. And this is also the end of the user's guide.

Bibliography

[1] Cube format. http://astronomy.swin.edu.au/~pbourke/geomformats/cube/

[2] Frisch MJ, Trucks GW, Schlegel HB, Scuseria GE, Robb MA, Cheeseman JR, Zakrzewski VG, Montgomery JA Jr., Stratmann RE, Burant JC, S. Dapprich S, Millam JM, Daniels AD, Kudin KN, Strain MC, Farkas O, Tomasi J, Barone V, Cossi M, Cammi R, Mennucci B, Pomelli C, Adamo C, Clifford S, Ochterski J, Petersson GA, Ayala PY, Cui Q, Morokuma K, Malick DK, Rabuck AD, Raghavachari K, Foresman JB, Cioslowski J, Ortiz JV, Baboul AG, Stefanov BB, Liu G, Liashenko A, Piskorz P, Komaromi I, Gomperts R, Martin RL, Fox DJ, Keith T, Al-Laham MA, Peng CY, Nanayakkara A, Gonzalez C, Challacombe M, Gill PMW, Johnson B, Chen W, Wong MW, Andres JL, Gonzalez C, Head-Gordon M, Replogle ES, Pople JA (2004) Gaussian 03, Revision C.02, Gaussian Inc., Wallingford CT

[3] MOLPRO, a package of ab initio programs designed by Werner H-J and Knowles PJ. Amos RD, Bernhardsson A, Berning A, Celani P, Cooper DL, Deegan MJO, Dobbyn AJ, Eckert F, Hampel C, Hetzer G, Knowles PJ, Korona T, Lindh R, Lloyd AW, McNicholas SJ, Manby FR, Meyer W, Mura ME, Nicklass A, Palmieri P, Pitzer R, Rauhut G, Schütz M, Schumann U, Stoll H, Stone AJ, Tarroni R, Thorsteinsson T, Werner H-J

[4] MOLCAS, Andersson K, Barysz M, Bernhardsson A, Blomberg MRA, Cooper DL, Fülscher MP, de Graaf C, Hess BA, Karlström G, Lindh R, Malmqvist P-Å, Nakajima T, Neogrády P, Olsen J, Roos BO, Schimmelpfennig B, Schütz M, Seijo L, Serrano-Andr'es L, Siegbahn PEM, Stålring J, Thorsteinsson T, Veryazov V, Widmark P-O, Lund University, Sweden

[5] ADF (Density Functional program ADF2007.01). Baerends EJ, Ellis DE, Ros P (1973) Chem Phys 2: 41; Versluis L, Ziegler T (1988) J Chem Phys 88: 322; te Velde G, Baerends EJ (1992) J Comput Phys 99: 84; Fonseca Guerre C, Snijders JG, G. Velde G, Baerends EJ (1998) Theor Chem Acc 99: 391

[6] Grace. http://plasma-gate.weizmann.ac.il/Grace/

[7] Kohout M, Wagner FR, Grin Y (2008) Theor Chem Acc 119: 413

[8] Becke AD, Edgecombe KE (1990) J Chem Phys 92: 5397

[9] Savin A, Jepsen O, Flad J, Andersen OK, Preuss H, von Schnering HG (1992) Angew Chem 104: 186

[10] Kohout M, Savin A (1996) Int J Quantum Chem 60: 875

[11] Tsirelson V, Stash A (2002) Chem Phys Lett 351: 142

[12] Kohout M (2004) Int J Quantum Chem 97: 651

[13] Kohout M, Pernal K, Wagner FR, Grin Y (2004) Theor Chem Acc 112: 453

[14] Kohout M, Wagner FR, Grin Y (2006) Int J Quantum Chem 106: 1499

[15] Kohout M, Pernal K, Wagner FR, Grin Y (2005) Theor Chem Acc 113: 287

[16] Schmider HL, Becke AD (2000) J Mol Struct (Theochem) 527: 51

[17] Hunter G (1986) Int J Quantum Chem 29: 197

[18] Wagner FR, Bezugly V, Kohout M, Grin Yu (2007) Chem Eur J 13: 5724

[19] G.Schaftenaar and J.H. Noordik (2000) Molden: a pre- and post-processing program for molecular and electronic structures. J Comput-Aided Mol Design 14: 123

[20] Bader RFW, Gatti C (1998) Chem Phys Lett 287: 233

[21] Gatti C, Cargnoni F, Bertini L (2002) J Comput Chem 24: 422

[22] Gatti C, Lasi D (2007) Faraday Discuss 135: 55

[23] Raub S, Jansen G (2001) Theor Chem Acc 106: 223

[24] Bader RFW (1990) Atoms in Molecules: A Quantum Theory. Oxford University Press, Oxford

[25] Silvi B, Savin A (1994) Nature 371: 683

[26] Jepsen O, Andersen OK (2000) The Stuttgart TB-LMTO-ASA Program, Version 4.7. Max-Planck-Institut für Festkörperforschung, Stuttgart

[27] Cremer D, Kraka E (1984) Croat Chem Acta 57: 1259