Miroslav Kohout

DGrid 4.6

– User’s Guide –

Springer

Preface

For molecules DGrid computes properties out of the information about the wavefunction as given in special basis file. All the values, gradients, and Hessian matrices are computed analytically (as far as possible). This was not the case for solid state results. There, only the precomputed grid were evaluated numerically. DGrid 4.6 is now able to access the data from the full potential solid state code Elk. Dr. Alexey I. Baranov succeeded in the creation of an interface to this (open source) program. This not only enables DGrid to determine the critical points and interconnection lines for solid state compound, but also to compute the overlap integrals and thus to evaluate the fluctuation and the delocalization indices.

The calculation of fine mesh grids, especially for large molecules respectively for solid state are timely very expensive (spending even weeks or months with the computation of the data). As the number of processors within a single machine is rapidly increasing, parallelization of code is the logical step. However, for ”true” parallelization lot of changes in the DGrid code were necessary. To start with I gave chosen another (simpler) strategy. The desired grid is cut into slices (the number of which corresponds to the number of processors available to the user) and for each slice a DGrid control file is generated by a script (generously donated by my colleague Dr. Frank R. Wagner). The script submits each job to separate processor. After all the grid slices are finished the total grid is completed by a DGrid utility. We have experienced almost linear scaling up to 40 processors.

The conversion routines of DGrid were extended so that the wavefunctions from the quantum chemical program GAMESS can now be evaluated.

The availability of the overlap integrals allowed the calculation of Fermi orbitals. The conversion of the basis file to the one containing Fermi orbitals (with or without the isopycnic transformation) enables the calculation and visualization of the orbitals.

The source code and the description of the implementation and usage of the interfaces to the visualization programs Avizo and OpenDX are again part of the DGrid package. Dr. Alexey I. Baranov, who is responsible for the interface to OpenDX, wrote modules to a third (free) visualization program, namely ParaView.

Rudimentary help for certain parts of DGrid is given by calling:

```
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
-----------------------------------------------
```

The manual was written with Latex using the SVMono style. The permission of Springer is acknowledged.

Radebeul, March 2011 *M. K.*

Contents

List of Tables

List of Figures

Acronyms

AO	atomic orbital
CP	critical point
DAFH	domain-averaged Fermi-hole
DFT	density functional theory
ELI	electron localizability indicator
ELI-D	electron localizability indicator based on D-restricted partitioning
HF	Hartree-Fock
ICL	interconnection line
MO	molecular orbital

Chapter 1
DGrid 4.6

1.1 Disclaimer

The program DGrid is distributed ‘as is’ without warranties of any kind. No responsibility of any kind is assumed from the use of DGrid. Use the following citation for the program DGrid:

M. Kohout, DGrid, version 4.6, Radebeul, 2011

1.2 Distribution

DGrid source code is distributed free of charge. The code remains in the ownership of the author. The copyright must not be changed when adapting the code. DGrid can be freely redistributed. The current version as well as all updates with changes and corrections implemented later can be downloaded from the web page:

http://www.cpfs.mpg.de/∼kohout/dgrid.html

The User’s guide as well as the source code, precompiled executables, modules for the visualization programs, and examples can be downloaded separately.

1.3 Installation

Executable versions of DGrid 4.6 are available over the DGrid web page for Linux (openSUSE 11.3) and Windows. Otherwise, the source code must be compiled. The program DGrid is written in C++, i.e., an appropriate compiler must be available. The installation procedures for a Unix system respectively for Windows PC (using MS Visual Studio) are described.

1.3.1 Windows

In this section the compilation of DGrid with Visual Studio 2008 is described. Note that for Windows there is no support for the evaluation of wavefunctions supplied by the solid state Elk package [6] (except when C++ and Fortran-90 compilers as well as *make* build system can be used, like MinGW environment).

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 file *dgrid-4.6.zip* 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-4.6*. The directory *dgrid-4.6* contains the project file *dgrid-4.6.vcproj* and the directory *src* with all the source code files.

Start the Visual Studio 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-4.6.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-4.6.exe   controlfilename
```

The output goes to the console, unless the *output* file name is given in the *control* file, cf. the Sec. 2.3, ‘Control file’, page 11).

1.3.2 Linux

For Linux there are two distributions of the DGrid – with and without the support for the solid state Elk package [6]. Each of the distributions has a separate makefile. Of course, the DGrid version with the Elk interface can be used for molecular wavefunctions as well.

1.3.2.1 DGrid without Elk support

Unpack the distribution tar file dgrid-4.6.tar.gz using the commands:

```
gunzip   dgrid-4.6.tar.gz
tar xf   dgbas-4.6.tar
```

This creates new directory *dgrid-4.6*. Change to the directory *dgrid-4.6*. You should find there the script file `dgrid_para`, the directory *src* with all the source code files and a makefile. View the makefile – you may have to change the name of the compiler (for instance the Intel compiler icpc) and the compiler options. Create DGrid with the command:

```
make
```

The executable `dgrid-4.6` will be written to the directory $HOME/bin (this can be changed in the makefile). For convenience, link this executable with the name dgrid:

```
ln -s   dgrid-4.6   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 Sec. 2.3, 'Control file', page 11).

1.3.2.2 DGrid with Elk support

Elk is a high-accuracy solid-state all-electron full-potential linearized augmented planewave (FP-LAPW) code distributed under GPL (http://elk.sourceforge.net). DGrid can be build with the Elk support, i.e., Elk calculation results can be directly processed with DGrid. This support makes it possible to use many features available in DGrid like evaluation of properties, topological analysis including search for critical points, and pair density analysis [13, 14].

Technically, Elk support is provided by a library (LIBELK) of Elk routines, libraries for BLAS, LAPACK and FFT routines used by Elk and DGrid with Elk support and certain additional components of DGrid necessary to work with Elk. A

Fortran-90 compiler (e.g. GNU gfortran) is necessary to build LIBELK. Currently, Elk version 1.2.20 is supported.

Described below is the installation under Linux with GNU compilers (g++ and gfortran). Similar procedure can be used for other systems where C++ and Fortran-90 compilers are also available and *make* build system can be used (like MinGW environment for Microsoft Windows).

Unpack the distribution tar file dgrid-4.6e.tar.gz using the commands:

```
gunzip   dgrid-4.6e.tar.gz
tar xf   dgbas-4.6e.tar
```

This creates new directory *dgrid-4.6e*.

1. Download and build Elk 1.2.20 from Elk website http://elk.sourceforge.net. Consult Elk user guide when necessary. This will build Elk itself together with BLAS (*blas.a*), LAPACK (*lapack.a*) and FFT (*fftlib.a*) libraries.
2. Build LIBELK. To do this, find the file Makefile.libelk in the *dgrid-4.6e* and copy it to the *src* folder of the Elk distribution (NOT the *src* folder within *dgrid-4.6e*). Change to it and do:

```
make -f Makefile.libelk
```

   This will create a file *libelk.a* of LIBELK.
3. Change back to the directory *dgrid-4.6e*. Besides the script file `dgrid_para` there should be the directory *src* with all the source code files and a makefile. Edit the makefile and specify correct path (`ELK_PATH`) to the *src* folder of Elk where the libraries are located.

   Remark 1.1. When Elk is built with libxc support, uncomment the line
   `LIBXC = $(ELK_PATH)/libxc.a`
4. Build DGrid with Elk support with the command:

```
make
```

The executable `dgrid-4.6e` will be written to the directory $HOME/bin (this can be changed in the makefile). For convenience, link this executable with the name dgrid:

```
ln -s   dgrid-4.6e   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 Sec. 2.3, ‘Control file’, page 11).

Remark 1.2. To evaluate Elk results the input file ‘elk.in’ (call the DGrid conversion routine always with this file name) needs to be converted. The name for the converted file (in wide sense corresponding to a *basis* file) should be given as a second argument (e.g. ‘Al.mte’). The generated file (MTE-file) contains the expansions of charge density and kinetic energy density together with their derivatives. This allows fast evaluation of many properties so that the ‘.mte’ file is sufficient to run the tasks. However, some tasks, like the evaluation of delocalization indices or projected properties require an access to the wavefunctions which are not saved in the ‘.mte’ file. In this case additional files from the Elk calculation should be available.

Chapter 2
Property Grids

2.1 Introduction

DGrid is a program for the generation of property values on an equidistant grid (respectively at a single point only) and the analysis of those gridded fields, inclusive the basin search. 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 [22], MOLPRO [47], MOLCAS [7], TURBOMOLE [5], GAMESS [3], and ADF [1], 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 information is 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.

2.2 Basis file

Each of the quantum chemical packages GAUSSIAN, MOLPRO, MOLCAS, TURBOMOLE, GAMESS, and ADF, respectively, 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.

For the GAMESS all the information is recovered form the output file (the precision of some output variables must be increased in few GAMESS routines). GAUSSIAN calculations supply the formatted *name*.wfn file as well as the binary check-point file *name*.chk, where the latter needs first 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 information 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 (it is anyway a good idea to check the proper normalization by charge integration of the Xe atom).

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 orbital occupations using the keyword **occupation** in the *control* file (bear in mind that the inclusion of virtual orbitals for partial ELI-D is not consistent with its definition).

Find in the *examples* directory (downloaded from the DGrid web page) the formatted GAUSSIAN03 checkpoint file C3H3NO_HF_6-31G.fchk for the oxazole molecule (geometry from Ref. [42] re-optimized with GAUSSIAN03). Using the command:

```
dgrid   C3H3NO_HF_6-31G.fchk
```

yields the DGrid *basis* file file C3H3NO_HF_6-31G.g03 with 51 orbitals (6-31G basis set). Of course, only 18 orbitals are (doubly) occupied. Converting data from other QM packages yields basis files with corresponding extender, cf. Table 2.1.

Table 2.1 Basis file name extender subject to QM package used

Program descr.	Source		Extender
ADF	TAPE21	$\mapsto$	adf
ADF	TAPE21 (loc. orbitals)	$\mapsto$	adf_loc
GAMESS	output file	$\mapsto$	gms
GAUSSIAN94	Checkpoint file	$\mapsto$	g94
GAUSSIAN98	Checkpoint file	$\mapsto$	g98
GAUSSIAN03	Checkpoint file	$\mapsto$	g03
GAUSSIAN09	Checkpoint file	$\mapsto$	g09
GAUSSIAN	WFN file	$\mapsto$	gwf
MOLPRO	molden file	$\mapsto$	mp
MOLCAS	molden file	$\mapsto$	mc
MOLDEN	molden file	$\mapsto$	md
CLEMENTI-ROETTI	Tables [17]	$\mapsto$	CR

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 used for 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 '3 pz_a' is the p_z AO on the atom number 3 (nitrogen), 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 C3H3NO_HF_6-31G.g98 from the above example reads:

```
BASISFILE   created by DGrid version 4.6   23-02-2011

program: GAUSSIAN03        date: Wed Feb 23 17:18:09 2011

:title
::C3H3NO HF 6-31G

:atom  No.            x              y              z           charge
:----------------------------------------------------------------------
  O     1:      -2.06183559     0.57482826     0.00000000        8.00
  C     2:       0.00000000     2.12106164    -0.00000000        6.00
  N     3:       2.08694733     0.89554761    -0.00000000        7.00
  C     4:       1.41345421    -1.67482606    -0.00000000        6.00
  C     5:      -1.11013465    -1.85958662     0.00000000        6.00
  H     6:      -0.30771125     4.10249907    -0.00000000        1.00
  H     7:       2.80485596    -3.12185095    -0.00000000        1.00
  H     8:      -2.43100862    -3.36800124     0.00000000        1.00
:----------------------------------------------------------------------
```

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

Energy=   -244.50273 Hartree              Electrons=   18 alpha  +    18 beta


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

: atom  No.       type            exponents and coefficients
:--------------------------------------------------------------------------------
   O     1          s    exp:     5.48467166e+03    8.25234946e+02    1.88046958e+02
                         coe:     1.83107443e-03    1.39501722e-02    6.84450781e-02
         ...
   H     6 -   8    s    exp:     1.61277759e-01
:--------------------------------------------------------------------------------

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


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

:orb       energy     coefficients
:------------------------------------------------------------------------
   1     -20.65207    0.9958683260      ...    -0.0005791664    0.0000000000
                         ...            ...             ...       ...
...
  51       2.16106    0.0133842933      ...     0.0285089560    0.0000000000
                         ...            ...             ...       ...
:------------------------------------------------------------------------
:end of MO data

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

:-----------------------------------------------------------
:  #  symmetry         orb          ALPHA           BETA
:-----------------------------------------------------------
   1  NONE              1      1.00000000000    1.00000000000
      ...
  51  NONE             51      0.00000000000    0.00000000000
:-----------------------------------------------------------
```

Notice that the molecular orbital number 51 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 18 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.

2.3 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
:------------------------------------------------|
::C3H3NO   HF    6-31G
:------------------------------------------------|

:KEYWORDS
:-------------------------------------------------
 basis=C3H3NO_HF_6-31G.g03

 output=C3H3NO_run

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

 mesh=0.05   4.0

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 *C3H3NO_run.dg*.

Remark 2.1. Better choice is the assignment '**output=.**' in which case the name of the *output* file will be generated from the *basis* file name (i.e., the file name *C3H3NO_HF_6-31G.g03.dg* would be generated from the *basis* file *C3H3NO_HF_6-31G.g03.g98*). This allows for clearer correspondence between the *basis* file and all the output files.

Omitting the **output** keyword will send the output to the console. In the *output* file some setup information and the timing as well as the calculation progress is shown. For a single point calculation (see the keyword '**point**=' in Sec. 2.6) 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 (case insensitive):

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

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 Tables 2.2 to 2.4 the possible descriptors are given together with the strings (extenders) that will be appended to the *result* file name (exemplary for the property *prop* and the generic name *filename* in coordinate space, i.e., with '_r' appended).

Table 2.2 Grid file name extender subject to property type

Property type		File name[a]
	$\mapsto$	*filename.prop_r*
gradient-mag	$\mapsto$	*filename.***grad-mag**_*prop_r*
gradient-vec	$\mapsto$	*filename.***grad**_*prop_r*
ln-derivative	$\mapsto$	*filename.***ln_deriv**_*prop_r*
hessian	$\mapsto$	*filename.***hess**_*prop_r*
laplacian	$\mapsto$	*filename.***lap**_*prop_r*
relative-laplacian	$\mapsto$	*filename.***rel-lap**_*prop_r*

[a] Exemplary for the generic name *filename* and property *prop* in real space

If <type> is omitted, cf. the first line in Tab. 2.2 then the property value will be computed (and the *result* file named *filename.prop_r*). The <type> descriptors can be listed with the command '`dgrid -t`'.

Table 2.3 Grid file name extender subject to property spin

Property spin		File name[a]
alpha	$\mapsto$	*filename.prop_r_***a**
beta	$\mapsto$	*filename.prop_r_***b**
both	$\mapsto$	*filename.prop_r*
singlet	$\mapsto$	*filename.prop_r_***s**
triplet	$\mapsto$	*filename.prop_r_***t**

[a] Exemplary for the generic name *filename* and property *prop* in real space

If <spin> is omitted, then total spin will be used for 1-particle property (i.e., the default value is `both`, cf. Tab. 2.3). For exceptions from this rule, as well as the choice in case of 2-particle property see remark 2.2 below. The <spin> descriptors can be listed with the command '`dgrid -s`'.

Table 2.4 Grid file name extender subject to property pair spin

Property pair spin		File name[a]
alpha-alpha	$\mapsto$	*filename.prop_r_***aa**
beta-beta	$\mapsto$	*filename.prop_r_***bb**
alpha-beta	$\mapsto$	*filename.prop_r_***ab**
beta-alpha	$\mapsto$	*filename.prop_r_***ba**
singlet-pair	$\mapsto$	*filename.prop_r_***sg**
triplet-pair	$\mapsto$	*filename.prop_r_***tr**
spinless	$\mapsto$	*filename.prop_r*

[a] Exemplary for the generic name *filename* and property *prop* in real space

If <pair> is omitted, then the default value is either `unknown` (in which case the file name is created as for `spinless`, cf. Tab. 2.4) or the value is spin dependent, see remark 2.2 below. The <pair> descriptors can be listed with the command '`dgrid -s`'.

Remark 2.2. The descriptors can be omitted when calling the **compute** command. Then the default values for <type>, <spin>, <pair> will apply. In case of <type> the property value will be computed. For <spin> and <pair> the default values depend on the property. For a 1-particle property (like the electron density, kinetic energy density, local source, etc.) the <spin> descriptor is set to `both`, except for '**compute**=phi' and overlap integrals, in which case the majority spin `alpha` is chosen. For 2-particle property the <pair> descriptor is set in accordance with the <spin> descriptor, i.e., `alpha-alpha` pair spin if `alpha` spin is given, etc. (if <spin> given, otherwise `spinless` is used). If only <pair> descriptor is given, then the <spin> descriptor is set accordingly (for the <pair> descriptor `spinless` the <spin> descriptor `both` is used).

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 2.5, 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 *C3H3NO_HF_6-31G.g03.rho_r* and *C3H3NO_HF_6-31G.g03.lap_rho_r*, respectively (because of the assignment **basis**=*C3H3NO_HF_6-31G.g03*).

As already mentioned, the generic file name can be changed using the keyword **result**. For instance, with the assignment '**result**=*C3H3NO*' the two *property* files would be named *C3H3NO.rho_r* and *C3H3NO.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. Sec. 2.6).

Per default all properties are computed in the real (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**=0.05 4.0' defines the grid region. The keyword '**mesh**=' is followed by the mesh-size, i.e., the 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 0.05 bohr and a border of 4.0 bohr will be created around the C_3H_3NO molecule. The program works as follows (cf. Fig. 2.2 on page 37). 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 (however, as given by the example, not necessarily parallel with the coordinate axes – to achieve this add the keyword **parallel**). The 'initial box' is enlarged in all three directions by the border of 4.0 bohr (the grid region dimensions being multiple of the mesh-size of 0.05 bohr). This results in a grid of $266 \times 309 \times 161$ points (with the dimensions of 13.25 bohr $\times$ 15.4 bohr $\times$ 8.0 bohr, cf. the lower diagram in Fig. 2.2).

Another possibility to define the grid region is shown in the *control* file *elid.inp* (see the *examples* directory):

```
:TITLE
:-----------------------------------------------|
::C3H3NO   HF    6-31G
:-----------------------------------------------|

:KEYWORDS
:------------------------------------------------
 basis=C3H3NO_HF_6-31G.g03

 output=.


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

```
 GRID_DEFINITION:   vectors
:----------------------------
:             X     Y     Z
:----------------------------
 origin:    -6.0  -7.0  -4.0
                                INTERVALS:
:---------------------------------------
 i-vector:  12.0   0.0   0.0     120
 j-vector:   0.0  14.0   0.0     140
 k-vector:   0.0   0.0   8.0      80

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 (parallel to the Cartesian axes) of $121 \times 141 \times 81$ points in a box of 12 bohr $\times$ 14 bohr $\times$ 8 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 *C3H3NO_HF_6-31G.g03.dg* as determined by the *basis* file name. The '**compute**=ELI-D triplet-pair' assignment yields a data grid written to the *result* file named *C3H3NO_HF_6-31G.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) [33]. 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**. For each chosen orbital the data must be written in single line. 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 17 and 18 of the symmetry 'NONE' (GAUSSIAN writes the orbitals without symmetry), both double occupied, are selected for the calculation. The occupation block must be closed with the **occupation_end** command on a separate line.

Remark 2.3. To compute the amplitude for single orbital, (i.e., not the orbital density) use the assignment '**compute**=phi', for instance, '**compute**=phi sym=NONE alpha 2' for the amplitude of the second alpha MO of the symmetry NONE. Unlike the DGrid 4.5 version, there can be more than one '**compute**=phi' assignment in the control file.

```
:TITLE
:-----------------------------------------------------|
::C3H3NO   HF    6-31G
:-----------------------------------------------------|

:KEYWORDS
:-----------------------------------------------------
 basis=C3H3NO_HF_6-31G.g03

 output=.


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

 mesh=0.1  4.0

 USERS_OCCUPATION_INPUT:  occupation
:-------------------------------------
: SYM      #        ALPHA  BETA
:-------------------------------
 NONE     17        1.0    1.0
 NONE     18        1.0    1.0

 occupation_end

END
```

Table 2.5 summarizes the available properties (the conditional properties are given in Table 2.6 on page 18). For almost all properties the derivatives can be computed using the <type> descriptors for the *compute* keyword, cf. page 12. Observe that ELI-D is written in Table 2.5 as $\Upsilon_D = \rho\tilde{V}_D$, i.e., as the electron density ρ multiplied by the pair-volume function $\tilde{V}_D$. As described in Ref. [28, 45] ELI-D can be exactly decomposed into orbital contributions, the same way as the electron density is given by the sum of 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 (number 17 and 18) 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 the contributions of all orbitals recovers the total ELI-D.

Remark 2.4. pELI-D contributions are possible only for occupied orbitals. DGrid does not compute contributions of virtual orbitals.

Table 2.5 Available properties

Keyword	Property	Description	Ref.	File ext.
compute=ELF	$\left[1+\chi^2\right]^{-1}$	ELF: 'spin-polarized'	[15, 39, 31]	elf
compute=ELF-cs	$\left[1+\chi_{cs}^2\right]^{-1}$	ELF: 'closed-shell'	[15, 39, 31]	elf_cs
compute=ELF-Kirzhnits	$\left[1+\chi_{kirzh}^2\right]^{-1}$	ELF: Tsirelson	[44]	elf_kirzh
compute=ELI-D	$\rho\,\tilde{V}_D$	ELI-D	[27, 29]	elid
compute=ELI-q	$\rho_2^{(s)}\,\tilde{V}_q^2$	ELI-q	[33]	eliq
compute=ELIA	$\rho_2^{\alpha\beta}/(\rho_\alpha\rho_\beta)$	ELIA	[30]	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	[41]	lol
compute=LS	$-\frac{1}{4\pi}\frac{\nabla^2\rho(r)}{\lvert r-r\prime\rvert}$	local source	[10, 24]	ls
compute=OEP	$\frac{\nabla^2\sqrt{\rho}}{2\sqrt{\rho}}$	one-electron potential	[25]	oep
compute=pair-volume-function	$\tilde{V}_D$	pair-volume function	[28, 45]	pair_vol
compute=pELI-D	$\rho_{orb}\tilde{V}_D$	orbital resolved ELI-D	[28, 45]	p_elid
compute=phi	ϕ_i	orbitals		phi
compute=charge-volume-function	$\tilde{V}_q$	charge-volume function	[33]	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

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

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

$\tilde{V}_D = [12/g^\sigma]^{3/8}$

$\tilde{V}_q = 1/\rho^{(s)};$ $\rho^{(s)}$ 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 non-interacting system

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

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

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

$\tau^{LSDA} = 2^{2/3}c_F\,\rho^{5/3}$ originally defined for single spin channel, i.e., ρ_σ [41]

Table 2.6 Conditional properties

Keyword	Property	Description	File ext.
compute=cond-pair-density	$P_{cond}(1,2)$	conditional pair density	pd_cond
compute=DAFH	$f_{\mathrm{av}}^{\sigma\sigma}(1)$	domain-averaged Fermi-hole	dafh
compute=hole-density	$\rho_{ex}^{\sigma}(1)$	hole density	hole_density
compute=pair-density	$\rho_2(1,2)$	pair density	pd

$\rho_2^{\sigma\sigma}(1,2) = \frac{1}{2}\sum_{i<j}^{\sigma}\sum_{k<l}^{\sigma} P_{ij,kl} \left|\phi_i(1)\phi_j(2)\right| \left|\phi_k^*(1)\phi_l^*(2)\right|$
$f_{\mathrm{av}}^{\sigma\sigma}(1) = \rho(1)\int_\Omega \rho(2)\mathrm{d}2 \; - \; 2\int_\Omega \rho_2^{\sigma\sigma}(1,2)\mathrm{d}2$
$P_{cond}^{\sigma\sigma}(1,2) = \rho_2^{\sigma\sigma}(1,2)/\rho_\sigma(2)$
$\rho_2^{\alpha\beta}(1,2) = \frac{1}{2}\sum_{i,j}^{\alpha}\sum_{k,l}^{\beta} P_{ij,kl} \left|\phi_i(1)\phi_j(2)\right| \left|\phi_k^*(1)\phi_l^*(2)\right|$
$P_{cond}^{\sigma\sigma'}(1,2) = \rho_2^{\sigma\sigma'}(1,2)/\rho_{\sigma'}(2)$
$\rho_2(1,2) = \rho_2^{\alpha\alpha}(1,2) + \rho_2^{\beta\beta}(1,2) + \rho_2^{\alpha\beta}(1,2) + \rho_2^{\beta\alpha}(1,2) = \rho_2^{(s)}(1,2) + \rho_2^{(t)}(1,2)$
$P_{cond}(1,2) = \rho_2(1,2)/\rho(2)$
$\rho_{ex}^{\sigma}(1) = 2P_{cond}^{\sigma\sigma}(1,2) - \rho_\sigma(1)$

Remark 2.5. 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 will NOT add up to total ELI-D.

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. to 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):

```
:TITLE
:-----------------------------------------------|
::C3H3NO   HF    6-31G
:-----------------------------------------------|

:KEYWORDS
:------------------------------------------------
 basis=C3H3NO_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  4.0

  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 2.6. Then, fixing the position of an electron yields a 3 dimensional property. The calculation using the *control* file *cond.input* yields the electron density (together with the gradient and Hessian) at the position [0.0, 0.0, 0.0] (for the electron density itself the reference point has no meaning). It can be seen 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.

Similar background follows the calculation of the domain-averaged Fermi-hole, DAFH [34], in which case one of the pair density coordinates is confined to given region Ω (for instance a basin). To compute DAHF overlap integrals over the confinement region are needed, cf. Sec. 3.7 and the example on page 76.

The last information in the *control* file is the keyword **end** (can be omitted if no data lines are following). Data lines following the keyword **end** are not read. All keywords are read case insensitive. In many cases the keywords can be written in any order and often there is no need to write each keyword in a separate 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. All available keywords are described in detail in Sec. 2.6 ‘Keywords’).

2.4 Property files

Each property is written to separate file distinguished by a descriptor according to the **compute** assignment, see Tables 2.5 and 2.6. 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 4.6   24-02-2011

:job was executed on:         Thu Feb 24 15:06:22 2011

basis_from_program:      GAUSSIAN03              basis=C3H3NO_HF_6-31G.g03

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

```
: property computed from orbitals
:-----------------------------------------------
: symmetry         No.          occupation
:-----------------------------------------------
  NONE             17     1.00000000   1.00000000
  NONE             18     1.00000000   1.00000000
:------------------------------------------------

::C3H3NO   HF    6-31G

:atom         x              y              z         vis
:-------------------------------------------------------
 O      -2.06183559     0.57482826     0.00000000       1
 C       0.00000000     2.12106164    -0.00000000       1
 N       2.08694733     0.89554761    -0.00000000       1
 C       1.41345421    -1.67482606    -0.00000000       1
 C      -1.11013465    -1.85958662     0.00000000       1
 H      -0.30771125     4.10249907    -0.00000000       1
 H       2.80485596    -3.12185095    -0.00000000       1
 H      -2.43100862    -3.36800124     0.00000000       1
:-------------------------------------------------------

Energy=    -244.50273 Hartree                    Electrons=   18 alpha  +    18 beta

:             lattice vectors
:-------------------------------------
:          a             b             c
:-------------------------------------------
x:    1.00000000    0.00000000    0.00000000
y:    0.00000000    1.00000000    0.00000000
z:    0.00000000    0.00000000    1.00000000
:-------------------------------------------

:          origin              v_i           v_j           v_k
:---------------------------------------------------------------
x:      -6.24317899       13.27862203   -0.87280762    0.00000000
y:      -7.65976747        0.75378840   15.37524656    0.00000000
z:      -4.00000000        0.00000000    0.00000000    8.00000000
:---------------------------------------------------------------
           points:             134           155            81

: start of data
:---------------------------------------------------------------------------
 3.2267796e-11   4.1766301e-11   5.3714085e-11   6.8637172e-11   8.7145274e-11
...
```

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.

Remark 2.6. In certain cases DGrid needs the access to the *basis* file. Thus, do not rename this file (otherwise 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 written into 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.

Remark 2.7. In *grid* files generated with old DGrid versions, 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’. Starting from version 4.4 Dgrid does not known the property ‘ELI’. 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 (the visibility flag is used by some visualization tools). 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 below 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 gradient vector of the property *prop* is 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:      -4.5000        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
...
```

2.5 Parallel calculation

The calculation of a property grid can be extremely time consuming task that can take for weeks or even months. As nowadays usually more than one processor is available in a computer it would be convenient to compute the grids in parallel. Although DGrid 4.6 itself is not ready yet for automatic parallelization (except for the calculation of overlap integrals) the script `dgrid_para`, included in the DGrid package, supplies a work around for such parallel runs. The script `dgrid_para` creates *control* files where job computes a part of the grid (a slice). The operating system distributes the jobs among the processors. As the final step all slices are merged together into the whole grid. Similar procedure can be applied to the search for the critical points.

2.5.1 Grid slices

The *property* grid is defined by three vectors i, j, k and the number of points $n_i \times n_j \times n_k$ along each direction, cf. the previous Section. Around the grid box there is a 2 points wide border. In DGrid the loops for the grid values generation run first along the i vector, followed by j and last is k. With the *control* command:

divide_grid=<div> <slc>

the whole grid is divided into <div> slices along the k direction and the slice number <slc> is computed. Thus the slice covers $(n_i + 4) \times (n_j + 4)$ points in the i, j directions (i.e., inclusive the grid border) and $(n_k + 4)/div$ points in the k direction (some slices can include 1 point more). The resulting slice of the *property* grid (with '.slice-<slc>' appended to the file name) will contain the same header as the full grid. However, before the array of grid values starts, there is an additional line with the information about the grid slice. For instance, the calculation of the ELI-D grid for the oxazole with the *control* file:

```
:TITLE
:-------------------------------------------------------------------|
::C3H3NO   HF    6-31G
:-------------------------------------------------------------------|

:KEYWORDS
:--------------------------------------------------------------------
 basis=C3H3NO_HF_6-31G.g03

 output=.

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

 divide_grid=4  1

 GRID_DEFINITION:   vectors
:             X     Y     Z
:----------------------------
 origin:    -6.0  -7.0  -4.0
```

```
:                              INTERVALS
:-----------------------------------------
 i-vector:  12.0   0.0   0.0     120
 j-vector:   0.0  14.0   0.0     140
 k-vector:   0.0   0.0   8.0      80

END
```

yields the ELI-D grid named *C3H3NO_HF_6-31G.g03.elid_r_t_tr.slice-1*, which is the first slice of the whole grid (defined by $121 \times 141 \times 81$ points) after the division into 4 slices. In the *property* file the (unchanged)grid definition is followed by a line with slice information:

```
:          origin              v_i           v_j           v_k
:-----------------------------------------------------------------
x:      -6.00000000       12.00000000    0.00000000    0.00000000
y:      -7.00000000        0.00000000   14.00000000    0.00000000
z:      -4.00000000        0.00000000    0.00000000    8.00000000
:-----------------------------------------------------------------
           points:             121           141            81

slices=  4      nr.   1               -2 -   19           border=2

: start of data
:------------------------------------------------------------------------------
 9.2985810e-02   9.5796936e-02   9.8560838e-02   1.0126902e-01   1.0391299e-01
 ...
```

The info line shows that this is the slice number 1 (of 4 slices). Because of the 2 points wide border the slice starts in the *k* direction at the point number (-2) and ends at point number 19 (i.e., in total 22 points in *k* direction). The grid itself, i.e., without the border points starts in the *k* direction at point number 0 and ends at point number 80. The slices number 2, 3, and 4 contain 21 points each in the *k* direction (20-40, 41-61, 62-82).

After each of the 4 slices is finished the whole grid is build by merging the slices together using the DGrid utility:

```
dgrid   C3H3NO_HF_6-31G.g03.elid_r_t_tr.slice-1   complete
```

The utility must be called with the file for the slice number 1. Also, all the remaining slice files must be present in the active directory. The resulting grid will be saved in the file *C3H3NO_HF_6-31G.g03.elid_r_t_tr*.

2.5.2 *Dgrid_para script*

The submission of the grid-slice jobs can be accomplished automatically using the script `dgrid_para`, written by Dr. F. R. Wagner and available in the DGrid package. The script is called the same way like an ordinary DGrid calculation:

```
dgrid_para    controlfilename
```

where the *control* file includes the **divide_grid** command (see the *control* file example at page 22; however, now the number of slices <div> as well as the slice number <nr> are not utilized). The script shows the following output:

```
=================================================================
|                         DGrid_Para                            |
|                                                               |
| Purpose      : start multiple slice jobs of sequential DGRID  |
|                                                               |
|   !!!requires  DGRID version 4.6 BUILD Feb 17 2011 or later!!!|
|                                                               |
|    for tasks : GRID calculation                               |
|                                                               |
|    for tasks : ATTRACTORS, MINIMA, or SADDLES determination   |
|                                                               |
|                                               FRW, 17.02.11 |
=================================================================
```

```
Initial No. of planes read  : 81
-->  4 border planes added => 85

Total No. of planes along z = 85
Give  No. of slices along z :
```

The script informs about the number of planes found along the *k* direction.

Remark 2.8. If the grid is defined in the *control* file by 3 vectors, then the line with the third vector (i.e., the *k*-vector) MUST start with the string ‘k-vector:’. Otherwise the `dgrid_para` script will not recongnize the script definition.

`dgrid_para` asks how many slices are desired (into how many parts the grid should be divided). Having 4 processors on my computer let us choose 4 slices (as already mentioned, the script is not utilizing the number of slices given by the **divide_grid** command – one could choose any other number as well). After typing ‘4’ the script proceeds and outputs:

```
           Variant (1):
              slices  =  4
==================================
------------  TIMING -------------
     3  jobs   with   21  planes
     1  jobs   with   22  planes
======>    Total Timing        =  22
==================================
-----------  CPU usage -----------
======>   Total # of JOBS      =  4
==================================

 Select Variant: 0(def)=other 1=variant1
```

With ‘0’ another division scheme could be tried. Typing ‘1’ confirms the choice of the ‘Variant (1)’ shown above:

```
--------------------------------------------------------------
Selected: ===> Run DGRID with nslices = 4 on nproc = 4
--------------------------------------------------------------


Trying to find and identify DGRID executable ....

REMIND: !!!requires  DGRID version 4.6 BUILD Feb 17 2011 or later!!!

/home/kohout/bin/dgrid
                             *********************************
                             *         D G R I D    4.6      *
                             *                               *
                             *   14-03-2011  (c) M. Kohout   *
                             *                               *
                             * Build  Mar 14 2011  11:26:38  *
                             *********************************

Press ENTER to START multiple DGrid jobs ...
```

The script shows which DGrid version (and build) will be used for the execution. Pressing ‘enter’ creates and submits *control* files for multiple jobs, which is confirmed by the message:

```
Submit:  dgrid elid_slice.inp-1 .
Submit:  dgrid elid_slice.inp-2 .
Submit:  dgrid elid_slice.inp-3 .
Submit:  dgrid elid_slice.inp-4 .
```

All 4 *control* files can be found in the active directory. After all jobs are finished the 4 grid-slice files, with names terminated by ‘.slice-1’ till ‘.slice-4’, are located in the active directory (respectively at the position given by the command **result** in the *control* file). There are also 4 files named *elid_slice.inp-1.out*, etc., containing either only the error messages (usually empty) or the DGrid output when the command **output**= was commented out. If the command **output**= was used then 4 DGrid output files (ending with ‘.dg’) will be created.

Remark 2.9. It is the better choice to write the DGrid output (the ‘.dg’ files) to the output files of the `dgrid_para` script (the ‘.out’ files), i.e., do not use the**output**= command in the *control* file. Then there will be a better correspondence between the numbering and the output files and the slices. Thus, ‘*.out-1’ shows the progress for the slice number 1, etc., whereas in case of the ‘.dg’ files it would be the file ‘*.dg’ and ‘*.dg_1’ for the slice number 2, etc.

Remark 2.10. The number of slices chosen with the `dgrid_para` script need not to coincide with the number of available processors. Often it is even better to use higher number of slices (i.e., processor tasks) than processors. Especially for large grids around complex molecules there is lot of ‘empty space’ where the primitive functions are almost zero valued, thus very fast evaluated. In this case some of the slices can be done much earlier than others, leaving idle processors without tasks. Then it is favorable to occupy the processors with more than one task.

To create the complete grid proceed with the DGrid `complete` utility as described in previous Section.

2.6 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 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 Sec. 3.8.

The keywords are given either as variables, like e.g. ‘**compute**=’, or as standalone expressions, like ‘**end**’. Some of them are followed by 1 or more numbers, e.g., ‘**mesh**=0.05 4.0’. The parsing routine discriminates between floats and integer numbers. Thus, the command ‘**point**= 0.5 1.0 3’ would not be valid, because ‘3’ is an integer number, whereas three floats are expected for the coordinates.

Often more keywords can be written in single line. But it is better to put every keyword in separate line, because sometimes additional data are assumed per default. Some data even need to be written in rigorous sequence, e.g., the definition of a grid using the **vectors** keyword or the choice of occupied orbitals. Table 2.7 summarizes all the keywords available for the grid and single point evaluation, respectively.

2.6.1 Basis

The keyword **basis** is followed by the name of the *basis* file (described in Sec. 2.2). 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 into 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* that will be written into the local directory where the calculation is per-

Table 2.7 Keywords and parameters available for the grid calculation

Keyword	Description	Value
basis=	basis set	basis set filename
close-atom_distance=	distance to nucleus	float number (default 0.1)
component=	write vector component	x, y, z, xx, yy, zz, xy, xz, yz
compute=	choose property	properties given in Table 2.5
contributions=	orbital contributions	property (rho, ELI-D), x, y, z
cp=	search for critical point	(max, saddle, ring), x, y, z (float)
curv_decomp=	decomposition of ELI-D curvature	3 Cartesian coordinates (float)
divide_grid=	grid division into slices	slices, slice number (2 integers)
end	end of input	
energy_window=	for projected property (Elk)	emin, emax (float numbers)
ev_projection=	gradient projection	property, x, y, z (float)
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
overlap_matrix=	specify overlap matrix file	overlap matrix, basin number
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

Default values are given in `Typewriter` font

formed. The same applies to the *output* file (where the file */home/test/molecule.adf.dg* would be generated).

2.6.2 Close-atom_distance

The keyword **close-atom_distance** is followed by the distance <dist>. At the distance <dist> from the nucleus the search for a critical point stops. The default value is 0.01 bohr.

close-atom_distance=<dist>

The reason is to avoid discontinuities or very large gradients close to the nucleus.

2.6.3 Component

The keyword **component** is followed by the desired component <prop_i> of a property-vector array.

component=<prop_i>

If a gradient or Hessian of a property is computed, all values of the vector are written in 1 line in the *output* file, cf. page 21. With the above assignment a single component can be chosen. The allowed values for the <prop_i> descriptor are X, Y, Z for the gradient and XX, YY, ZZ, XY, XZ, YZ for the Hessian.

2.6.4 Compute

The keyword **compute** is followed by the desired property <prop>. All available properties are given in Tables 2.5 and 2.6 on pages 17 and 18, respectively.

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

The descriptors <type>, <spin>, and <pair>, in detail given in the Sec. 2.3 on page 12, 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).

The input for the calculation of an orbital deviates somewhat from the above assignment.

compute=phi **sym**=<symnam> <num> <spin> <part>

The descriptors can be written in any order. Here, <num> is the (integer) number of the chosen orbital in the <symnam> (default is 'NONE'). <spin> can be one of 'alpha' or 'beta'. With the descriptor <part> the real (descriptor 'real', which is the default) or imaginary ('imag') part of the orbital can be computed (the imaginary part is usually present for orbitals in momentum space). Other possibility is given for orbital values expressed in polar form, where the descriptors 'amplitude' and 'phase' computes the corresponding parts. The orbital symmetry and number as well as the chosen spin and part form the extender of the *result* file. Unlike the DGrid 4.5 version, there can be more than one '**compute**=phi' assignment in the control file.

Figure 2.1 shows the orbitals number 17 and 18 (which is the HOMO) for the oxazole molecule in different representations. As can be seen from the usual coordinate space representations in diagrams 2.1a the MO 17 is roughly the nitrogen lone pair and MO 18 is a π orbital (the red colored isosurfaces have negative isovalue). Using the *basis* file *C3H3NO_HF_6-31G.g03* the data for the orbitals were computed with the assignment '**compute**=phi 17 alpha real' and '**compute**=phi 18 alpha real' in the *control* file (the symmetry 'NONE' was set by default). The diagrams 2.1b-d show isosurfaces for the orbitals computed in the momentum space representation. Each orbital is generally given by a complex function:

$$\phi(\mathbf{p}) \;=\; \varphi(\mathbf{p}) \,+\, \mathrm{i}\,\chi(\mathbf{p}) \tag{2.1}$$

with the real and imaginary parts φ and χ, respectively. In the diagrams 2.1b the isosurfaces for the real parts of the corresponding orbitals are depicted (the gray colored isosurfaces have negative isovalue; observe the inversion symmetry). The diagrams 2.1c show the imaginary of the orbitals.

Both parts of an complex orbital can be unified into single diagram using the polar form of the orbital:

$$\phi(\mathbf{p}) \;=\; \vartheta(\mathbf{p})\, \mathrm{e}^{\mathrm{i}\,\omega(\mathbf{p})} \qquad \vartheta = \sqrt{\varphi^2+\chi^2} \qquad \omega = \mathrm{sgn}(\chi)\ \arccos(\varphi/\vartheta) \tag{2.2}$$

with the amplitude ϑ and phase ω. The diagrams 2.1d show the isosurface for the amplitudes of orbitals number 17 and 18. Each isosurface is colored by the phase (the range for the color scale is $\pm\pi$). Interestingly, the isosurface for amplitude of the orbital 17 resembles the isosurface for the imaginary part of the MO, cf. Fig. 2.1c, whereas the amplitude for the orbital number 18 is close the isosurface for the corresponding real part of the MO 18. This is also confirmed by the coloring of the amplitude isosurfaces (deep blue and white correspond to the phase $\pm\pi$ which stands for the real part (similarly, green color means phase $\omega \approx 0$, i.e. again the real part). In case of MO 17 the overall coloring of the amplitude isosurface is around $\pm\pi/2$, i.e., phase value for the imaginary part.

2.6.5 Contributions

The keyword **contributions** is followed by the property <prop> together with the descriptors <type>, <spin> and <pair> and the coordinates for the position.

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

For the property only one of 'rho', 'rho laplacian', or 'ELI-D', respectively, is allowed by now. DGrid computes the property and its orbital contributions at the position given by the Cartesian coordinates [*x y z*] (3 float numbers).

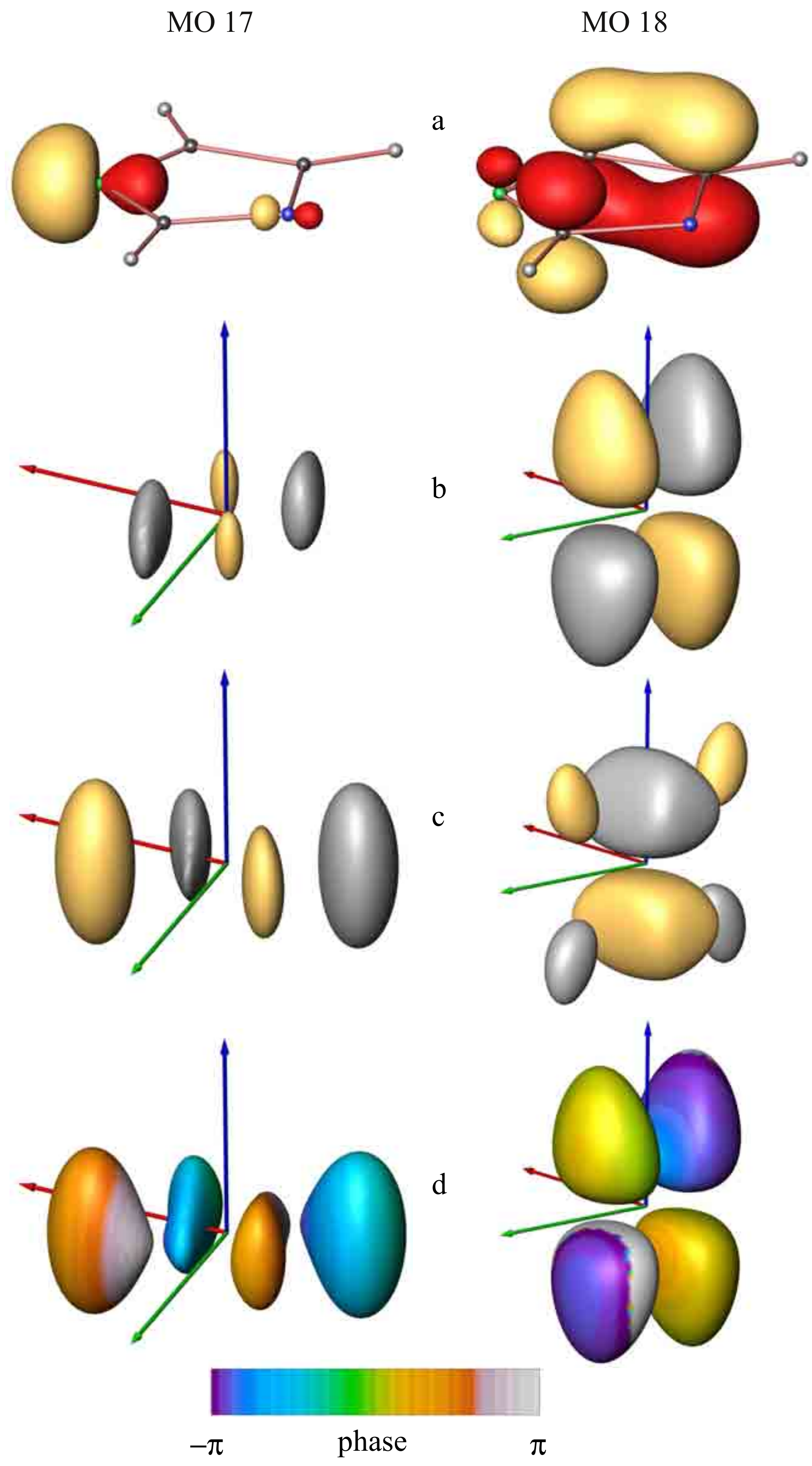

Fig. 2.1 Coordinate (**a**) and momentum space (**b-d**) representations of the orbitals number 17 and 18 (HOMO) for oxazole C_3H_3NO (C black, H gray, N green, O blue). **a**: ±0.16 and ±0.10 isosurfaces for MO 17 and 18, respectively; **b**: real parts of the MOs (±0.20 isosurfaces); **c**: imaginary parts of the MOs (±0.26 and ±0.05 isosurfaces); **d**: amplitudes for the MOs (±0.26 and ±0.20 isosurfaces colored by the phase)

For example, using the *control* file *elid_contrib.inp* (from the *examples* directory) yields the *output* file *C3H3NO_elid_contrib.dg*. Below is the result of the analysis:

```
----------------------------------------------------------------------------
contributions at point           [  1.050510,    1.500850,    0.000000]
----------------------------------------------------------------------------

                     spin :        alpha
                     pair :        alpha-alpha
                                 ----------
                 ELI-D[r] :       1.73735

--------------------
      MO         %
--------------------
   7    7      37.6
  14   14      18.6
   9    9      11.0
   8    8       9.3
  17   17       6.7
   6    6       5.9
  10   10       4.4
  15   15       4.3
  11   11       1.3
--------------------
               99.0
```

The orbital contributions are given at the position of the ELI-D bond attractor between the nitrogen and carbon (found using the keyword **cp**, see below). The analysis shows that the ELI-D value 1.73735, which is proportional to the population of α-spin electrons needed to form a fixed fraction of an $\alpha\alpha$-spin pair [33, 45], is a sum of mainly six contributions. The two highest contributions originate from the canonical orbitals $\phi_7\phi_7^*$ (37.6%) and $\phi_{14}\phi_{14}^*$ (18.6%). The orbital indexes correspond to the numbering in the *basis* file *C3H3NO_HF_6-31G.g03* set in the *control* file. In the above case the sum of the orbital contributions does not yield 100% because contributions smaller than 1% are not listed.

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.

2.6.6 Cp

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

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

The search for the critical point starts from the position given by the Cartesian coordinates [*x y z*] (3 float numbers). The property for which the critical point will be searched must be given in a **compute** assignment. The position of the critical

Table 2.8 Descriptors for critical point search

<crit>		description	signature
min	$\mapsto$	minimum	(3, +3)
max	$\mapsto$	attractor	(3, -3)
saddle	$\mapsto$	saddle point	(3, -1)
ring	$\mapsto$	ring critical point	(3, +1)

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 Sec. A.6 in the Appendix A). In the *control* file *elid_cp.inp* (from the *examples* directory) for the search of an ELI-D attractor the starting point [1.5, 2.0, 0.0] was chosen. The resulting attractor (C-C bond descriptor) position is [1.05051, 1.50085, 0.00000], see the *output* file *C3H3NO_elid_cp.dg*. This position was used in the above example for the ELI-D contributions.

2.6.7 *Curv_decomp*

The keyword **curv_decomp** is followed by the property <prop> together with the descriptors <type>, <spin> and <pair> and the coordinates for the position.

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

Only ELI-D is allowed at the time for <prop>. At the position given by the Cartesian coordinates [*x y z*] (3 float numbers) the value, gradient, Laplacian and Hessian, together with the curvature eigenvalues and eigenvectors are computed and printed to the console. Along the principal ELI-D curvatures the ELI-D Laplacian is decomposed into the electron density and pair-volume function dependent terms [46]. The decomposed values are printed:

```
----------------------------------------------------------------------------
ELI-D decomposition at point          [  1.050510,    1.500850,    0.000000]
----------------------------------------------------------------------------

                     spin :        alpha
                     pair :        alpha-alpha
                                 ----------
                 ELI-D[r] :       1.73735

                 gradient :     9.086e-06
               components :  [ -4.216e-06, -8.049e-06,    0.00000 ]
```

```
                Laplacian :      -7.69085

                  Hessian :  |   -4.47877,    1.27618,    0.00000 |
                             |    1.27618,   -2.79458,    0.00000 |
                             |    0.00000,    0.00000,   -0.41750 |

                                      Eigenvectors               curvature
                             -------------------------------    -----------
                  1. ev :  [  0.88056, -0.47394,  0.00000 ]      -5.16565
                  2. ev :  [  0.47394,  0.88056,  0.00000 ]      -2.10770
                  3. ev :  [  0.00000,  0.00000,  1.00000 ]      -0.41750
                                 x         y         z
----------------------------------------------------------------------------


Data projected onto the ELI-D eigenvectors
----------------------------------------------------------------------------

             Y''/Y           rho''/rho             V''/V      2rho'/rho*V'/V
----------------------------------------------------------------------------
 11 |      -2.97329            1.79670          -4.52209            -0.24790
 22 |      -1.21317           -2.26056           1.05507            -0.00768
 33 |      -0.24031           -1.95647           1.71617             0.00000
----------------------------------------------------------------------------
sum |      -4.42677           -2.42033          -1.75085            -0.25558
----------------------------------------------------------------------------
```

Here the 11, 22, and 33 rows contain data for curvatures along the corresponding principal curvatures, e.g., 11 for the curvature along the first eingenvector (‘1. ev’ in the list above). ELI-D is given by the symbol Y (meaning Υ_D) and the pair-volume function by the symbol V.

2.6.8 *Divide_grid*

The *property* grid defined by $n_i \times n_j \times n_k$ points in each direction can be divided into slices along the k direction (in DGrid the loop over the grid points runs first over the i direction, followed by j and k), cf. Sec. 2.5. The grid division into slices includes the grid border as well. The calculation of a grid slice is accomplished with the keyword **divide_grid** that followed by the number <div> of slices and the slice number <slc>:

divide_grid=<div>   <slc>

For each slice a separate *control* file is needed (thus, <div> jobs must be submitted to get all the slices for the whole grid). To the *property* file for each slice the string ‘.slice-<slc>’ is appended to discriminate between the slices. Having all slices computed the whole grid can be build by merging the slices together calling the ‘complete’ utility:

```
dgrid   gridname.slice-1   complete
```

The resulting complete *property* grid *gridname* includes all the points.

2.6.9 End

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

2.6.10 Energy_window

Used only for projected properties computed from Elk [6] '.mte' data. The keyword **energy_window** is followed by 2 float numbers for the energy range:

```
energy_window=<emin>   <emax>
```

All k-states within the energy window from <emin> to <emax> will be included into the calculation of the projected property. The energy values for the window must be given in a.u. relative to the Fermi level. Note that only the valence states in terms of Elk can be projected, not the core states.

2.6.11 Ev_projection

The keyword **ev_projection** is followed by the property <prop> together with the descriptors <type>, <spin> and <pair> and the coordinates for the position.

```
ev_projection=<prop>   <type>  <spin>  <pair>   <x>  <y>  <z>
```

For the property <prop> the value, gradient, Laplacian and Hessian, together with the curvature eigenvalues and eigenvectors are computed at the position given by the Cartesian coordinates [*x y z*] (3 float numbers) and printed to the console. The gradients of properties given by **compute** assignments are projected onto the principal <prop> curvatures. Using the example *evproj.inp*:

```
 ----------------------------------------------------------------------------
 eigenvectors at point           [  0.773035,    1.624593,    0.000000]
 ----------------------------------------------------------------------------

                     spin :        alpha
                     pair :         --
                                 ----------
                   rho[r] :       0.18699

                 gradient :     1.631e-07
               components :  [  8.461e-08,  1.394e-07,    0.00000 ]

                Laplacian :      -0.61740

                  Hessian :  |    0.03440,   -0.23782,    0.00000 |
                             |   -0.23782,   -0.30100,    0.00000 |
                             |    0.00000,    0.00000,   -0.35080 |

                                       Eigenvectors                curvature
                             --------------------------------    -----------
                    1. ev :  [  0.88777, -0.46028,  0.00000 ]       0.15770
                    2. ev :  [  0.46028,  0.88777,  0.00000 ]      -0.42430
                    3. ev :  [  0.00000,  0.00000,  1.00000 ]      -0.35080
                                  x         y         z
 ----------------------------------------------------------------------------
```

shows the value, the Laplacian, and the principal curvatures of α-spin electron density of C_3H_3NO at the position [0.773035, 1.624593, 0.000000] which is the bond critical point between the nitrogen and carbon. This is followed by the projection of the ELI-D gradient in the direction of the mentioned eigenvectors:

```
 gradient projections onto the eigenvectors:
 ----------------------------------------------------------------------------

                     spin :        alpha
                     pair :        alpha-alpha
                                 ----------
                 ELI-D[r] :       1.43897

                 gradient :       2.07429
               components :  [    1.76076,   -1.09655,    0.00000 ]

    component projections :  [    2.06787,   -0.16304,    0.00000 ]

                Laplacian :      -4.93182

                  Hessian :  |   -3.96054,    3.13798,    0.00000 |
                             |    3.13798,   -2.17949,    0.00000 |
                             |    0.00000,    0.00000,    1.20821 |


          rotated Hessian :  |   -6.14773,    1.08057,    0.00000 |
                             |    1.08057,    0.00770,    0.00000 |
                             |    0.00000,    0.00000,    1.20821 |
 ----------------------------------------------------------------------------
```

First the ELI-D value and the gradient components along the Cartesian axes are given, followed by the gradient components projected onto the principal density curvatures (of course, the gradient vector remains unchanged). It can be seen that the ELI-D gradient vector is oriented mainly along one of the principal density curvatures. Similar projection is performed for the ELI-D Hessian components.

2.6.12 Format

The keyword **format** is followed by the descriptor <fmt> (case insensitive) for the format of the *result* file (not used for the input grid files, for instance with the assignments '**property**=' or '**integrate**=', in which case the grid format is detected automatically):

format=<fmt>

Table 2.9 Format descriptors

<fmt>		Format
cube	$\mapsto$	CUBE [2]
dgrid	$\mapsto$	DGrid *result* file
grace	$\mapsto$	Grace [4]
lmto	$\mapsto$	TB-LMTO-ASA [26]

Remark 2.11. 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.

2.6.13 Mesh

The computed grid region can be conveniently defined with the keyword **mesh**. It is followed by the mesh-size, i.e., the distance between neighboring grid points. Optionally, a border around the molecule can be given. To orient the grid along the Cartesian axes append the keyword `parallel`.

mesh=<size> <border> `parallel`

The mesh-size must be given by 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 (cf. example on page 11) where the assignment '**mesh**=0.05 4.0' is used. In the example a grid with the distance between neighboring points of 0.05 bohr and a border of 4.0 bohr is created around the C_3H_3NO molecule.

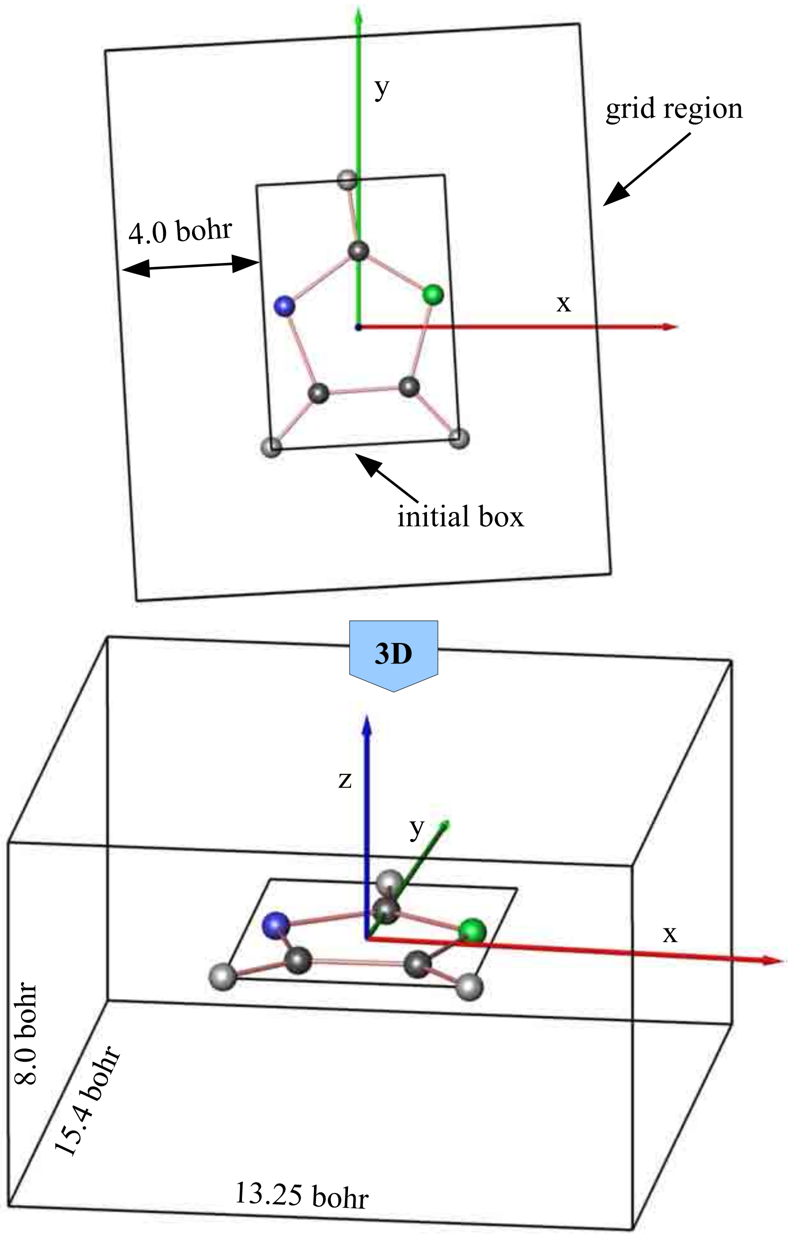

Fig. 2.2 Grid region for oxazole C_3H_3NO (C black, H gray, N green, O blue) created by the assignment ‘**mesh**=0.05 4.0’

The routine works as follows (cf. Fig. 2.2). 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. As shown in the example, where the box is slightly rotated in the xy plane, the box need not be parallel to the Cartesian axes. To force a layout with grid box parallel with the Cartesian axes the keyword `parallel` must be appended.

The ‘initial box’ is enlarged in all three directions by the border of 4.0 bohr (the grid region dimensions being multiple of the mesh-size of 0.05 bohr). This gives a grid with $266 \times 309 \times 161$ points (measuring 13.25 bohr $\times$ 15.4 bohr $\times$ 8.0 bohr, cf. the lower diagram in Fig. 2.2).

Remark 2.12. For the calculation of overlap integrals form Gauss-type orbitals, used for the fluctuation and delocalization indexes (see Sec. 3.7), it is necessary that the grid is oriented parallel to the Cartesian axes.

2.6.14 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 9, 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 occupation input is closed by the following line:

occupation_end

2.6.15 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. If the file is already present, then the string ‘_1’, ‘_2’, etc. will be appended to avoid overwriting of data.

2.6.16 Overlap_matrix

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

overlap_matrix=<name.sij> <basnr>

The overlap matrix for all molecular orbitals over the chosen basin is set up in DGrid. The data are needed for instance to compute the domain-averaged Fermi-hole or the Fermi orbitals.

2.6.17 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 separate **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 [9]. 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_3NO at the position [0.77304, 1.62459, 0.00000], the N-C bond critical point in electron density):

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

                -1.2348   -0.8647    0.2780    3.1104   -0.5867
------------------------------------------------------------
```

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

2.6.18 *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 default value for <step> is 0.02 bohr.

2.6.19 *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 2.6 on page 18. 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').

2.6.20 *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 2.5 and 2.6 on page 17 and 18, 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.

2.6.21 *Space*

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

```
space=<spc>
```

The default is the evaluation in position space representation. In case of momentum space representation the grid definition is using the momentum coordinates.

Remark 2.13. 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 given by the modulus of squared

Table 2.10 Space representation descriptors

<spc>		Representation
position	$\mapsto$	coordinate space [$x\ y\ z$]
momentum	$\mapsto$	momentum space [$p_x\ p_y\ p_z$]

orbitals has in momentum space similar meaning as in the real space. However, not the kinetic energy density, because this property is computed by DGrid from squared orbital gradients (of course, in momentum space the kinetic energy is proportional to the expectation value of p^2).

2.6.22 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.

2.6.23 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 by the Cartesian coordinates (float numbers):

```
  <x>   <y>   <z>
```

The strings ‘GRID_DEFINITION:’, ‘origin:’, ‘i-vector:’, etc. in the *control* file are not parsed due to the colon at the end of the strings – they used just for clarity (however, bear in mind the remark given below with respect to the usage of the `dgrid_para` script):

```
 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
```

Remark 2.14. The strings 'i-vector:', 'j-vector:' and 'k-vector:' are not parsed by the DGrid routine. However, in case of parallel grid calculations, using the script `dgrid_para` for slicing the grid, the mentioned string are necessary to find the grid definitions.

The next 3 lines after the origin 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 bohr 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 bohr in x direction and 4 bohr 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.

Chapter 3
Basin evaluation

3.1 Introduction

On an equidistant grid of property values regions surrounded by the surfaces of zero-flux of the property gradient (i.e., basins [9]) are determined. Alternatively, localization domains [43], 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 (a *basin* file).

If the *basis* file is present, the attractors, minima, saddle points and ring points are determined, together with the corresponding curvatures and Hessian eigenvectors. Additionally, the interconnection graphs can be produced (written in the STR format, see the Appendix A). The interconnection graphs are given by all the attractors and saddle points, respectively ring points and minima, connected by gradient field lines. In case of the electron density such diagram for the attractors and saddle points is called the molecular graph.

In case of Gauss-type basis the MO overlap integrals over the basins can be computed. Those overlap integrals are the prerequisite for the calculation of the fluctuation and delocalization indexes as well as Fermi orbitals.

The evaluation is performed by DGrid using a *control* file with keywords specific for the chosen analysis. The control file must comply with the rules given in the introduction to the DGrid program (cf. page 7).

3.2 Control file

For the *control* file any name can be chosen. The *control* file supplies all the information concerning the evaluation. The data must comply with the rules described in the introduction to chapter 1, page 7. A simple basin calculation can be performed with very few commands. 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
:------------------------------------------------------------|
::C3H3NO cropped density basins
:------------------------------------------------------------|

:KEYWORDS
:-------------------------------------------------------------
 property =C3H3NO_HF_6-31G.g03.rho_r
 crop     =C3H3NO_HF_6-31G.g03.rho_r       0.001
 integrate=C3H3NO_HF_6-31G.g03.rho_r

 output=.

end
```

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 Sec. 3.3 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 *property* grid file *C3H3NO_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 program determines the basins, i.e., regions enclosing the manifold of all property gradient trajectories terminating at given attractor.

Remark 3.1. Alternatively, a *basin* file (see below) with basin data from previous run can be read in instead of the *property* file. In this case the basins will not be recalculated (and written on disc again), thus saving time and disc space.

For a molecule the outer basins extend into infinity. Of course, in an actual calculation those basins extend just till the border of the chosen grid region. Obviously, it is hard to compare such basin volumes in some reasonable way. It seems to be convenient first to crop the region around the molecule by certain isodensity surface, a kind of envelope. The electron density isovalues used are 10^{-3}–10^{-5} bohr^{-3}. The cropping is invoked by the keyword **crop** followed by the name of the grid file utilized for the isosurface generation and the cropping value. With this, all grid points outside the cropped region will not be searched for basins. Thus, it avoids the time consuming basin search in regions of low gradients as well. The result for the above *control* file is shown in Fig. 3.1.

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 (and basins). Alternatively, the file from a DGrid refinement can be used.

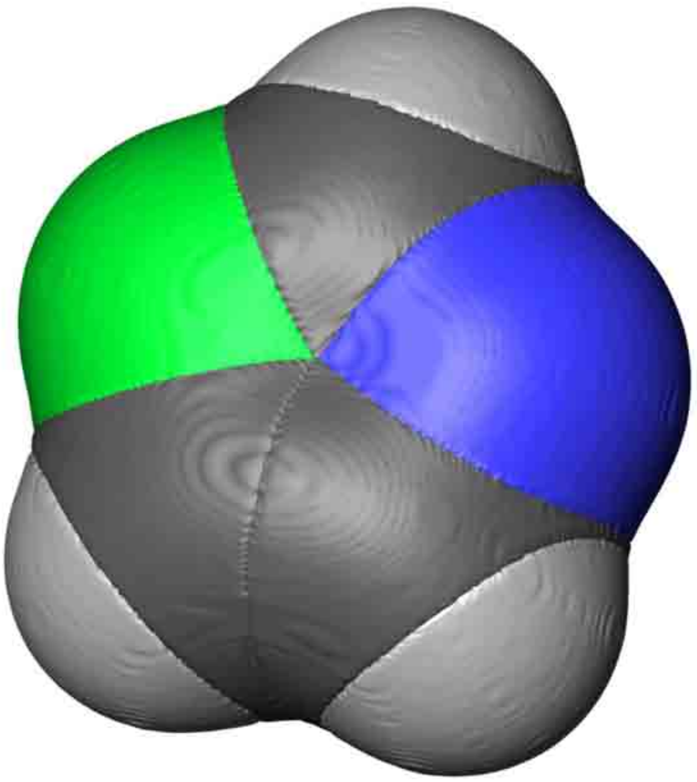

Fig. 3.1 Electron density basins for C_3H_3NO, cropped by 0.001 bohr^{-3} electron density isosurface. Blue: oxygen basin; green: nitrogen basin; black: carbon basins; gray: hydrogen basins

Remark 3.2. For electron density based on Gauss-type orbitals the overlap integrals over basins can be calculated (keyword **overlap**, cf. Sec. 3.7.1). In the subsequent evaluation of the fluctuation also the basin populations are computed analytically. The remaining error in the basin population is due to the form of the basin borders derived from a discrete grid (i.e., small cubes).

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 *C3H3NO_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 Sec. 3.8.

3.2.1 Attractors

The attractors, the prerequisite for a basin, are searched on the discrete grid. This often yields much more discrete grid maxima than actually given by the attractors of the vector field. There are two possibilities for the DGrid program to decide how to proceed, depending on the presence of the *basis* file.

Remark 3.3. The location of the *basis* file is given in the *property* file – bear in mind that the *basis* file must be present at the given location!

3.2.1.1 Case: 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 grid maxima on the grid (the list length depends on the additional parameters, see keyword **attractors** in Sec. 3.8). It supplies the following information:

```
                          ********************************************
                          *     D I S C R E T E     M A X I M A      *
                          ********************************************

-----------------------------------------------------------------------------------------------------------
grid-max basin    i   j   k       value     min.dist   value diff.       x       y       z      atom    dist
-----------------------------------------------------------------------------------------------------------
    1      1     92 160  80     232.6176     2.57       129.4017     -2.054   0.591   0.000     O1     0.02
    2      2    175 161  80     155.4799     2.41        52.2641      2.086   0.876   0.000     N3     0.02
    3      3    108 110  80     117.4257     1.94       116.9988     -1.114  -1.860   0.000     C5     0.00
    4      4    135 188  80     103.2158     1.96       102.7852      0.013   2.111   0.000     C2     0.02
    5      5    159 111  80     100.9578     1.95       100.5273      1.429  -1.665   0.000     C4     0.02
    6      6    131 227  80       0.4307     1.96      -102.7852     -0.297   4.046   0.000     H6     0.06
    7      7    184  81  80       0.4305     1.95      -100.5273      2.762  -3.092   0.000     H7     0.05
    8      8     81  82  80       0.4269     1.94      -116.9988     -2.382  -3.334   0.000     H8     0.06
-----------------------------------------------------------------------------------------------------------

Grid maxima written to the file C3H3NO_HF_6-31G.g03.rho_r.gridmax.str
```

The above result is for the *control* file *rho_basins.inp* including the keyword **attractors** and the *basis* file removed form the directory (thus unknown for DGrid). Grid-max is the running number of the grid maximum. The value *basin* indicates to which basin the grid maximum belongs.

Remark 3.4. After the trajectory search the basins will be renumbered in ascending order according to the volume size.

The indexes *i j k* are the grid coordinates of the grid maximum, whereas *x y z* are the corresponding Cartesian coordinates. *value* shows the function value (here the electron density) at the grid position. The grid maxima are given in descending order with respect to the function values. *min.dist* is the distance to the closest grid maximum (given in the same units as the grid parameters). *value diff.* is the value difference between the grid maximum and the closest grid maximum. *atom* shows the symbol of the closest atom (at the distance *dist*).

With the keyword **attractors** the file *C3H3NO_HF_6-31G.g03.rho_r.gridmax.str* will be created with the coordinates of the grid maxima written in the STR format (for visualization, cf. the Appendix A). In this case, no search for the basins is performed. For grids from solid state calculations, where the wave function information is not given, 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 grid maxima will be written to the output as well as into the file *gridmax.str*. The number of attractors written out can be changed appending additional parameters to the keyword **attractors**, cf. the Sec. 3.8.

Remark 3.5. To start the basin search the keyword **attractors** must not be active!

3.2.1.2 Case: using *basis* file

In this case all grid maxima on the discrete *property* grid will be tested by DGrid using the 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 grid maxima on previous page, now for a run with the *basis* file included):

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

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

--------------------------------------------------------------------------------------------------------
 Nr.  Basin       x          y          z         value     Laplacian                curvature
--------------------------------------------------------------------------------------------------------
  1     1      -2.0618     0.5748     0.0000     291.3237        -                      O1
  2     2       2.0869     0.8955    -0.0000     192.1030        -                      N3
  3     3      -1.1101    -1.8596     0.0000     118.3147        -                      C5
  4     4       0.0000     2.1211    -0.0000     118.3135        -                      C2
  5     5       1.4135    -1.6748    -0.0000     118.3130        -                      C4
  6     6      -0.3026     4.0665     0.0000       0.4320     -19.1957     -6.6122    -5.9827    -6.6008
  7     7       2.7802    -3.0966     0.0000       0.4315     -19.1472     -6.5837    -5.9881    -6.5754
  8     8      -2.4073    -3.3406     0.0000       0.4289     -18.9880     -6.5374    -5.9234    -6.5272
--------------------------------------------------------------------------------------------------------
```

The string ‘rho[r]’ together with the spin-pair information show that the attractors were searched in the total 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 always given, because some attractors correspond to positions closer then 0.01 bohr to a nucleus (cf. keyword **close-atom_distance**, Sec. 2.6.2). Instead of the three curvatures only the atomic symbols are printed.

Remark 3.6. 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:

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

  grid maxima closer then the distance *dist* and differing in the function values by less then *vdiff* get the same basin number. If omitted *vdiff* is set to the property value range.

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

  grid maxima inside *iso*-localization domains get the same basin number.

- using the keyword **eli_core**:

  grid maxima inside the respective atomic core defined by ELI-D (internal table) get the same basin number.

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 3 possibilities to compensate for 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 [1] 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 Sec. 4.6). The file name from the refinement procedure is then used with the keyword **integrate** (see the help using `dgrid -u`).

- For Gauss-type orbitals see remark 3.2.

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.

Remark 3.7. The last command in the *control* file is the keyword **end**. All data after this keyword will be ignored. Without the **end** keyword the data will be read till end of file.

3.3 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 *C3H3NO_HF_6-31G.g03.rho_r.bsn* created by the *control* file *rho_basins.inp* mentioned in the introduction:

```
BASINFILE   created by DGrid version 4.6   25-02-2011

:job was executed on:         Fri Feb 25 16:32:09 2011

property_grid=C3H3NO_HF_6-31G.g03.rho_r

basis_from_program:    GAUSSIAN03        basis=C3H3NO_HF_6-31G.g03

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

Property basins cropped by 0.001000 C3H3NO_HF_6-31G.g03.rho_r isosurface

::C3H3NO   HF    6-31G

:atom         x              y              z         vis
:---------------------------------------------------------
 O      -2.06183559     0.57482826     0.00000000       1
 C       0.00000000     2.12106164    -0.00000000       1
 N       2.08694733     0.89554761    -0.00000000       1
 C       1.41345421    -1.67482606    -0.00000000       1
 C      -1.11013465    -1.85958662     0.00000000       1
 H      -0.30771125     4.10249907    -0.00000000       1
 H       2.80485596    -3.12185095    -0.00000000       1
 H      -2.43100862    -3.36800124     0.00000000       1
:---------------------------------------------------------

Energy=    -244.50273 Hartree        Electrons= 18 alpha  +  18 beta
```

```
:             lattice vectors
:------------------------------------
:          a             b             c
:------------------------------------------
x:    1.00000000    0.00000000    0.00000000
y:    0.00000000    1.00000000    0.00000000
z:    0.00000000    0.00000000    1.00000000
:------------------------------------------

:          origin              v_i           v_j           v_k
:--------------------------------------------------------------
x:      -6.19325935       13.22870240   -0.87280762    0.00000000
y:      -7.65693368        0.75095461   15.37524656    0.00000000
z:      -4.00000000        0.00000000    0.00000000    8.00000000
:--------------------------------------------------------------
           points:              266           309           161
```

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. This is followed by a line with the information about the cropping of the grid region. Bear in mind that DGrid reads this grid for cropping. Thus, it should be available at the given location.

The title (line starting with double colon) is followed by the atomic coordinates as 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 (in the above example set per default to a unit box). 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 atom the respective basin can be attributed.

```
:basin  volume   maximum     x       y       z       i    j    k descriptor
:--------------------------------------------------------------------------
    1   40.543    0.4307  -0.297   4.046   0.000   131  227   80         H1
    2   41.243    0.4269  -2.382  -3.334   0.000    81   82   80         H3
    3   42.713    0.4305   2.762  -3.092   0.000   184   81   80         H2
    4   62.995  103.2158   0.013   2.111   0.000   135  188   80         C1
    5   78.962  100.9578   1.429  -1.665   0.000   159  111   80         C2
    6   79.534  117.4257  -1.114  -1.860   0.000   108  110   80         C3
    7   99.674  232.6176  -2.054   0.591   0.000    92  160   80         O1
    8  116.217  155.4799   2.086   0.876   0.000   175  161   80         N1
:--------------------------------------------------------------------------
```

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. The data line below shows that there is a ‘basin’ number zero. This corresponds to the grid points outside the molecular envelope.

```
: start of data
:-------------------------------------------------------------------------
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

It is always a good idea to visualize the basins, cf. Fig. 3.1, to see whether there are some unusual artifacts. Special visualization techniques should be utilized (this creates surfaces around blocks of constant data values, cf. the manual for the Avizo STR-module).

Remark 3.8. Using isosurfaces of basin values (instead of the special techniques) to create the basins one should bear in mind that, for instance, the isosurface with (basin) value 3 is not only found around the basin 3, but also between the basins 1 and 4, 2 and 4, etc.

In DGrid 4.6 there is a new section following the (integer) basin data. This section termed ‘ATTRACTOR SEARCH DATA’ contains (coded) data from the attractor search routine for each grid maximum (for instance the positions of the attractors the grid maximum points towards).

3.4 Output file

The output of the 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 is repeated till the basins reach the final shapes. After the basins are determined the integration section starts. If the **integrate** keyword was not given, then only the volumes of each basin will be determined.

If there is a property file assigned by the **integrate** command, like the electron density grid *C3H3NO_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 property value at the attractor position (maximum of the scalar property that was used to define the basins) as well as the attractor position itself are given:

```
                  rho[r]      rho[r]
BASIN  VOLUME  INTEGRAL    MAXIMUM     <X>    <Y>    <Z>    ATOMS DIST ECCNT
----------------------------------------------------------------------------
   1   40.543    0.8673     0.4320   -0.303  4.067  0.000   H1    0.04   -
   2   41.243    0.8794     0.4289   -2.407 -3.341  0.000   H3    0.04   -
   3   42.713    0.9063     0.4315    2.780 -3.097  0.000   H2    0.04   -
   4   62.995    4.9640   118.3135    0.000  2.121 -0.000   C1    0.00   -
   5   78.962    5.6660   118.3130    1.413 -1.675 -0.000   C2    0.00   -
   6   79.534    5.6154   118.3147   -1.110 -1.860  0.000   C3    0.00   -
   7   99.674    9.0038   291.3237   -2.062  0.575  0.000   O1    0.00   -
   8  116.217    7.9438   192.1030    2.087  0.896 -0.000   N1    0.00   -
----------------------------------------------------------------------------
TOT   561.881   35.8460
```

```
Volume difference:  -1070.518

Basin data written on file C3H3NO_HF_6-31G.g03.rho_r.bsn
```

The above output shows that the QTAIM basins for the hydrogens are slightly positive (around +0.1 each), whereas the carbon atoms are markedly positive (especially C1, cf. basin 4). The oxygen and nitrogen basins are both negatively charged).

If the *basis* file is not present then the attractor positions and property values correspond to the closest grid point (this situation is highlighted by the '+' sign next to the 'MAXIMUM' value). 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 for atoms other then hydrogen, because in this case the attractors of electron density coincide with the atomic positions).

Remark 3.9. When the basins are cropped by an isosurface (keyword **crop**) then the sum of the basin volumes is less then the total grid region volume. In that case the line 'Volume difference:' will be appended.

Following is the output example for the ELI-D basins of the C_3H_3NO molecule (cropped by the 0.001 bohr^{-3} density isosurface):

```
               rho[r] ELI-D[r]
BASIN VOLUME INTEGRAL  MAXIMUM    <X>    <Y>    <Z>    ATOMS  DIST       ECCNT
-----------------------------------------------------------------------------
   1   0.268   2.1288  13.1296  -2.062  0.575  0.000   O1     0.00         -
   2   0.453   2.1099  14.9864   2.087  0.896 -0.000   N1     0.00         -
   3   0.813   2.0918  17.2159   0.000  2.121 -0.000   C1     0.00         -
   4   0.813   2.0952  17.1006   1.413 -1.675 -0.000   C2     0.00         -
   5   0.818   2.0942  17.2365  -1.110 -1.860  0.000   C3     0.00         -
   6   5.462   1.1927   1.5295  -1.633 -0.561  0.000   O1-C3  1.21-1.40 0.014
   7   5.802   1.2585   1.5247  -1.101  1.296  0.000   O1-C1  1.20-1.38 0.001
   8  10.918   1.6963   1.6748   1.797 -0.386  0.000   N1-C2  1.31-1.34 0.044
   9  40.697   2.8042   1.6768   1.051  1.501  0.000   N1-C1  1.20-1.22 0.003
  10  71.846   2.2711   7.6437  -2.501 -3.442  0.000   H3     0.10         -
  11  73.888   2.2487   7.5731   2.883 -3.195  0.000   H2     0.11         -
  12  78.033   2.3840   7.9727  -0.329  4.216  0.000   H1     0.12         -
  13  85.723   4.7544   1.6749  -3.091  0.870  0.000   LP_O1  1.07         -
  14  91.855   3.0218   1.9272   3.314  1.377  0.000   LP_N1  1.32         -
  15  94.491   3.6944   1.7583   0.145 -1.876  0.000   C3-C2  1.26-1.28 0.108
-----------------------------------------------------------------------------
TOT  561.881  35.8460

Volume difference:  -1070.518

Basin data written on file C3H3NO_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 8 atoms). The first 5 basins show the atomic cores of oxygen, nitrogen, and carbon, with ELI-D attractors at the corresponding atoms, i.e., *DIST=0.00*, cf. also the colored basins in Fig. 3.2. The cores are populated roughly by 2.1 electrons.

The basins 6–9 correspond to the O-C and N-C bonds, respectively, with ELI-D attractors each located between two atomic nuclei. *DIST* shows the distance to the particular neighbors. The attractor number 6 is located 1.21 bohr from oxygen and 1.40 bohr from carbon. It is not located exactly at the straight line between the atoms, as shown by the eccentricity (perpendicular distance to the O1-C5 line)

ECCNT=0.014 bohr (whereas the attractor number 7 is already located practically at the interatomic line).

The basins number 10-12 are attributed to the C-H bonds, 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).

Similarly to the basins number 10-12 the ELI-D attractors number 13 and 14 are located in proximity to the oxygen and nitrogen nucleus, respectively. A closer inspection of the data reveals that the corresponding basins describe lone-pairs, cf. the ELI-D diagram in Fig. 3.2. Interestingly, there is just single oxygen lone-pair basin (with the attractor located in the molecular plane) populated by 4.75 electrons (i.e., not split into two separate parts, like in water molecule).

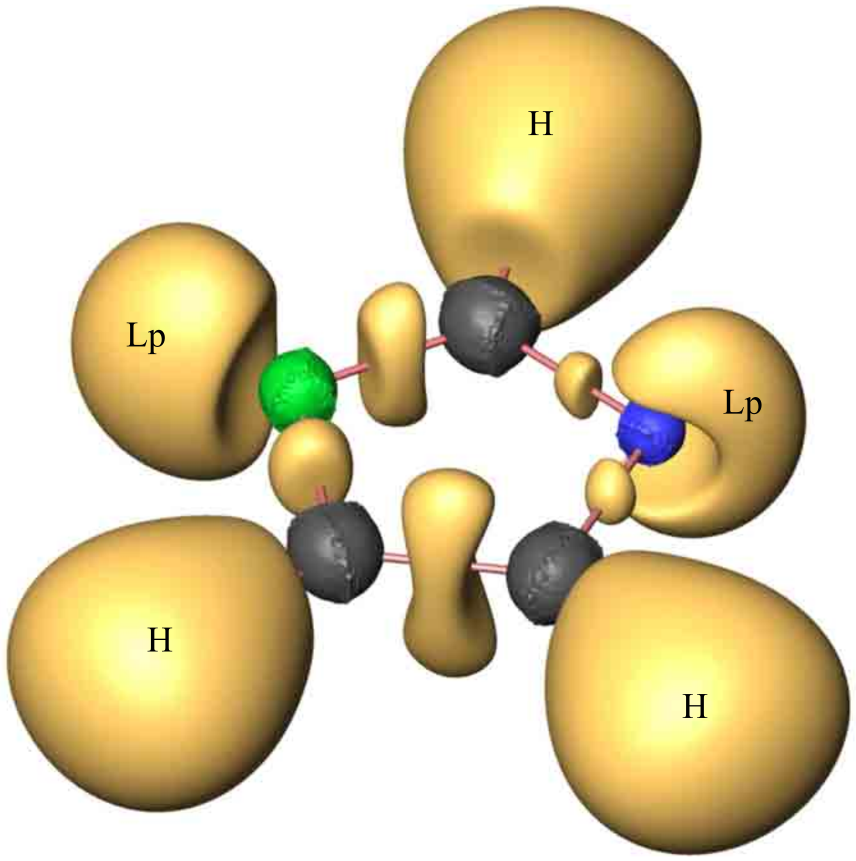

Fig. 3.2 ELI-D for C_3H_3NO. Gold colored (irreducible) 1.45-localization domains (‘H’ and ‘Lp’ mark the hydrogen and lone-pair domains, respectively. Blue: oxygen core basin; green: nitrogen core basin; black: carbon core basins

The distance of the property maximum (attractor) 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 straight 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.

Remark 3.10. The basin descriptors should serve as a hint or help. It is not intended to be used as a rigid classification.

3.5 Search for critical points

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. An important application is the search for the critical points followed by the evaluation of some quantities at those points. In the *control* file below (modified *rho_basins.inp* file) the density *basin* file is used with the keyword **property** and the command **topology**:

```
:TITLE
:------------------------------------------------------------|
::C3H3NO density basins
:------------------------------------------------------------|

:KEYWORDS
:---------------------------------------------------------
 property =C3H3NO_HF_6-31G.g03.rho_r.bsn

 output=.

 topology
 icl_graph=full

end
```

In the above example the basins are not recomputed by DGrid (this would be otherwise the case if the density grid file were used – which also means that it is possible to run the basin search immediately followed by the critical point search in single job). Thus, the basins are directly read from the *basin* file *C3H6_HF_3NO-31G.g03.rho_r.bsn* created in previous run. The headers in the *output* file (the file name ends with ‘*.cp*’) will be the same as already discussed (with a ‘Volume’ section included). Additionally, the ‘Topology’ section will appear in the *output* file. The first part of this section shows all the electron density attractors:

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


------------------------------------------------------------------------------------------------------------
 nr.  basin       x           y           z          value      Laplacian                curvature
------------------------------------------------------------------------------------------------------------
  1     7     -2.0618     0.5748     0.0000     291.3237          -                         O1
  2     8      2.0869     0.8955    -0.0000     192.1030          -                         N3
  3     6     -1.1101    -1.8596     0.0000     118.3147          -                         C5
  4     4      0.0000     2.1211    -0.0000     118.3135          -                         C2
  5     5      1.4135    -1.6748    -0.0000     118.3130          -                         C4
  6     1     -0.3026     4.0665     0.0000       0.4320     -19.1957      -6.6122    -5.9827    -6.6008
  7     3      2.7802    -3.0966     0.0000       0.4315     -19.1472      -6.5837    -5.9881    -6.5754
  8     2     -2.4073    -3.3406     0.0000       0.4289     -18.9880      -6.5374    -5.9234    -6.5272
------------------------------------------------------------------------------------------------------------
```

This part is followed by the information about the saddle points (points with 2 negative and 1 positive principal curvatures):

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


------------------------------------------------------------------------------------------------------------
 nr. basins      x       y       z     value  Laplacian           curvature         ellip  |V|/G      H
------------------------------------------------------------------------------------------------------------
  1    8- 4   0.7730  1.6246  0.0000   0.3740   -1.2348    0.3154  -0.8486  -0.7016  0.21  3.1103  -0.5867
  2    6- 5   0.2026 -1.7737  0.0000   0.3307   -0.8850    0.3332  -0.6833  -0.5349  0.28  3.7727  -0.3461
  3    4- 1  -0.2059  3.4012  0.0000   0.2858   -1.0034   -0.7896   0.5492  -0.7631  0.03  8.8532  -0.2875
  4    6- 2  -1.9621 -2.8230  0.0000   0.2824   -0.9697   -0.7670   0.5333  -0.7360  0.04  8.2561  -0.2812
  5    8- 5   1.6778 -0.6138  0.0000   0.2802   -0.7097   -0.5396   0.3407  -0.5107  0.06  3.3400  -0.3098
  6    5- 3   2.2982 -2.5930  0.0000   0.2798   -0.9334   -0.7398   0.5286  -0.7221  0.02  7.5816  -0.2752
  7    7- 4  -0.6803  1.5630  0.0000   0.2619   -0.1784    0.6952  -0.4217  -0.4519  0.07  2.1419  -0.3589
  8    7- 6  -1.4000 -1.0226  0.0000   0.2446   -0.0688   -0.3527   0.6642  -0.3803  0.08  2.0563  -0.3226
------------------------------------------------------------------------------------------------------------
```

The column ‘basins’ shows between which basins the saddle point is located (the numbering from the ‘Volume’ section is used). 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. From the above data it can be seen that the density Laplacian is negative at all bond critical points of the oxazole molecule.

The ellipticity at the saddle point is given as $c_1/c_2 - 1$, with the two (negative) curvatures c_1 and c_2 perpendicular to the interaction path, whereby $|c_1| \geq |c_2|$. The virial ratio $|V|/G$ is computed from the positive definite kinetic energy density $G = \nabla\nabla'\Gamma(r,r')$ and the local potential energy density [19] is set to $V = \frac{1}{4}\nabla^2\rho - 2G$. The energy density at the saddle point is given by $H = G + V$.

Similar table is given for the ring critical points (points with 1 negative and 2 positive principal curvatures). For the oxazole molecule there is 1 ring critical point:

```
* * *  (3,+1)   R I N G    P O I N T S    I N    R E G I O N  * * *

------------------------------------------------------------------------------------------------------------
 nr. basins      x       y       z     value  Laplacian            curvature         ellip  |V|/G      H
------------------------------------------------------------------------------------------------------------
  1    8- 7   0.0141  0.0039  0.0000   0.0514    0.4363    0.2378   0.2551  -0.0566  0.07  0.9104   0.0090
------------------------------------------------------------------------------------------------------------
```

The column ‘basins’ shows only two of the basins touching at the ring point (instead of all five). The ellipticity is computed similar to the saddle point ellipticity, but using the two positive curvatures.

The last table includes data for the minima (no data for C_3H_3NO – the electron density minimum is at infinity) in the region. The ‘Topology’ section is closed by the information about the Euler characteristic inside the cropped region:

```
Euler characteristics (inside cropped grid):
              ->      8 (Attr)  -  8 (Saddle)  +  1 (Ring)  -  0 (Min)  =  1
```

showing that the Poincare-Hopf sum is fulfilled. This need not be the case if just part of the system was calculated. As mentioned above, the sum applies only to the computed (cropped) region.

The command '**icl_graph**=full' direct the program to create the interconnection lines (ICLs) between the saddle points and the attractors as well as between the ring critical points and the minima (parameter 'full', cf. Sec. 3.8.11, page 86). The resulting data are written to the file *C3H3NO_HF_6-31G.g03.rho_r.bsn.graph.str* in the STR format. In case of electron density the diagram formed by the lines from saddles to the attractors is called molecular graph [9]. Fig. 3.3 shows (program Avizo) the molecular graph for oxazole, together with the nitrogen and hydrogen basins (cropped with 0.001 bohr^{-3} isodensity surface). It can be seen that the saddle points (small green spheres) are located in the zero-flux surfaces (basin borders). The ring critical point (blue sphere) is located within the molecular ring. The blue line follows the gradient path to the minimum in the direction of the negative principal curvature (only the part till the grid border is drawn as it is a long way into the infinity). It can nicely be seen that the trajectory runs along the edge of the nitrogen basin.

Instead of computing the ICLs directly after the topology evaluation it is possible to perform this task with an utility, cf. Sec. 4.7). Commenting out the '**icl_graph**=full' command would result in a topology run with all the data described above, but with-

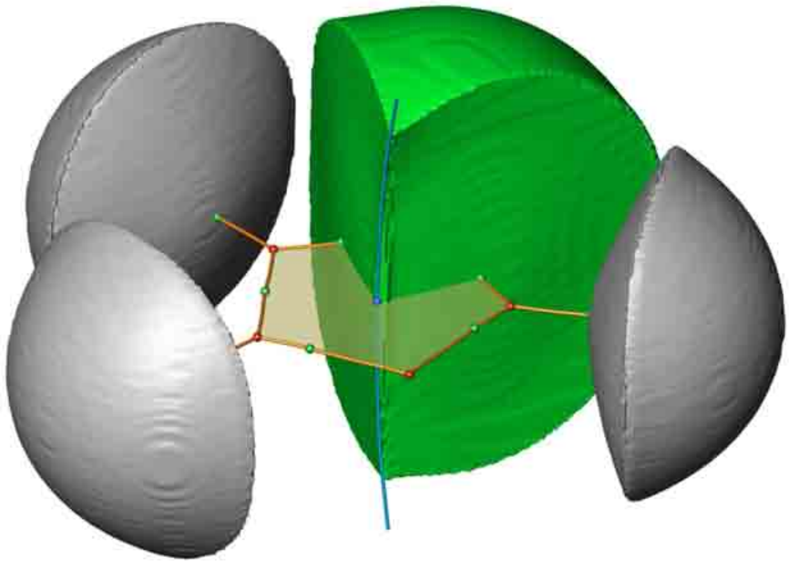

Fig. 3.3 ρ-basins (cropped by the 0.001 bohr^{-3} isosurface of electron density) and the molecular graph for C_3H_3NO. Hydrogen basins are in gray, the nitrogen basin is green colored (the oxygen and carbon basins are not shown). Red spheres: attractors; green spheres: saddle points; blue sphere: ring critical point

out the ICLs. The critical points would be saved in a file named *C3H3NO_HF_6-31G.g03.rho_r.bsn.cp.str*, i.e., without the data for the lines. With this file the interconnection graph can be computed using:

```
dgrid   C3H3NO_HF_6-31G.g03.rho_r.bsn.cp.str   full
```

In case of the density evaluation of the oxazole molecule the topology and ICL graph is a short calculation. However, for larger systems and more oscillating functions with high gradients, like $\nabla^2\rho$ or ELI-D, with large number of critical points it can turn into a time consuming task. Then it is better first to perform the search for the critical points (possibly parallel, cf. Sec. 3.5.1), check whether the Euler characteristics is the desired one (depending on the region or periodicity) and then create the ICL graphs (which can take lot of time as well).

In case of ELI-D there are much more basins then in case of electron density. Especially if heavy elements are involved one encounters the problem of resolving the large number of inner shell basins connected with huge number of critical points.

Remark 3.11. DGrid is working on an equidistant grid of data. Thus, it cannot resolve details hidden between the grid points. Strong oscillations in atomic regions can be taken into account only by appropriately fine mesh. At least 0.05 bohr mesh should be used to properly resolve the core-valence separation for heavier elements. Even then, due to the very high gradients and curvatures, the determination of critical points can fail.

To avoid the complicated as well as time consuming search for critical points in the core region of heavier atoms it is possible to exclude this region from the search. In case of ELI-D tabulated shell radii are utilized with the keyword **ELI_core**. Then, the procedure will not search for critical points within the core region, cf. Table 3.1.

Table 3.1 Core regions with ELI_core keyword

Atoms		Core shells
Li–Ne	$\mapsto$	1^{st}
Na–Ar	$\mapsto$	$1^{st}+2^{nd}$
K	$\mapsto$	$1^{st}+2^{nd}+3^{rd}$
Ca–Cu	$\mapsto$	$1^{st}+2^{nd}$
Zn–Kr	$\mapsto$	$1^{st}+2^{nd}+3^{rd}$
Rb	$\mapsto$	$1^{st}+2^{nd}+3^{rd}+4^{th}$
Sr–Ag	$\mapsto$	$1^{st}+2^{nd}+3^{rd}$
Cd–Xe	$\mapsto$	$1^{st}+2^{nd}+3^{rd}+4^{th}$
Cs–La	$\mapsto$	$1^{st}+2^{nd}+3^{rd}+4^{th}$
Ce–Lu	$\mapsto$	$1^{st}+2^{nd}+3^{rd}$
Hf–Au	$\mapsto$	$1^{st}+2^{nd}+3^{rd}+4^{th}$
Hg–Rn	$\mapsto$	$1^{st}+2^{nd}+3^{rd}+4^{th}+5^{th}$

The radii can be found in the constructor of the class Per_Sys, respectively in Ref. [31]

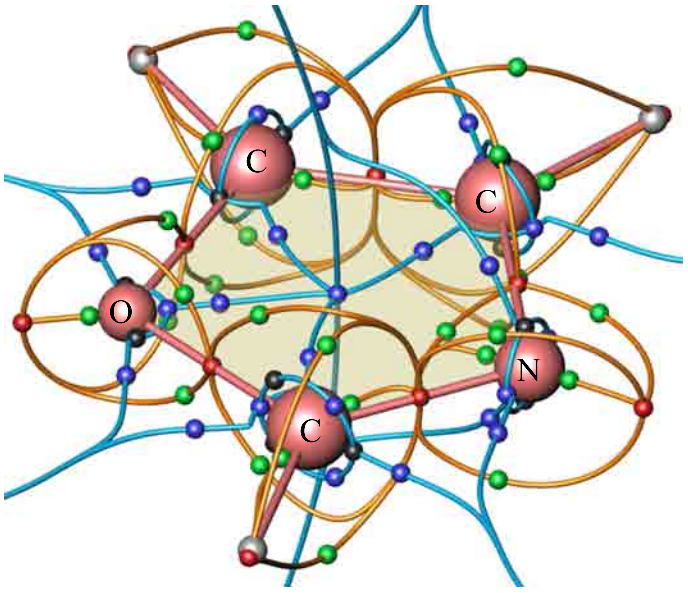

Fig. 3.4 ELI-D ICL graphs for C_3H_3NO. The small colored spheres have following encoding: red - attractors; green - saddle points; blue - ring critical points; black - minima. The hydrogen positions are marked by gray spheres. The gold colored ICLs run toward the attractors, whereas the blue ones run toward the ELI-D minima. The large rose-colored spheres are the respective core regions. The rose-colored stick are used to highlight the molecular structure

The reason for the choice of the 1st and 2nd shell as the core region for the series Ca–Cu is the observation that the structurization of the penultimate atomic shell seems to be important [32] (and similarly for other series). The critical points within the core-valence separation will be searched, whereas each core region will be replaced by a single entity, called ‘core’ and treated as an attractor in the determination of Euler characteristic.

Remark 3.12. In DGrid 4.6 the data from the attractor search are appended to the *basin* file. There, also the usage of **ELI_core** keyword is marked. DGrid stops execution if *basin* file prepared without **ELI_core** is used later for job requiring that keyword.

As an example let us utilize the **ELI_core** keyword in the evaluation of the ELI-D topology for the oxazole molecule. Of course, in this case the core region just replace a single attractor (thus, the same result would be given for the evaluation without **ELI_core** – bear in mind the above remark and prepare the *basin* file with the **ELI_core** keyword active). The following *control* file:

```
:TITLE
:------------------------------------------------------------|
::C3H3NO ELI-D basins
:------------------------------------------------------------|

:KEYWORDS
:---------------------------------------------------------
 property =C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn

 output=.

 ELI_core
 topology

end
```

yields a ‘Topology’ section similar to the one for the evaluation of the electron density shown above. However, the table for the attractors is now followed by a table with the data for the core region:

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

--------------------------------------------------------------------------------
 Nr.  Atom  Basin       x           y           z          value        radius
--------------------------------------------------------------------------------
  1    C5      5    -1.1101     -1.8596      0.0000      17.2365       0.4350
  2    C2      4     0.0000      2.1211     -0.0000      17.2159       0.4350
  3    C4      3     1.4135     -1.6748     -0.0000      17.1006       0.4350
  4    N3      2     2.0869      0.8955     -0.0000      14.9864       0.3600
  5    O1      1    -2.0618      0.5748      0.0000      13.1296       0.3000
--------------------------------------------------------------------------------
```

For the 5 atomic cores the corresponding atomic symbols and basin numbers are given, followed by the position of the atom with the ELI-D value as well as the radius used. For ELI-D there are also 11 minima found in the computed region:

```
* * *  M I N I M A   I N   R E G I O N  * * *

----------------------------------------------------------------------------------------------------------
 nr.     x         y         z      value     Laplacian            curvature              |V|/G     H
----------------------------------------------------------------------------------------------------------
   1  -1.9212   0.9616   0.0000    0.6680     33.7741      0.8159   32.1033    0.8549     1.0338  -0.7313
   2  -2.1365   0.1696   0.0000    0.6674     33.6895      0.8103   32.0235    0.8557     1.0330  -0.7088
   3   2.0331   0.8449  -0.4841    0.6409     24.5149      0.8422    0.1686   23.5042     0.9834   0.1377
   4   2.0331   0.8449   0.4841    0.6409     24.5149      0.8422    0.1686   23.5042     0.9834   0.1377
   5   0.4958   2.4271  -0.0000    0.6290     17.6059     16.9168    0.6421    0.0470     0.9498   0.1645
   6   1.4679  -1.6105   0.5859    0.6281     16.8135      0.3210    0.1148   16.3777     0.9455   0.1603
   7   1.4679  -1.6105  -0.5859    0.6281     16.8135      0.3210    0.1148   16.3777     0.9455   0.1603
   8  -0.8765  -2.3950  -0.0000    0.6230     17.6143      0.6679   16.7977    0.1486     0.9456   0.1755
   9  -0.3759   1.9230  -0.4179    0.6201     16.8559     16.4228    0.3485    0.0846     0.9330   0.1893
  10  -0.3759   1.9230   0.4179    0.6201     16.8559     16.4228    0.3485    0.0846     0.9330   0.1893
  11  -1.3781  -1.3290  -0.0000    0.6118     18.3149      0.2749   18.0043    0.0357     0.9115   0.2545
----------------------------------------------------------------------------------------------------------

Euler characteristics (inside cropped grid):  10 (Attr) - 29 (Bcp) + 26 (Rcp) - 11 (Min) + 5 (Core) = 1

Topology data written to the file C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn.cp.str
```

The addition of the 5 cores yields the proper Euler characteristic. The data for the critical points are written to the file *C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn.cp.str* which can be used to create the ICL graphs for the oxazole molecule:

```
dgrid   C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn.cp.str   full
```

The resulting ELI-D ICL-graphs are not very complicated thus, both ICL graphs (from saddles to attractors and from rings to minima) can be shown at once, cf. Fig. 3.4. The core spheres in the figure have the radius given in the above table of atomic core regions.

Fig. 3.5 shows the same ICL graphs like Fig. 3.4. Additionally, 3 ELI-D basins are shown (corresponding to the C-O and C-N bonds). The borders of the basins (the ELI-D zero-flux surfaces) are colored according to the ELI-D value at the given position. It can nicely be seen, that the ELI-D maxima (2-dimensional; rank 2, signature -2) in the zero-flux surfaces correspond to ELI-D saddle points (3,-1), whereas the ELI-D saddle points in the zero-flux surface (again 2-dimensional; rank 2 signature 0) correspond to the ring critical points (3,+1).

Remark 3.13. The search for critical points is using certain thresholds for gradient, curvatures, etc. in DGrid. If the critical points are in regions below the thresholds,

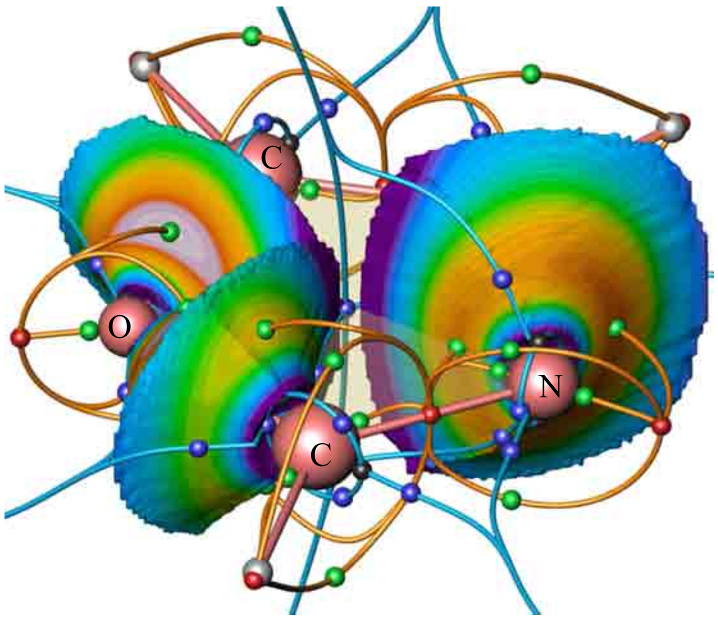

Fig. 3.5 ELI-D basins and ICL graphs for C_3H_3NO. The small colored spheres have following encoding: red - attractors; green - saddle points; blue - ring critical points; black - minima. The hydrogen positions are marked by gray spheres. The gold colored ICLs run toward the attractors, whereas the blue ones run toward the ELI-D minima. The large rose-colored spheres are the respective core regions. The rose-colored stick are used to highlight the molecular structure. The 3 ELI-D basins (corresponding to the C-O and C-N bonds) are colored according the the ELI-D value at the zero-flux surface

then they cannot be detected. For instance in almost spherical regions around atoms in a molecule, which of course cannot be perfectly spherical, however the changes in gradient and curvatures are close to numerical instabilities. Even if the Euler characteristics is the proper one, there still could be some critical points missing (note that if both an attractor *and* a saddle point were not found the Euler characteristics remains unchanged).

3.5.1 Parallel search for critical points

In DGrid the determination of critical points is performed by different routines depending on whether the attractors and minima or saddle and ring points, respectively are searched.

In case of attractors first the discrete grid maxima are determined. Here, a discrete grid maximum is a grid point with property value higher than at the 26 closest points. Whether a grid minimum is actually an attractor is finally determined by trajectory search. For steep functions respectively coarse grids many grid maxima can lead to the same attractor. When the search for attractors consume too much time it is convenient to perform the task in parallel. This can be done in two ways (like in case of a grid-slice calculation, cf. Sec. 2.5.1). Running DGrid with the following *control* file:

```
:TITLE
:-------------------------------------------------------|
::C3H3NO ELI-D basins
:-------------------------------------------------------|
 property =C3H3NO_HF_6-31G.g03.elid_r_t_tr

 output=.

 attractors   slices=3  1

END
```

finds 15 ELI-D grid maxima (potential attractors), prepares 3 slices with 5 maxima each, and searches in the first slice for the attractors. This can be seen from the output:

```
15 grid maxima

Computing slice nr. 1 of 3

                grid maxima
0                                                 5
|----*----|----*----|----*----|----*----|----*----|
|>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>|

Search data written to file C3H3NO_HF_6-31G.g03.elid_r_t_tr.search_max.slice-1
```

The results of the search is written to a file with '.search_max.slice-1' appended. After all 3 slice files are generated, the search file can be created with the DGrid 'complete' utility:

```
dgrid   name.search_max.slice-1   complete
```

Remark 3.14. Observe that the search for the attractors is done for the grid file with the ELI-D field and not for the corresponding *basin* file. In case of the *basin* file the attractor search results are already appended to the *basin* file. Thus, no search is necessary and DGrid gives out only a warning message.

The whole slicing procedure can be done with the *dgrid_para* script and the above *control* file (with the command **output**= commented out). The script detects the 'attractors' command in the *control* file and asks for the number of slices. The corresponding number of *control* files will be generated and submitted to the processors. Finally, the complete search file is built as described above.

Similar procedure can be used for the (parallel) detection of minima, saddle points and ring critical points. Internally, concerning the search for the saddle and ring points, the detection of possible candidates is done by other routines that in case of attractors and minima. However, the overall procedure is roughly the same. Using the *dgrid_para* script only the corresponding *control* file must be changed. Instead of **attractors** use the keywords **minima** and **saddles**, respectively. Additionally, the search for the saddle and ring points can be started from the corresponding *basin* file.

For further evaluation the completed search files can be utilized in a *control* file using the keyword **search_file**=, as described in Sec. 3.8.

3.6 Superbasins

In some cases it is desirable to unify basins into single (connected or disconnected) object, so called superbasin. Be it all the core basins of an atom, some basins in the valence region or even basins of different chemical groups.

As a prerequisite to the creation of superbasins DGrid must identify all the basins in the computed region and write the basin data into the *basin* file, cf. the previous section. Then, for instance, the *C3H3NO_HF_6-31G.g03.rho_r.bsn* file can be read with the **property** keyword and the integration of electron density over basins performed. But in contrast to the example at page 53, now with the C and H basins grouped into C-H superbasins:

```
:TITLE
:-----------------------------------------------------------|
::C3H3NO density superbasins
:-----------------------------------------------------------|
:KEYWORDS
:----------------------------------------------------------
 property =C3H3NO_HF_6-31G.g03.rho_r.bsn
 integrate=C3H3NO_HF_6-31G.g03.rho_r

 output=.

 superbasin
   C1-H1 = 4 1
   C2-H2 = 5 3
   C3-H3 = 6 2
 superbasin_end

end
```

The definition of the superbasins starts with the keyword **superbasin**. It must be the only keyword on the line. The block must be closed with the keyword **superbasin_end**. Within the block each superbasins is given a descriptor, e.g., 'C1-H1' for the first C-H group followed by the equal sign and the basin numbers of the constituting basins (accessible from the file *C3H3NO_HF_6-31G.g03.rho_r.bas*). Any basins can be chosen to form a superbasins.

Remark 3.15. In contrast to a superbasin, so called basin set [32] is given by special choice of basins. Basin set is join of sets of interconnected basins where the basin interconnection points are above or equal to a specified value (the basin interconnection point is the defined by the position and property value for the saddle point connecting two basins). Thus, the basin set is always an object formed by connected basins. Not every superbasins is at the same time a basin set. In the left diagram of Fig. 3.6 the 0.3-localization domains are shown (isosurfaces for the density value 0.3 bohr^{-3}). It can be seen that none of the C and H density basins, as chosen in the above *control* file, can form separate basin set (by lowering the density to connect the C-H one cannot avoid that C-N is already connected as well). However, the N and C as well as the two carbon basins form (2 membered) basin sets to the interconnection value $\rho_{\text{ic}} = 0.3$, whereas the hydrogen and oxygen basins remain separated. Lowering the interconnection value to $\rho_{\text{ic}} = 0.281$ gives two 3-membered basin sets (NCH and CCH) and two separate basins (H and O). For the corresponding basins cf. Fig. 3.1.

In the output an additional section preceding the basin information will appear:

```
Creating  3  superbasins:

SB   Descriptor        #   basins
--------------------------------------------------------
1    C1-H1             1   4
2    C2-H2             3   5
3    C3-H3             2   6
--------------------------------------------------------
```

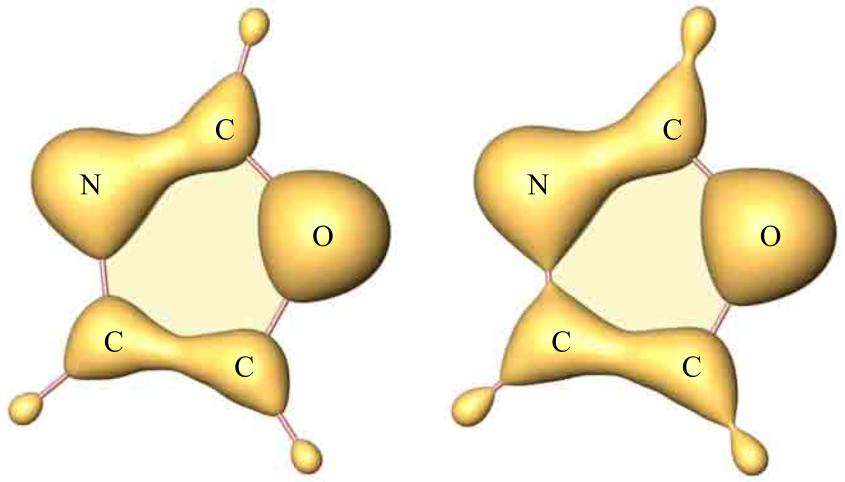

Fig. 3.6 Electron density isosurfaces for C_3H_3NO. Left: 0.3-localization domains; the N-C and C-C groups are within separate reducible localization domains. Right: 0.281-localization domains; now the N-C-H and C-C-H groups are within separate reducible localization domains

showing that 3 superbasins were created, with corresponding descriptors. The numbers for the constituting basins are also repeated in the output. In the basin data following this section the correspondence between the basins and superbasins is highlighted:

```
                    rho[r]     rho[r]
BASIN    VOLUME  INTEGRAL    MAXIMUM     <X>    <Y>    <Z>    ATOMS DIST ECCNT
------------------------------------------------------------------------------
   1 S1 103.537    5.8313     0.4320   -0.303  4.067  0.000   H1    0.04   -
   2 S3 120.778    6.4948     0.4289   -2.407 -3.341  0.000   H3    0.04   -
   3 S2 121.675    6.5723     0.4315    2.780 -3.097  0.000   H2    0.04   -
   4 S1    -         -      118.3135    0.000  2.121 -0.000   C1    0.00   -
   5 S2    -         -      118.3130    1.413 -1.675 -0.000   C2    0.00   -
   6 S3    -         -      118.3147   -1.110 -1.860  0.000   C3    0.00   -
   7      99.674   9.0038   291.3237   -2.062  0.575  0.000   O1    0.00   -
   8     116.217   7.9438   192.1030    2.087  0.896 -0.000   N1    0.00   -
------------------------------------------------------------------------------
TOT     561.881   35.8460

Volume difference:  -1070.519

Basin data written on file C3H3NO_HF_6-31G.g03.rho_r.bsn.super
```

Here, for instance, S1 means basin belonging to the superbasin number 1, i.e. superbasin ‘C1-H1’. Observe that the total volume (as well the total integral) of the superbasin number 1 is given for the volume of the first basin member of the superbasin (in the example basin for the hydrogen H1). For the remaining members, in this case the basin number 4, there are just periods instead of the volume and integral.

A new *basin* file, incorporating the superbasins, with *.super* appended to the original *basin* file name is created. In this file the correspondence between the basins and

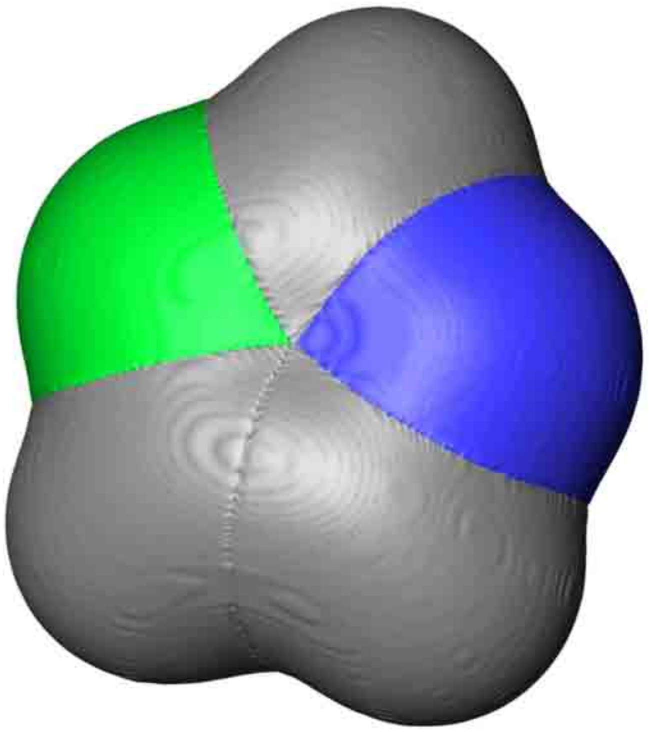

Fig. 3.7 Density superbasins for C_3H_3NO, cropped by 0.001 bohr^{-3} electron density isosurface. Blue: oxygen basin; green: nitrogen basin; dark gray: three C-H superbasins;

superbasins is given in the basin descriptor section. Fig. 3.7 visualizes the resulting superbasin data file. Now there are only 5 basins instead of 8, cf. Fig. 3.1, because each of the previously separate C and H basins form C-H groups – the superbasins.

3.7 Overlap integrals

The integration of electron density over basins yields the basin populations (the average number of electrons within given region), which provides information about the atomic charges, respectively speculative reference to the bond order (for ELI-D basins). For the examination of chemical bonding it is of interest to analyze the electron pair density as well. In Sec. 2.3 it was already mentioned that it is possible to compute some pair-density properties, cf. Table 2.6, however there the focus was on the conditional properties. The reason was that the pair density itself is a 6-

dimensional function and one need to reduce 3 dimensions to get a 3D-property grid.

This section deals with integrals of electron pair density. For molecules DGrid is using Slater-type or Gauss-type orbitals approximating the wave function by one or more determinants build up from the orbitals. From this Ansatz the 1-matrix and 2-matrix is formed, providing the pivot for all the property evaluations.

Remark 3.16. In DGrid the 2-matrix is created formally by the reduction of the determinantal Ansatz. It is not guaranteed that the 2-matrix properly describe the pair interaction. For instance, starting from a DFT result yielding wavefunction represented by Kohn-Sham orbitals DGrid creates the 2-matrix the same way as for a HF calculation, even if the DFT wavefunction is not meant to properly describe the local pair interactions, but 'just' the (correlated) electron density.

The electron pair density $\rho_2(\mathbf{r}_1,\mathbf{r}_2)$ is the diagonal part of the reduced 2-matrix $\rho_2(\mathbf{r}_1'\mathbf{r}_2',\mathbf{r}_1\mathbf{r}_2)$. The 2-matrix can be decomposed into same-spin and opposite-spin contributions, which in the (restricted) orbital representation can be written as:

$$\rho_2^{\sigma\sigma}(\mathbf{r}_1'\mathbf{r}_2',\mathbf{r}_1\mathbf{r}_2) = \frac{1}{2}\sum_{i<j}^{n}\sum_{k<l}^{n} P_{ij,kl}^{\sigma\sigma}\,|\phi_i(\mathbf{r}_1')\phi_j(\mathbf{r}_2')|\,|\phi_k^*(\mathbf{r}_1)\phi_l^*(\mathbf{r}_2)| \tag{3.1}$$

and

$$\rho_2^{\sigma\sigma'}(\mathbf{r}_1'\mathbf{r}_2',\mathbf{r}_1\mathbf{r}_2) = \frac{1}{2}\sum_{i,j}^{n}\sum_{k,l}^{n} P_{ij,kl}^{\sigma\sigma'}\,\phi_i(\mathbf{r}_1')\phi_j(\mathbf{r}_2')\phi_k^*(\mathbf{r}_1)\phi_l^*(\mathbf{r}_2). \tag{3.2}$$

Thus, the determination of the integrals of electron pair density over basins are reduced to the calculation of the overlap integrals $S_{ik} = \int_\Omega \phi_i(\mathbf{r})\phi_k^*(\mathbf{r})\mathrm{d}V$ of the molecular orbitals over the basins. To accomplish this the factorization of the Gauss functions can be utilized.

Remark 3.17. This factorization is necessary to achieve adequate speed up for the evaluation. For this reasons the calculation of the overlap integrals is performed in DGrid only for Gauss-type orbitals and only if the grid is oriented parallel with the Cartesian axes.

First, let us compute for the oxazole molecule the electron density grid parallel with the Cartesian axes:

```
:TITLE
:-------------------------------------------------|
::C3H3NO   HF    6-31G      parallel mesh
:-------------------------------------------------|

:KEYWORDS
:-------------------------------------------------
 basis=C3H3NO_HF_6-31G.g03

 result=C3H3NO_HF_6-31G_parallel
```

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

 mesh=0.05   4.0   parallel

END
```

The resulting density grid is written to the file *C3H3NO_HF_6-31G_parallel.rho_r*. The grid-box is parallel with the Cartesian axes which can readily be seen in the output where the definition of the grid region is written:

```
REGION :  origin = [ -6.450,  -7.350,  -4.000]

i = [ 13.250,  0.000, 0.000]   |i| =  13.250 by 266 points   d|i| = 5.000e-02
j = [  0.000, 15.450, 0.000]   |j| =  15.450 by 310 points   d|j| = 5.000e-02
k = [  0.000,  0.000, 8.000]   |k| =   8.000 by 161 points   d|k| = 5.000e-02
```

The 3 grid vectors i, j, k form a diagonal matrix. Consult Sec. 3.2 for the determination of the basins for the resulting density grid (conveniently with the 0.001 cropping). The overlap integrals over the density basins are computed with the following *control* file using the keyword **overlap**:

```
:TITLE
:-----------------------------------------------------------|
::C3H3NO overlap over density basins
:-----------------------------------------------------------|

:KEYWORDS
:-----------------------------------------------------------
 property =C3H3NO_HF_6-31G_parallel.rho_r.bsn
 integrate=C3H3NO_HF_6-31G_parallel.rho_r

 output=.

 overlap

end
```

The command ‘**property**=C3H3NO_HF_6-31G_parallel.rho_r.bsn’ tells DGrid where to find the file with basins over which to compute the overlap integrals. With the assignment ‘**integrate**=C3H3NO_HF_6-31G_parallel.rho_r’ the basis set is chosen (namely the one used to compute the density grid).

Remark 3.18. With this combination it is possible to compute the basins with chosen basis set and then determine the overlap integrals for a different basis set over those basins. One could even create a *basin* file with artificial basins and then compute the desired overlap integrals.

3.7.1 Fluctuation and delocalization indices

The output from the overlap run is written to a file with the extension '*.ovl*' (instead of '*.bas*'). There the 'Integration' section is followed by the 'Pair density analysis' section. The progress for the calculation of the overlap integrals of the primitive functions (S_munu) and of the MOs (S_ij) is shown by running arrows.

Remark 3.19. The arrows are not shown in case of parallel calculation of the overlap integrals. Such a parallel calculation using `proc` processors is started with the command:

```
dgrid  -n proc    controlfilename
```

After the calculation of the overlap integrals is finished the name of the file where the overlap tables are saved is given. The resulting overlap integrals are used to evaluate the localization (LI) and delocalization indexes (DI). The indices are given by integrals over the exchange-correlation part of the pair density (exchange only for HF) [20, 21]:

$$\int_A \rho(\mathbf{r}_1)\mathrm{d}\mathbf{r}_1 \int_B \rho(\mathbf{r}_2)\mathrm{d}\mathbf{r}_2 - \int_A \mathrm{d}\mathbf{r}_1 \int_B \rho_2(\mathbf{r}_1\mathbf{r}_2)\mathrm{d}\mathbf{r}_2 \tag{3.3}$$

which for the HF case yields:

$$\sum_{i,j} \int_A \phi_i(\mathbf{r}_1)\phi_j^*(\mathbf{r}_1)\mathrm{d}\mathbf{r}_1 \int_B \phi_j(\mathbf{r}_2)\phi_i^*(\mathbf{r}_2)\mathrm{d}\mathbf{r}_2 = \sum_{i,j} S_{ji}^A S_{ji}^B \tag{3.4}$$

For $A = B$ the LI is obtained otherwise the DI results. With natural orbitals (from correlated calculation) two different formulae are used. The idea of sharing index [23] the above overlap integrals S_{ij} are multiplied with the square root of the product occupation numbers, whereas following Ángyán [8] the product of the occupation numbers is used.

First the table for the localization indices is given:

```
                  +----------------------+
                  | Localization indices |
                  +----------------------+

Basin Descriptor Q       Daa     Dbb     Dab    sigma2    LI     LIaa    LIbb
------------------------------------------------------------------------------
   1  H1       0.865   0.006   0.006   0.187    0.514   0.350   0.175   0.175
   2  H3       0.879   0.007   0.007   0.193    0.520   0.360   0.180   0.180
   3  H2       0.908   0.008   0.008   0.206    0.527   0.381   0.190   0.190
   4  C1       4.968   2.285   2.285   6.170    1.767   3.201   1.600   1.600
   5  C2       5.662   3.084   3.084   8.015    1.968   3.695   1.847   1.847
   6  C3       5.617   3.010   3.010   7.886    1.883   3.733   1.867   1.867
   7  O1       9.001   8.170   8.170  20.257    1.169   7.832   3.916   3.916
   8  N1       7.943   6.292   6.292  15.772    1.569   6.374   3.187   3.187
------------------------------------------------------------------------------
              35.843  22.862  22.862  58.686    9.917  25.926  12.963  12.963
```

For each basin the descriptor and the electron population (Q) is given. The populations are computed analytically and can be compared to the numerical results in the preceding ‘Integration’ section.

Remark 3.20. The density integrals are computed analytically in this section. However, the regions (basins) over which the integrations are performed are still the same ones formed from small cubes given by the grid mesh. Especially for small basins are coarse mesh one should not mistake the populations with ‘exact’ basin populations for which also an analytical determination of the zero-flux surfaces would be necessary (but the errors are very small).

In next columns the average numbers of same-spin electron pairs (Daa and Dbb) as well as opposite-spin pairs (Dab) are given. The numbers are given by the integral of the corresponding pair densities, cf. Eqs. 3.1 and 3.2, with both coordinates confined to the given basin region.

The last 4 columns ‘sigma2’, ‘LI’, ‘LIaa’, and ‘LIbb’ shows the fluctuation (variance) in the average population of the respective region and the localization indexes (total as well as spin resolved). Note that in literature sometimes differing symbols can be found (Fradera [21] used λ for the localization index LI, respectively σ^2 for the fluctuation, whereas Bader [11, 12] used $F(\Omega,\Omega)$ for LI, Λ for the fluctuation (and λ for the relative fluctuation). Table 3.2 should clarify the meaning of the symbols as used by DGrid.

Remark 3.21. The indices ‘sigma2’ and ‘LI’ are the total values summed up over the spin components.

The information about localization is followed by table of delocalization indices $\delta(A,B)$, summed up over the spin components. The delocalization indices can be connected with the number of electron pairs shared between the basins (when considering the number Q_AQ_B of pairs as not being shared – with the total number of electron pairs between the regions A and B equal to $Q_AQ_B - \delta(A,B)$):

```
                +------------------------+         +-------+
                | Delocalization indices |         | Total |
                +------------------------+         +-------+


   Basin      1        2        3        4        5        6        7        8
   :--------------------------------------------------------------------------------
      1       x     0.0005   0.0004   0.4436   0.0050   0.0053   0.0230   0.0267
      2    0.0005     x      0.0006   0.0048   0.0189   0.4586   0.0215   0.0052
      3    0.0004   0.0006     x      0.0032   0.4638   0.0224   0.0060   0.0203
      4    0.4436   0.0048   0.0032     x      0.0262   0.0536   0.4550   0.7595
      5    0.0050   0.0189   0.4638   0.0262     x      0.7946   0.0710   0.5618
      6    0.0053   0.4586   0.0224   0.0536   0.7946     x      0.4641   0.0578
      7    0.0230   0.0215   0.0060   0.4550   0.0710   0.4641     x      0.1075
      8    0.0267   0.0052   0.0203   0.7595   0.5618   0.0578   0.1075     x
   :--------------------------------------------------------------------------------
```

The delocalization indices $\delta(A,B)$ between the basins are given only for $A \neq B$. The diagonal values in the table ($A = B$) are the localization indices LI. Following sum rule is valid: $\sigma^2(A) = \sum_{A\neq B}\delta(A,B)$, respectively half of the sum when using

Table 3.2 Localization and delocalization indices

Index	Description	Equation
$N(\Omega)$	average population in Ω	$\int_\Omega \rho(\mathbf{r})\mathrm{d}\mathbf{r}$
$D^{\alpha\alpha}(\Omega)$	$\alpha\alpha$-pairs in Ω	$\iint_\Omega \rho_2^{\alpha\alpha}(\mathbf{r}_1,\mathbf{r}_2)\mathrm{d}\mathbf{r}_1\mathrm{d}\mathbf{r}_2$
$D^{\alpha\beta}(\Omega)$	$\alpha\beta$-pairs in Ω	$\iint_\Omega \rho_2^{\alpha\beta}(\mathbf{r}_1,\mathbf{r}_2)\mathrm{d}\mathbf{r}_1\mathrm{d}\mathbf{r}_2$
$D^{\beta\alpha}(\Omega)$	$\beta\alpha$-pairs in Ω	$\iint_\Omega \rho_2^{\beta\alpha}(\mathbf{r}_1,\mathbf{r}_2)\mathrm{d}\mathbf{r}_1\mathrm{d}\mathbf{r}_2$
$D(\Omega)$	average number of pairs in Ω	$D^{\alpha\alpha}(\Omega)+D^{\beta\beta}(\Omega)+D^{\alpha\beta}(\Omega)+D^{\beta\alpha}(\Omega)$
$F(\Omega)$		$N^2(\Omega)-2D(\Omega)^a$
$\sigma^2(\Omega)$	fluctuation in population in Ω	$\langle N^2\rangle_\Omega-\langle N\rangle_\Omega^2=N(\Omega)-F(\Omega)$
$D(\Omega,\Omega')$	$\mathbf{r}_1$ in Ω, $\mathbf{r}_2$ in Ω'	$\int_\Omega \mathrm{d}\mathbf{r}_1\int_{\Omega'}\rho_2(\mathbf{r}_1,\mathbf{r}_2)\mathrm{d}\mathbf{r}_2$
$F(\Omega,\Omega')$	$\mathbf{r}_1$ in Ω, $\mathbf{r}_2$ in Ω'	$N(\Omega)N(\Omega')-2[D(\Omega,\Omega')$
Q	charge	$N(\Omega)$
Daa	$\alpha\alpha$-pairs	$D^{\alpha\alpha}(\Omega)$
Dbb	$\beta\beta$-pairs	$D^{\beta\beta}(\Omega)$
Dab	opposite-spin pairs	$D^{\alpha\beta}(\Omega)+D^{\beta\alpha}(\Omega)$
sigma2	fluctuation	$\sigma^2(\Omega)$
lambda	relative fluctuation	$\sigma^2(\Omega)/N(\Omega)$
LI	localization index	$F(\Omega)$
$\delta(A,B)$	delocalization index	$F(\Omega,\Omega')^b$
$D(A,B)$	pairs between regions A and B	$D(\Omega,\Omega')$

^{a} Number of pairs normalized to $N(N-1)/2$. In Ω: $D(\Omega)=\frac{1}{2}\left[N^2(\Omega)-F(\Omega)\right]$
^{b} often defined as $F(\Omega,\Omega')+F(\Omega',\Omega)$

A,B as cumulative symbol for $A,B+B,A$, which depends on the definition of the delocalization index, see Table 3.2. However, summing up the delocalization indices for the basin number 8 (nitrogen) yields 1.539, which is less then $\sigma^2(8)=1.569$ in the table of the localization indices. This discrepancy is due to the cropping of the basins. The missing part is the delocalization between the nitrogen basin 8 and the region outside the 0.001-envelope (which contains 0.157 electrons, cf. the total charge in the localization table).

Remark 3.22. In case of natural orbitals the LI and DI indices are given first for the Ángyán form ($\sum_{i,j} n_i n_j\, S_{ji}^A\, S_{ji}^B$) and then for the Fulton formulation using $\sum_{i,j}\sqrt{n_i n_j}\, S_{ji}^A\, S_{ji}^B$.

Remark 3.23. For correlated wavefunctions the LI/DI indices are computed from the integral over the whole correlated pair density followed by the subtraction of the electron populations in the examined regions, cf. Eq. 3.3. Subtracting from the total correlated LI/DI indices the LI/DI in the Ángyán form (form which first the natural orbitals need to be created) yields the cumulant contribution to the indices.

The last two tables in the output show the number of same-spin and opposite-spin pairs formed between the basins.

```
             +-----------------+
             | Same-spin pairs |
             +-----------------+

Basin     1        2        3        4        5        6        7        8
:-----------------------------------------------------------------------------
   1    0.012    0.190    0.196    0.852    1.222    1.212    1.934    1.704
   2    0.190    0.014    0.199    1.090    1.235    1.006    1.968    1.744
   3    0.196    0.199    0.016    1.126    1.053    1.263    2.040    1.792
   4    0.852    1.090    1.126    4.570    7.019    6.949   10.953    9.485
   5    1.222    1.235    1.053    7.019    6.168    7.553   12.706   10.962
   6    1.212    1.006    1.263    6.949    7.553    6.020   12.407   11.124
   7    1.934    1.968    2.040   10.953   12.706   12.407   16.340   17.820
   8    1.704    1.744    1.792    9.485   10.962   11.124   17.820   12.585
:-----------------------------------------------------------------------------
      303.334 same-spin pairs
```

For a closed-shell calculation the number of same-spin electron pairs within a basin (the diagonal part of the table) is twice the ‘Daa’ value in the table of localization indices. The total number of same-spin pairs for oxazole is $2(18 \times 17)/2 = 306$. The difference between this value and the one found is again due to the cropping of the basins. The same applies to the number of opposite-spin pair and the total number of electron pairs.

3.7.1.1 Fluctuation and delocalization indices in solids

The localization and delocalization indices can also be evaluated for solid state calculations. In this case the overlap integrals between the Bloch functions at each *k* point are computed. The procedure was implemented [14] for the full-potential APW DFT program Elk [6]. . The *control* file as well as the output is similar to the molecular case.

3.7.2 Overlap over superbasins

The same procedure for the calculation of the overlap integrals with the successive evaluation of the fluctuation and delocalization indices can be applied to the superbasins, cf. Sec. 3.6. There are two possibilities to perform such calculation:

- either create the *basin* file with the desired superbasin in advance by a separate job, see for example the *control* file on page 64. Then the overlap integrals can be computed with the *control* file from page 69 using the *basin* file *name.bsn.super*.
- or the original *basin* file is used with the **property** keyword, but the ‘superbasin’ input section, cf. example on page 64, is included into the *control* file.

In both cases the data in the ‘Pair density analysis’ section will be given only for the superbasins (i.e., the basins having only periods for the volumes in the ‘Integration’ section will be omitted):

```
                 +----------------------+
                 | Localization indices |
                 +----------------------+


Basin Descriptor  Q      Daa     Dbb     Dab    sigma2    LI     LIaa    LIbb
------------------------------------------------------------------------------
   1 S1 C1-H1   5.833   3.143   3.143   8.505    1.394   4.439   2.219   2.219
   2 S3 C3-H3   6.496   4.022   4.022  10.549    1.486   5.010   2.505   2.505
   3 S2 C2-H2   6.570   4.145   4.145  10.791    1.567   5.003   2.502   2.502
   7    O1      9.001   8.170   8.170  20.257    1.169   7.832   3.916   3.916
   8    N1      7.943   6.292   6.292  15.772    1.569   6.374   3.187   3.187
------------------------------------------------------------------------------
               35.843  25.773  25.773  65.874    7.185  28.658  14.329  14.329
```

The correspondence between the basins and superbasin descriptors is given in the ‘Integration’ section as described in Sec. 3.6. The same applies to the table of delocalization indices:

```
                 +------------------------+         +-------+
                 | Delocalization indices |         | Total |
                 +------------------------+         +-------+


Basin        1        2        3        7        8
:----------------------------------------------------
   1         x      0.0641   0.0348   0.4780   0.7862
   2      0.0641      x      0.8366   0.4856   0.0630
   3      0.0348   0.8366     x       0.0770   0.5821
   7      0.4780   0.4856   0.0770     x      0.1075
   8      0.7862   0.0630   0.5821   0.1075     x
:----------------------------------------------------
```

Observe that the localization index (LI) for the superbasin S1 (formed by the basins number 1 and 4) is not just the sum $3.551 = 0.350 + 3.201$ of the LI(1) and LI(4), cf. table on page 70. One needs to add the delocalization indices $\delta(1,4) = \delta(4,1)$, cf. table on page 71 as well, $3.551 + 2 \times 0.444 = 4.439$. Thus, it is possible to compute the superbasin localization indices from the ones for the original basins, however with inconvenience increasing with the number of basins participating on the superbasin.

The overlap matrix file *C3H3NO_HF_6-31G_parallel.rho_r.bsn.super.sij* contains only the overlap matrices for the superbasins (and the unchanged basins).

3.7.3 Domain-averaged Fermi-hole

Running the *control* file given on page 69 creates a new input file *C3H3NO_HF_6-31G_parallel.rho_r.bsn.sij* (i.e., with ‘*.sij*’ appended) containing tables with the overlap integrals over the basins:

```
OVERLAP  computed by DGrid version 4.6   09-03-2011

:job was executed on:         Wed Mar  9 11:10:52 2011

for_basin_grid=C3H3NO_HF_6-31G_parallel.rho_r.bsn

basis_from_program:      GAUSSIAN03             basis=C3H3NO_HF_6-31G.g03

:            +---------------------------------------+
             | OVERLAP MATRICES    basis:   RESTRICTED |
:            +---------------------------------------+

:-------------------------------------------------------------
For_basins_No.:    1    2    3    4    5    6    7    8
:-------------------------------------------------------------

MOs =  18 alpha +   18 beta

Basin:   1   H1

  MO          1             2             3             4             5
:-----------------------------------------------------------------------
   1    0.0000000580
   2   -0.0000000568  0.0000000793
   3    0.0000000420  0.0000000501  0.0000057449
   4   -0.0000000052  0.0000000171  0.0000000316  0.0000000289
   5    0.0000000008  0.0000000073  0.0000000188  0.0000000202  0.0000000148
   6    0.0000006484 -0.0000012432  0.0000659691 -0.0000011816 -0.0000007307
   7   -0.0000025793  0.0000018676  0.0000546727 -0.0000004441 -0.0000003223
   8   -0.0000001993  0.0000020303 -0.0002454305  0.0000028692  0.0000015930
   9    0.0000066205 -0.0000084712  0.0002560130 -0.0000029055 -0.0000010913
  10   -0.0000020415 -0.0000015992  0.0005130272 -0.0000039027 -0.0000016260
  11   -0.0000000316  0.0000034565 -0.0003417760  0.0000036060  0.0000016809
  12   -0.0000000000 -0.0000000000 -0.0000000000 -0.0000000000  0.0000000000
  13   -0.0000041345 -0.0000002224  0.0005186220 -0.0000029215 -0.0000006770
  14    0.0000066047 -0.0000102771  0.0003739709 -0.0000033945 -0.0000007897
  15    0.0000009393 -0.0000014054  0.0001945107 -0.0000001080  0.0000006587
  16    0.0000000000 -0.0000000000 -0.0000000000  0.0000000000  0.0000000000
  17   -0.0000035488  0.0000051762 -0.0001414969  0.0000004660 -0.0000005724
  18   -0.0000000000 -0.0000000000  0.0000000000 -0.0000000000 -0.0000000000
:-----------------------------------------------------------------------

...
...

  MO         16            17            18
:-------------------------------------------
  16    0.0012143388
  17    0.0000000000  0.0113340027
  18   -0.0025370523 -0.0000000000  0.0055276644
:-------------------------------------------


Basin:   2   H3

...
```

The header start with the descriptor OVERLAP. In next lines the file name of the *basin* file as well as the information about the basis set is given. List of basin numbers for which the overlap were computed follows. The ‘MOs =’ informs about the total number of orbitals used for each spin channel. For each of the selected basins a table of overlap integrals follows. For calculations with unrestricted basis set the overlaps for the beta channel are also given.

The overlap data can be further utilized, for instance, to compute the domain-averaged Fermi-hole (DAFH) [34], cf. Table 2.6 on page 18. To compute DAFH the overlap integrals as well as the basin to which one of the pair density coordinates is confined are introduced to DGrid with the keyword **overlap_matrix** in the *control* file:

```
:TITLE
:--------------------------------------------------|
::C3H3NO   HF    6-31G
:--------------------------------------------------|

:KEYWORDS
:---------------------------------------------------
 basis=C3H3NO_HF_6-31G.g03

 overlap_matrix=C3H3NO_HF_6-31G_parallel.rho_r.bsn.sij   7

:CHOOSE THE DESIRED PROPERTIES
:-----------------------------------------
 compute=DAFH      alpha
:-----------------------------------------

 mesh=0.1  4.0

END
```

This would compute data grid for DAFH where one electron is confined to the oxygen density basin (number 7). The DAFH can be also averaged over a superbasins (using the correspond overlap matrix file for the superbasins, see previous section).

3.7.4 Fermi orbitals

The overlap integrals over basins used by the calculation of the domain averaged Fermi hole (DAHF) [34] can be also utilized to compute so called Fermi orbitals analysis of which was put forward by R. Ponec [35, 36]. In case of HF wavefunction the pair density can be written as:

$$\rho_2(\mathbf{r}_1\mathbf{r}_2) = \frac{1}{2}\left[\rho(\mathbf{r}_1)\rho(\mathbf{r}_2) - \sum_{i,j}\phi_i(\mathbf{r}_1)\phi_j^*(\mathbf{r}_1)\,\phi_j(\mathbf{r}_2)\phi_i^*(\mathbf{r}_2)\right] \tag{3.5}$$

Integrating the coordinate $\mathbf{r}_2$ over the region A (e.g., basin) yields:

$$\int_A \rho_2(\mathbf{r}_1\mathbf{r}_2)\mathrm{d}\mathbf{r}_2 = \frac{1}{2}\left[\rho(\mathbf{r}_1)q_A - \sum_{i,j}\phi_i(\mathbf{r}_1)\phi_j^*(\mathbf{r}_1)\,S_{ji}^A\right] \tag{3.6}$$

where q_A is the average population (charge) in the region A and S_{ji}^A the overlap integrals over A for the MOs j and i. The domain averaged Fermi-hole is the term:

$$\mathrm{DAFH} = \sum_{i,j} \phi_i(\mathbf{r}_1)\phi_j^*(\mathbf{r}_1)\, S_{ji}^A \tag{3.7}$$

The overlap matrix S_{ji}^A is symmetric and usually not diagonal. The diagonal elements S_{ii}^A are the orbital populations $\int_A |\phi_i|^2 \mathrm{d}r$ within A, i.e., the trace of S is the average population q_A of the basin A. DAFH integrated over the whole space yields the population q_A in the basin A. Thus, it can be regarded as kind of density distribution (different from the electron density) normalized to q_A. Actually, $(q_A\rho - \mathrm{DAFH})$ is proportional to twice the probability density to find an electron at $\mathbf{r}$ when the other electron is confined to the region A. One can seek for a 'natural' representation of the DAFH, i.e., such which is expressed by diagonal terms only. This is done by diagonalization of the overlap matrix S.

The diagonalization of S leaves the trace unchanged creating the eigenvectors φ_i. This orbitals, 'natural' with respect to DAFH, have the property of being mutually orthogonal within the basin A (this is what the S diagonalization is about). The diagonal elements S_{ii} – the eigenvalues – describe the 'contributions' of the rotated orbitals φ_i to the total population in basin A. The eigenvalues are the weighting factors of the squared orbitals to yield the DAFH density (formally like the 'natural occupations'). Each of the (spin) orbitals φ_i is of course occupied by single electron – it is still HF wavefunction, but with rotated orbitals – recovering the total electron density as$\rho = \sum \varphi_i^2$. Additionally, the integration of the DAFH in the 'natural' representation (using φ_i and the 'natural occupations' yields the localization index LI.

For a 'chemical' interpretation one would like to have the orbitals localized. The localization is accomplished with the isopycnic localization procedure of Cioslowski [16], which leaves the DAFH density unchanged and localizes the orbitals maximally into the basins. The resulting orbitals are called Fermi orbitals. It should be emphasized that after the isopycnic transformation the Fermi orbitals are no more orthogonal to each other within the basin and also do not recover the electron density by the sum of squared orbitals. The 'natural occupations' now describe the contribution of the orbital to the (domain averaged) hole density.

To compute a grid with the values for chosen Fermi orbital new *basis* file with the Fermi orbitals must be created. This is done with the command:

```
dgrid   name.bsn.sij   fermi   <basnum>
```

where *name.bsn.sij* is the overlap file described in previous sections (see the format description in Sec. 3.7.3) and <basnum> is the basin number (with the addition of the keyword `noisopyc` the resulting orbitals will not be localized, i.e., it will be just the eigenvectors after the diagonalization).

Let us use the data file *C3H3NO_HF_6-31G_parallel.rho_r.bsn.sij* created with the *control* file on page 69. The overlap integrals are computed over the density basins. We like to analyze the Fermi orbitals for the atomic basin of the oxygen which is the basin number 7 in the *basin* file *C3H3NO_HF_6-31G_parallel.rho_r.bsn* used to compute the overlap integrals. The command:

```
dgrid   C3H3NO_HF_6-31G_parallel.rho_r.bsn.sij   fermi   7
```

starts the whole procedure. The output is written to the console (or redirected into a file using '> outfile'). After the header data of the *basin* file the 'Fermi' section starts with the information about the location of the reference electron. The diagonalization of the overlap matrix is performed and the eigenvalues for the orbitals 'natural' with respect to the DAFH are written out (FO):

```
  Reference electron located in basin number 7   O1

                              +------------------------+
                              |       EIGENVALUES      |
                              +------------------------+

   FO        Alpha        CFO          Beta        CFO          Total        CFO
  --------------------   -----      -------------   -----      -------------   -----
    1    0.99999850774   1.0      0.99999850774    1.0      1.99999701549    2.0
    2    0.00000010180    -       0.00000010180     -       0.00000020361     -
    3    0.00012471637    -       0.00012471637     -       0.00024943275     -
    4    0.00006610157    -       0.00006610157     -       0.00013220314     -
    5    0.00000005342    -       0.00000005342     -       0.00000010684     -
    6    0.98820749800   1.0      0.98820749800    1.0      1.97641499599    2.0
    7    0.00041933464    -       0.00041933464     -       0.00083866928     -
    8    0.00000952153    -       0.00000952153     -       0.00001904305     -
    9    0.71775730515   0.7      0.71775730515    0.7      1.43551461030    1.4
   10    0.00017170490    -       0.00017170490     -       0.00034340980     -
   11    0.02333814790    -       0.02333814790     -       0.04667629580     -
   12    0.87901326439   0.9      0.87901326439    0.9      1.75802652879    1.8
   13    0.00675825176    -       0.00675825176     -       0.01351650352     -
   14    0.01483020045    -       0.01483020045     -       0.02966040089     -
   15    0.80581203071   0.8      0.80581203071    0.8      1.61162406141    1.6
   16    0.01373650791    -       0.01373650791     -       0.02747301581     -
   17    0.01589933704    -       0.01589933704     -       0.03179867409     -
   18    0.03460168832    -       0.03460168832     -       0.06920337663    0.1
  --------------------   -----      -------------   -----      -------------   -----
         4.50074427360   4.5      4.50074427360    4.5      9.00148854720    9.0
```

The column CFO shows the contributions of the rotated orbitals to the basin population rounded to 1 figure (contributions less than 0.1 are marked by period). It can be seen that the basin population of 9 electrons (i.e., negatively charged atomic oxygen basin) is described mainly by 5 orbitals (omitting the 0.1 contribution of FO number 18). This is followed by the information about the isopycnic localization and the resulting localized Fermi orbitals.

```
                               +--------------------------------------+
                               |   Isopycnic  localization procedure   |
                               +--------------------------------------+

                                      +------------------------+
                                      |       EIGENVALUES      |
                                      +------------------------+

 FO        Alpha        CFO           Beta        CFO          Total        CFO
--------------------   -----      -------------   -----      -------------   -----
  1    0.99999850758    1.0       0.99999850758    1.0       1.99999701516    2.0
  2    0.00001073593     -        0.00001073593     -        0.00002246516     -
  3    0.00014427145     -        0.00014427145     -        0.00022490045     -
  4    0.00012109153     -        0.00012109153     -        0.00023879084     -
  5    0.00000005342     -        0.00000005342     -        0.00000015526     -
  6    0.98818927341    1.0       0.98818927341    1.0       1.97637854683    2.0
  7    0.00044738968     -        0.00044738968     -        0.00089470383     -
  8    0.00009876346     -        0.00009876346     -        0.00000015524     -
  9    0.76482898291    0.8       0.76482898291    0.8       1.52965796581    1.5
 10    0.00000010182     -        0.00000010182     -        0.00026365299     -
 11    0.01342855438     -        0.01342855438     -        0.02685710724     -
 12    0.87901296688    0.9       0.87901296688    0.9       1.75802593376    1.8
 13    0.01539575171     -        0.01539575171     -        0.03079150402     -
 14    0.01638214194     -        0.01638214194     -        0.03276427094     -
 15    0.75874609563    0.8       0.75874609563    0.8       1.51749219126    1.5
 16    0.02810531017     -        0.02810531017     -        0.05621062035    0.1
 17    0.01560109694     -        0.01560109694     -        0.03120219859     -
 18    0.02023318474     -        0.02023318474     -        0.04046636949     -
--------------------   -----      -------------   -----      -------------   -----
       4.50074427360    4.5       4.50074427360    4.5       9.00148854720    9.0

==> Basis transformed to Fermi spinorbitals <==

==> GAUSSIAN03 data written to file C3H3NO_HF_6-31G.g03.FSO-7 <==
```

In this example the impact of the isopycnic localization on the eigenvalues is only marginal equalizing the contributions of the Fermi orbitals number 9 and 16 (from 1.4, 1.6 to 1.5, 1.5).

New *basis* file *C3H3NO_HF_6-31G.g03.FSO-7* is created with the 18 Fermi orbitals (the appendix '.FSO-7' shows that the basis describes Fermi orbitals for the basin number 7 (oxygen) after the Cioslowski localization. The grids for the Fermi orbitals (cf. page 28 in Sec. 2.6.4) can be now computed with the *control* file *fermi_orb.inp* from the *examples* directory:

```
::C3H3NO   HF    6-31G
:-------------------------------------------------------|
 basis=C3H3NO_HF_6-31G.g03.FSO-7

:CHOOSE THE DESIRED PROPERTIES
:------------------------------------------
 compute=phi    alpha  1
 compute=phi    alpha  6
 compute=phi    alpha  9
 compute=phi    alpha 12
 compute=phi    alpha 15
:------------------------------------------

 mesh=0.10   4.0   parallel
END
```

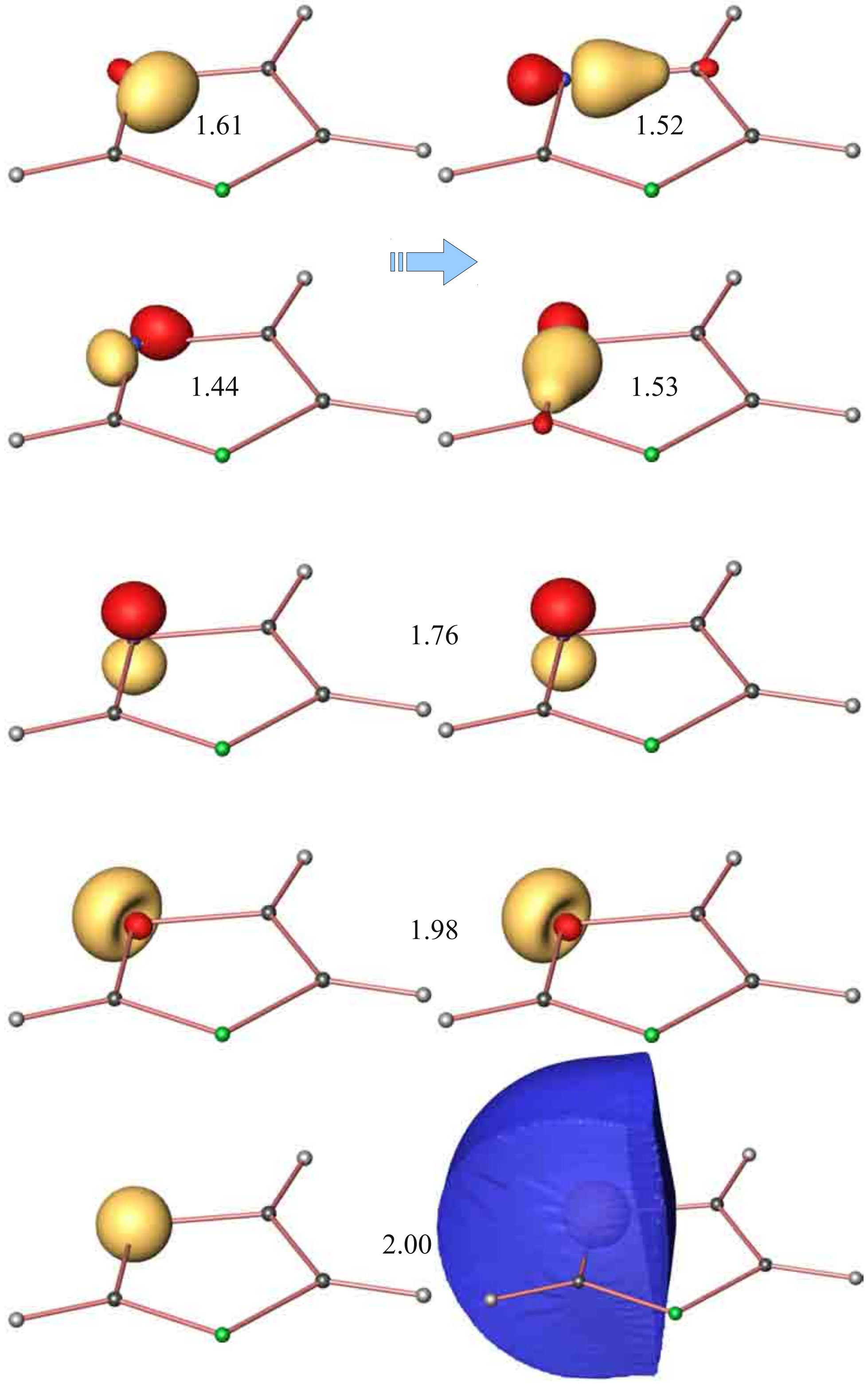

Fig. 3.8 Strongly contributing Fermi orbitals for C_3H_3NO from the overlap diagonalization for the oxygen atomic basin number 7 (blue colored). The numbers correspond to the contribution of the orbitals to the basin population (respectively to the hole density). Left: localization domains for the 'natural' orbitals (the isopycnic localization not active). Right: localization domains for the Fermi orbitals. Only the two uppermost orbitals are affected by the isopycnic localization

The resulting strongly contributing Fermi orbitals are visualized in Fig. 3.8 (right column). In the same Figure the orbitals determined by the eigenvectors of the overlap matrix diagonalization (without the isopycnic localization, i.e., with the keyword `noisopyc` added, when creating the orbitals) are shown in the left column. In the bottom right diagram the atomic basin (blue colored) of the oxygen is depicted.

The 2 lowermost orbitals in Fig. 3.8 are contributing with 2 electrons to the oxygen basin population. One of the orbitals can clearly be recognized as the oxygen core orbitals, the other as the oxygen 'lone' pair. The third orbital contributes by 1.78 electrons and can be attributed also to the oxygen 'lone' pair. The isopycnic transformation has almost no effect on these 3 orbitals (they are already properly localized within the oxygen basin). In contrast, the two uppermost orbitals are changed by the isopycnic localization. The resulting Fermi orbitals describe the two O-C bonds. They contribute by 1.52 to the hole density, showing the polarity of the bond (non polar bond would show Fermi orbital contribution close to 1).

The transformation to Fermi orbitals can be performed for open-shell systems as well [37]. In case of restricted basis it can be chosen to analyze the Fermi orbitals for the total hole density (similar to 'natural orbitals'), whereas for unrestricted basis only Fermi orbitals for each spin channel separately are available. In DGrid this possibilities are discriminated by the *basis* descriptor 'FO' and 'FSO' (meaning Fermi orbitals and Fermi spin-orbitals, respectively).

Using overlap integrals evaluated for natural orbitals changes somewhat the procedure. In this case the Fermi-hole can be approximated (pseudo-DAFH) by the sum including overlap integrals weighted by the square root of the product of natural occupations $\sqrt{n_i n_j}$ [18], expression which conserves the basin charge:

$$\text{pDAFH} = \sum_{i,j} \phi_i(\mathbf{r}_1)\phi_j^*(\mathbf{r}_1)\,\sqrt{n_i n_j} S_{ji}^A \tag{3.8}$$

3.8 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 calculation and examination of basins is performed by DGrid. However, there is a separate *control* file needed for the analysis.

The keywords are given either as variables, like e.g., '**property**=', or as standalone expressions, like '**end**'. Some of them are followed by 1 or more numbers, e.g., '**top**=0.8'. The parsing routine discriminates between floats and integer numbers. If integer number is given instead of float number 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 in separate line, because sometimes additional data are assumed per default. Some data even need to be written in rigorous sequence. For instance the symmetry oper-

Table 3.3 Keywords and parameters available for the basin evaluation

Keyword	Description	Value
attractors	write attractors to file	first, last (integers, default 1-50)
basin_mte=	multipole expansion parameters	lmax, gmax (default 30, 50.0)
close-atom_distance=	distance to nucleus	float number (default 0.1)
compactify	unify grid 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	
icl_graph=	create ICL graphs	saddle, ring, full
integrate=	integrate property	grid file name
localization_domains	search for localiz. domains	two isovalues (floats)
max_corrections	max. num. of corr. in basin search	integer
minima	write minima to file	use 'slices' for parallel calculation
output=	specify output file	output file name
overlap	overlap integrals over basins	
property=	scalar field defining basins	grid file name
reduced_step=	step for trajectories	step (float), default 0.04 bohr
result=	specify result file name	generic name of the property file
saddles	write saddle points to file	use 'slices' for parallel calculation
search_file=	file with critical points	file name
structure=	specify coordinates file name	file name with atomic coordinates
superbasin	set up superbasins	list terminated by superbasin_end
symmetry=	using symmetry	mirror, translation
top=	first cut	field value (`maximum`) (float)
topology	analyze critical points	

Default values are given in `Typewriter` font

ation during the basin search using the **symmetry** keyword. Table 3.3 summarizes the available keywords.

3.8.1 Attractors

With the keyword **attractors** the search for the basins is disabled. Instead, only the attractor positions will be searched and printed to the output when the *basis* file is accessible. If the *basis* file is unknown to DGrid, cf. page 48, then a separate file named *name.gridmax.str* will be created, containing the positions of the discrete grid maxima written in STR format (cf. Appendix A). Only first 50 local maxima will be printed (this is true also for the output into the *name.gridmax.str* file). To output another number of grid maxima use the command:

```
attractors    print=<from>    <to>
```

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

The attractors can be computed slice by slice and written to a file, which can be used in later evaluation. This procedure is accomplished with the command:

```
attractors    slices=<num>    <nr>
```

The total number of discrete maxima found on grid will be cut into <num> slices. The job will compute the slice number <nr>. For each slice the results of the search will be written on the file with '.search_max.slice-<nr>' appended to the generic *gridname*. The slice calculation can be done in parallel using the script `dgrid_para`, as described in Sec. 3.5.1.

After all slices are computed the slice files will be merged into single file with the DGrid utility 'complete':

```
dgrid   gridname.search_max.slice-1   complete
```

cf. page 100. This yields the file *gridname.search_max* that can be used in later evaluation (read into the program by the *control* file command **search_file=***filename*.

Using **slices=**1 will search all grid maxima in single run writing the results on the file *gridname.search_max*.

3.8.2 *Basin_mte*

Used for the calculation of overlap integrals over basins (respectively delocalization indexes) from the Elk [6] calculations. With the keyword **basin_mte** the expansion size for the multipole (<lmax>, integer value) and the Fourier expansions (<gmax>, float value) of basin shape are specified:

```
basin_mte=    <lmax>    <gmax>
```

The default values for <lmax> and <gmax> are 30 and 50.0, respectively. For QTAIM partitioning where the MT-spheres lie completely inside the basins, <lmax> can be reduced to `lmaxapw` used in Elk.

3.8.3 Close-atom_distance

The keyword **close-atom_distance** is followed by the distance <dist>. At the distance <dist> from the nucleus the search for a critical point stops. The default value is 0.01 bohr.

```
close-atom_distance=<dist>
```

The reason for this command is to avoid discontinuities or very large gradients close to the nucleus.

3.8.4 Compactify

With the keyword **compactify** all attractors (respectively the grid maxima) closer than <dist> (in units used for the grid definition) and differing in property value by less than <vdiff> will be assigned to the same basin, cf. page 50:

```
compactify    <dist>    <vdiff>
```

Especially if the wave function information is not present (i.e., without the *basis* file, as is usually the case for solid state calculations (other than Elk), there can be large number of discrete grid maxima. The keyword **compactify** enables a grouping of otherwise large number of (possibly artificial) basins. The default value for <dist> is twice the grid point distance. The default value for <vdiff> is the property range.

3.8.5 Core_charge

The keyword **core_charge** is used to invoke the inclusion of precomputed core charge into the integration procedure, see also page 50. The achieved integration precision is less than 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).

Remark 3.24. For Gauss-type orbitals and grids parallel with the Cartesian axes the analytical integration can be performed using the keyword **overlap**.

3.8.6 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 based on the field from <XY_prop> file with the isovalue <iso> (default is 0.001). The basins must be created during the run (not read in). As an example cf. the ρ-basins in Fig. 3.1 on page 47 cropped by the 0.001 bohr^{-3} isosurface of the oxazole electron density.

Remark 3.25. If the cropping procedure is not used then the outer basins for a molecular system extend to the borders of the computed region. Already computed basins can be conveniently cropped after the basin search using the DGrid utility 'crop', see Sec. 4.10.

Of course, with the cropping procedure part of the volume (outside the cropping isosurface) will not be considered for the integration. Consequently, 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 *rho_basins.inp* (from the *example* directory, see also page 46) in the corresponding output, cf. Sec. 3.4:

```
TOT   561.881   35.8460

Volume difference:  -1070.519
```

which shows the missing volume (the sum of the the total basin volume and the volume difference yields the total grid-box volume of 1632.4 bohr^3). The charge in the excluded region (1070.519 bohr^3) will not be considered by the integration routine.

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.

3.8.7 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 the inner core atomic shell, cf. page 50). Additionally, DGrid does not search for critical points in the core region defined as 0.75*ELI-D radius, cf. page 59.

3.8.8 End

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

3.8.9 Format

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

format=<fmt>

Table 3.4 Format descriptors

<fmt>		Format
cube	$\mapsto$	CUBE [2]
dgrid	$\mapsto$	DGrid *basin* file
grace	$\mapsto$	Grace [4]
lmto	$\mapsto$	TB-LMTO-ASA [26]

Remark 3.26. 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.

3.8.10 Gravity

Using the keyword **gravity** changes the output section, where the volumes of basins are given, cf. page 53. Then, the property values are given for the center of gravity of the corresponding basin.

3.8.11 Icl_graph

The keyword **gravity** can be used in a 'topology' run, cf. Sec. 3.5. After the critical points are found the routine searches for the interconnection lines (ICLs):

icl_graph=<icl>

Table 3.5 ICL descriptors

<icl>		Trajectories
saddle	$\mapsto$	from saddle to attractor
ring	$\mapsto$	from ring to minimum
full	$\mapsto$	like bcp and rcp

The routine creates the desired ICL graphs and writes the coordinates each 0.04 bohr to an output file *name.bsn.graph.str* using the STR format. Because the search for the critical points can be heavily time demanding (and the calculation of the trajectories as well) it is possible to perform the creation of the ICL graphs using a utility, cf. Sec. 4.7.

3.8.12 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 DGrid (dgrid, lmto, cube). If the **integrate** keyword is omitted, then only the basin volumes 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-box region will be determined.

3.8.13 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>.

3.8.14 Max_corrections

The basin search is done using Taylor expansions. Sometimes corrections with analytical Hessian are needed. This can be very time consuming. With the command:

max_corrections=<num>

the number of analytical corrections is restricted to <num>. After that again the expansions are used for the basin search.

3.8.15 Minima

The minima on the grid can be searched separately and written on a file using the command:

minima **slices=**<num> <nr>

The total number of discrete minima found on grid will be cut into <num> slices. The job will compute the slice number <nr>. For each slice the results of the search will be written on the file with '.search_min.slice-<nr>' appended to the generic *gridname*. The slice calculation can be done in parallel using the script `dgrid_para`, as described in Sec. 3.5.1.

After all slices are computed the slice files will be merged into single file with the DGrid utility 'complete', `dgrid` *gridname.search_min.slice-1* `complete`, cf. page 100. This yields the file *gridname.search_min* that can be used in later evaluation (read into the program by the *control* file command **search_file=***filename*.

Using **slices=**1 will search all grid minima in single run writing the results on the file *gridname.search_min*.

3.8.16 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.

3.8.17 Overlap

The keyword **overlap** can be used only for Gauss-type basis set. Additionally, the grid-box must be oriented parallel with the Cartesian axes (cf. keyword **mesh**, Sec. 2.6.13):

```
overlap
```

For each basin the overlap integrals $S_{ij} = \int_{\Omega} \phi_i \phi_j^* \mathrm{d}V$ will be computed and the tables written to a file (with string '.sij' appended; format see page 3.7.3) for further evaluation, cf. Secs. 3.7.4 and 3.7.3. Additionally, the localization and delocalization indices will be computed and written to the output, cf. Sec. 3.7.1.

3.8.18 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 DGrid (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).

3.8.19 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 default value for <step> is 0.02 bohr.

3.8.20 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.

3.8.21 Saddles

The saddle (and ring) points on the grid can be searched separately and written on a file using the command:

```
saddles    slices=<num>    <nr>
```

Each grid point is tested to be a candidate for a saddle point (using Taylor expansion to second order and then determining the rough position of the critical point). The total number of candidates found on grid will be cut into <num> slices. The job will compute the slice number <nr>. For each slice the results of the search will be written on the file with ‘.search_saddle.slice-<nr>’ appended to the generic *gridname*. The slice calculation can be done in parallel using the script `dgrid_para`, as described in Sec. 3.5.1.

After all slices are computed the slice files will be merged into single file with the DGrid utility ‘complete’, `dgrid` *gridname.search_saddle.slice-1* `complete`, cf. page 100. This yields the file *gridname.search_saddle* that can be used in later evaluation (read into the program by the *control* file command **search_file=***filename*.

Using **slices=**1 will search all grid minima in single run writing the results on the file *gridname.search_saddle*.

3.8.22 Search_file

The keyword **search_file** is followed by the name of the *search* file with results of the search for attractors, saddles, and minima, respectively. This can also be a UNIX path to this file.

search_file=<filename>

With this it is possible to test different strategies for the basin evaluation without waiting for the search for the critical points to finish.

3.8.23 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 Sec. A.6).

3.8.24 *Superbasin*

The keywords **superbasin** and **superbasin_end** (on separate lines) mark a block containing the definition of superbasins, one definition per line:

```
superbasin
<sup_i>=   <nr1>   <nr2>   ...
...
<sup_j>=   <nr3>   <nr4>   ...
superbasin_end
```

Each superbasin is given by a descriptor <sup_i> (any string). The equal sign is followed by a list of basin numbers <nr1> (known from previous basin job) forming the superbasins. For range of basin numbers the format <nr_from> - <nr_to> can be used. The superbasins must not overlap. The superbasin descriptor and the correspondence between the basins and superbasins will be written into the new *basin* file (with '.super' appended).

3.8.25 *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
```

Table 3.6 Symmetry descriptors

<sym>		Format
translation	↦	translate in given direction
mirror	↦	mirror at the start and end of the 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).

3.8.26 *Top*

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

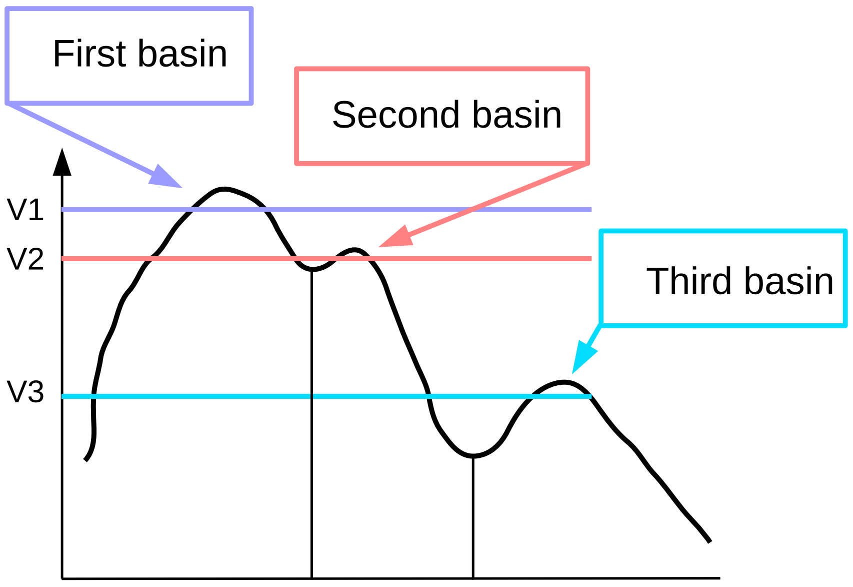

Fig. 3.9 Usage of the **top** keyword. The vertical lines mark the basin borders (saddle points)

top=<val>

Each attractor has its corresponding basin. However, if the *basis* file is absent (e.g. for solid state calculation) often larger number of discrete grid maxima are found around the attractor position. With the *top* value <val> DGrid '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. 3.9 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, when the basins have common scalar field isosurface, for superbasins see Sec. 3.6). 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.

3.8.27 Topology

Using the keyword **topology** invokes the topological analysis:

```
topology
```

It can be included in the *control* file either at the stage of the basin search or after the search, in which case the keyword **property** is followed by the *basin* file from previous run.

To perform the topological analysis the program needs to access the basins. This is due to the structure of the program. To invoke the search for the attractors as well as to get access to basin descriptors. For a detailed description of the results see Sec. 3.5. The topology section creates either a file (with the ending 'cp.str') in which the critical points are saved in the STR format or, if the search for the ICL graph was invoked with the **icl_graph** keyword, a file (with the ending 'graph.str') with the positions of the critical points as well as the ICL paths (in case of electron density – bond paths). The connection graph can be visualized with suitable interface (for the program Avizo see separate manual for the STR module). In case of the electron density evaluation the corresponding connection graph is called a molecular graph, cf. Fig. 3.3.

Chapter 4
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, *basin* file, or a slice file, respectively. The DGrid utilities work without the *control* file. Quick reference to the syntax is given by the command:

```
dgrid  -u
```

4.1 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 2.2, and must be created in advance from data supplied by quantum mechanical packages used.

4.1.1 ADF

In case of the program ADF [1] the formatted TAPE21 *filename.kf* (using the ‘dmpkf’ utility of ADF) 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
```

In this case the *basis* file will be named *filename.adf_loc*.

4.1.2 Gamess

The program Gamess [3] writes the basis and MO data to an output file. DGrid extracts the necessary data from that file using the command:

```
dgrid   filename.out
```

This creates the *basis* file named *filename.gms*.

Remark 4.1. The precision of the AO expansion coefficients in the Gamess output is low (format F11.6). To get more figures into the *basis* file the Gamess format has to be changed. To do this go to the installation directory of Gamess. There you will find the directory *source* with Gamess source code. Open the file *mthlib.src* and locate the subroutine PREVS. Change the format directives:

```
9028 FORMAT(15X,10(4X,I4,3X))
9048 FORMAT(I5,2X,A8,10F11.6)
9068 FORMAT(15X,10F11.4)
9078 FORMAT(16X,10(5X,A4,2X))
```

to:

```
9008 FORMAT(1X)
9028 FORMAT(15X,10(8X,I4,5X))
9048 FORMAT(I5,2X,A8,10F17.10)
9068 FORMAT(15X,10F17.10)
9078 FORMAT(16X,10(8X,A4,5X))
```

and recompile Gamess as usual.

DGrid will create the *basis* file also for the Gamess with the original format. However, with lower precision for the AO coefficients. To read from the Gamess output-file the natural orbitals use the command:

```
dgrid   filename.out natorb
```

Remark 4.2. The *basis* file can be also created from the WFN data included in the Gamess '.dat' file when the variable 'AIMPAC=.true.' was used in the Gamess input. The DGrid command is then:

```
dgrid   filename.dat wfn
```

4.1.3 Gaussian

In case of Gaussian [22] the formatted Checkpoint file *filename.fchk* (using the 'formchk' utility of Gaussian) must be convert into the *basis* file using the DGrid command:

```
dgrid   filename.fchk
```

yielding the *basis* file named *filename.g98*, respectively *filename.g03*. Another possibility is to use the information in the Gaussian WFN file:

```
dgrid   filename.wfn
```

yielding the *basis* file named *filename.gwf*.

Remark 4.3. In contrast to the *basis* file created from the Checkpoint file the *filename.gwf* does not contain virtual orbitals (as those are not present in the WFN file).

Remark 4.4. Even for a Gaussian calculation yielding at once both the Checkpoint and the WFN files the resulting *basis* files will be different! The reason is that in the WFN file the basis is already decontracted (and so will be the data in the corresponding *basis* file). In contrast, the *basis* file created from the Checkpoint file has contracted basis and the decontraction is performed internally during a DGrid job. However, the property values from a DGrid calculation will be the same, regardless which route is chosen.

Remark 4.5. IMPORTANT!
In case of ROHF (restricted open-shell) calculation Gaussian03 does not write properly the basis set data to the WFN file. All orbitals are doubly occupied in a WFN

file from such calculation, i.e., the information about the spin polarization is not given. After the conversion the occupation for the single occupied orbitals must be changed manually. The data in the WFN file from an UHF calculation are fine.

4.1.4 *Molpro, Molcas, Turbomole*

For the programs Molpro [47], Molcas [7], and Turbomole the basis data output must be written in the Molden [40] 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*, respectively *filename.mp*, *filename.mc* and *filename.tm* for the recognized programs. In case of the program Turbomole include in the Molden output file manually as the second line the string:

```
[TURBOMOLE]
```

otherwise the normalization will not be the proper one.

Remark 4.6. For natural orbitals from spin-polarized calculation the occupation is declared only as the α-spin channel. In the corresponding DGrid *basis* file the orbital occupations are declared as ‘natural’, i.e., there is only one number between 0-2 for each orbital. It is no more possible to extract the spin polarization from the natural occupation (for this the natural spinorbitals were needed, which result from diagonalization separate for each spin channel).

4.2 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.

4.3 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 A.6 in the Appendix A (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.

4.4 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 coefficients (the default value for the threshold is 0.1) and <occ> the threshold for the MO occupation (with the default value 0.0). For instance:

```
dgrid  C3H3NO_HF_6-31G.g03  0.05
```

prints out each molecular orbital from the *basis* file C3H3NO_HF_6-31G.g03 as a linear combination of atomic orbitals with coefficients of absolute value larger or equal 0.05 and MO occupation larger than 0.0 (which is the default value for <occ>):

```
******************************************************************************
AO expansion coefficients above the threshold of  0.0500    (occup >  0.0000)
******************************************************************************

input from  GAUSSIAN03      basis file  C3H3NO_HF_6-31G.g03

entitled:   C3H3NO HF 6-31G

Total energy =    -244.5027 hartree


--------------------------
CLOSED SHELL DATA
--------------------------

NONE  #  1:          occup = 2.00000000       Energy =     -20.6521

 0.9959  O  1 ( s       )

NONE  #  2:          occup = 2.00000000       Energy =     -15.5905

 0.9962  N  3 ( s       )
```

```
...
...

NONE  # 18:          occup = 2.00000000       Energy =      -0.3677

-0.0768  O  1 ( pz  )     -0.0722  O  1 ( pz_a )     -0.3026  C  2 ( pz  )     -0.2363  C  2 ( pz_a )
-0.1675  N  3 ( pz  )     -0.1454  N  3 ( pz_a )      0.3077  C  4 ( pz  )      0.2474  C  4 ( pz_a )
 0.3764  C  5 ( pz  )      0.3033  C  5 ( pz_a )
```

The virtual orbitals will not be printed for the default <occ> value. The printing of virtual orbitals can forced by any negative value for <occ> (for instance -1.0).

4.5 Completion of sliced fields

The calculation of grids as well as the determination of critical points can be performed in parallel by submitting slices of the fields to each processor (for grids cf. Sec. 2.5 and for the critical points cf. Sec. 3.5.1). The slice submition can be done with the script *dgrid_para*, cf. Sec. 2.5.2. The resulting data for each slice are saved in files named '*.slice-1', '*.slice-2', etc. The complete field is created with a DGrid utility using the slice number 1 as the first argument followed by the keyword `complete`:

```
dgrid   filename.slice-1   complete
```

Note, that all the remaining slice files must be present in the active directory. The resulting field will be saved in the file *filename*, i.e. with the slice numbering stripped off.

4.6 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 finer 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 of the integrated property. 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 *C3H3NO_HF_6-31G.g03.rho_r* (created in a DGrid job using the *control* file *rho.inp*) yields the total charge of 35.9901 electrons in the grid region (the total number of electrons for C_3H_3NO is 36). The result for this light molecule (and 0.05 bohr mesh) is close to the expected number, but the number of significant figures is certainly larger then the precision of the integration. Now, the electron density grid can be refined using the DGrid command:

```
dgrid  C3H3NO_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. =   0.09300

0                                               414
|----*----|----*----|----*----|----*----|----*----|
|>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>|

5796 values computed. Integrals written to file  C3H3NO_HF_6-31G.g03.rho_r.rfn

Sum of absolute changes =            0.0652327
```

It can be seen, that the DGrid refined the cells around 414 points computing 5796 additional values. The refinement data (refined integrals for grid points chosen by program) are saved in the file *C3H3NO_HF_6-31G.g03.rho_r.rfn* (i.e., the original file will not be changed). The subsequent integration using a *control* file with:

integrate = *C3H3NO_HF_6-31G.g03.rho_r.rfn*

yields the total charge of 35.9846 electrons in the grid region. This is a small decrease by 0.0055 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 65 173 points are refined by additional 1 201 286 points. Of course, the time demand increases adequately. The subsequent integration using the refined data gives 35.9807 electrons, i.e., total electron population lower by 0.0039 electron with respect to the 0.1 refinement.

The impact of the refinement procedure is much higher for functions with highly non-linear behavior, for instance for the electron density of heavier atoms or for the density Laplacian. With an appropriate *control* file the density Laplacian can be computed yielding the file *C3H3NO_HF_6-31G.g03.lap_rho_r* with data

grid of 0.1 bohr mesh size (and 4 bohr border). The integral of density Laplacian over the whole space equals zero. The integral over the computed region (ca. 1639 bohr^3 around the C_3H_3NO molecule) should only slightly deviate from zero. However, the numerical integration yields -123.0404 bohr^{-5}. The refinement of the density Laplacian grid with the precision parameter 1.0 decreases the integral to -0.0917 bohr^{-5} (computing additional 1 062 752 points).
Choosing the precision parameter 0.1 changes the total integral to the value of -0.0865 bohr^{-5} (computing 6 580 850 points). The change with respect to the 1.0 refinement is already very small ($\Delta = 0.0052$). Decreasing the precision parameter further to 0.01 yields the total integral of the density Laplacian of -0.0875 bohr^{-5} (i.e., lowering by 0.001) over the computed region (at the expense of computing the huge amount of additional 23 914 342 points). The exact value for the integral of the density Laplacian over the box region is of course not zero, because of the remaining small positive contribution outside the box.

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.

Remark 4.7. For electron density based on Gauss-type orbitals the overlap integrals over basins can be calculated (keyword **overlap**, cf. section 3.7.1). In the subsequent evaluation of the fluctuation also the electron population in the basins is computed analytically. The remaining error in the basin population is due only to the shape of the basin borders derived from a discrete grid (i.e., small cubes).

4.6.1 Evaluation of the Source Function

The source function contribution $S(r,\Omega)$ is defined as the integral of the local source $LS(r,r')$ [10, 24]

$$LS(r,r') = -\frac{1}{4\pi}\,\frac{\nabla^2\rho(r')}{|r-r'|} \tag{4.1}$$

over given region Ω

$$S(r,\Omega) = -\frac{1}{4\pi}\int_{\Omega}\frac{\nabla^2\rho(r')}{|r-r'|}\,dr' \tag{4.2}$$

It shows, how the region Ω 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 an example for the oxazole molecule:

```
:-----------------------------------------------|
::C3H3NO   HF    6-31G
:-----------------------------------------------|
 basis=C3H3NO_HF_6-31G.g03

 output=.

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

 ref_point= -1.4000    -1.0226     0.0000

 mesh=0.05   4.0

END
```

As the reference position usually a saddle point in the electron density is chosen. The above position is the electron density saddle point between the atoms O1-C5, cf. the saddle point Nr.8 in the output of the 'Topology' run on page 57, in Sec. 3.5. 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= -1.4000 -1.0226 0.0000   ref_density= 0.24458
```

Thus, the *grid* file contains 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 should recover 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 Gatti [10, 24], the source function contribution $S(r,\Omega)$, i.e., the local source integrals over the basins, are output by DGrid as percentage of source contribution to the density at the reference point.

Integrating the above local source field from the file *C3H3NO_HF_6-31G.g03.ls_r* yields value much larger than the density $\rho(r_b) = 0.24458$ bohr^{-3} at the saddle point. It gives 1531% (!) of the expected value. Refining the local source grid by the precision parameter 0.1 (computing additional 632 876 points) yields by integration 102.13% of the reference density. Increasing the precision with the refinement parameter 0.01 (evaluating 3 795 276 additional points) recovers by integration 100.54% of the reference density. Further increase of the precision (refinement parameter 0.001) does not significantly change the recovering of the density at the reference position (100.51% – 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 describes a test of the quality of the local source integration. 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 *C3H3NO_HF_6-31G.g03.rho_r* grid , cf. page 46 (0.05 bohr mesh size, cropped by 0.001 bohr^{-3} density isosurface), yielding the *basin* file *C3H3NO_HF_6-31G.g03.rho_r.bsn*. The *control* file for the source function contributions with the refined local source grid (refinement parameter 0.01) and the mentioned *basin* file reads as follows:

```
:-------------------------------------------------------|
::C3H3NO source function contribution over density basins
:-------------------------------------------------------|
 property =C3H3NO_HF_6-31G.g03.rho_r.bsn
 integrate=C3H3NO_HF_6-31G.g03.ls_r.rfn

 output=.

END
```

DGrid reads the basin data directly from the *basin* file without the time consuming basin search. The *C3H3NO_HF_6-31G.g03.ls_r.rfn* file contains not only the refinement data but also the name of the local-source 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   40.543    1.4631    0.4320  -0.303  4.067  0.000   H1    0.04   -
   2 R   41.243    5.2366    0.4289  -2.407 -3.341  0.000   H3    0.04   -
   3 R   42.713    1.7966    0.4315   2.780 -3.097  0.000   H2    0.04   -
   4 R   62.995    2.2435  118.3135   0.000  2.121 -0.000   C1    0.00   -
   5 R   78.962    6.5989  118.3130   1.413 -1.675 -0.000   C2    0.00   -
   6 R   79.534   42.6031  118.3147  -1.110 -1.860  0.000   C3    0.00   -
   7 R   99.674   41.2419  291.3237  -2.062  0.575  0.000   O1    0.00   -
   8 R  116.217    3.7847  192.1030   2.087  0.896 -0.000   N1    0.00   -
----------------------------------------------------------------------------
TOT     561.881  104.9685

Volume difference:  -1070.518
```

Each basin number is followed by an ‘R’ showing that the integrated data were refined for the basin. The volume difference at the end is a hint that the basins were cropped. Thus, part of the volume of the box-region was not considered. The impact of the cropping is that the recovering of the electron density at the reference position amounts to ca. 105%.

It can be seen that all the atomic basins are contributing as a source of the density at the saddle point between the oxygen and carbon nr. 3, with main contributions from the the neighboring basins (basins nr. 6 and 7).

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 for hours! Then it is possible to perform the refinement for local source in the desired basins only. In this case the command for the refinement includes the *basin* filename and reads as follows:

```
dgrid gridfile refine <prec> basinfile <bas1> <bas2> ...
```

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

As an example let us take the same data as before, but perform the source function evaluation for the basins number 7 and 8 (the oxygen and nitrogen atomic basins):

```
dgrid C3H3NO_HF_6-31G.g03.ls_r  refine 0.01
        ↪  C3H3NO_HF_6-31G.g03.rho_r.bsn  7  8
```

The information about the basins is included in the resulting refinement file. For the above refinement 826 616 additional points were computed (cf. the 3 795 276 points for the refinement with the same precision parameter computed previously in the total box-region). Taking closer look on the participation of basin number 8 (nitrogen) on the recovering of the saddle point electron density shows a change from 3.78% to 3.83%:

```
          Local-source[r]    rho[r]
BASIN    VOLUME  INTEGRAL   MAXIMUM    <X>    <Y>    <Z>    ATOMS DIST ECCNT
----------------------------------------------------------------------------
   1     40.542    1.4646    0.4320  -0.303  4.067  0.000   H1    0.04   -
   2     41.243    5.2395    0.4289  -2.407 -3.341  0.000   H3    0.04   -
   3     42.715    1.7981    0.4315   2.780 -3.097  0.000   H2    0.04   -
   4     62.999   50.6511  118.3135   0.000  2.121 -0.000   C1    0.00   -
   5     78.960   28.8798  118.3130   1.413 -1.675 -0.000   C2    0.00   -
   6     79.533 1416.7646  118.3147  -1.110 -1.860  0.000   C3    0.00   -
   7 R   99.673   41.2407  291.3237  -2.062  0.575  0.000   O1    0.00   -
   8 R  116.216    3.8260  192.1030   2.087  0.895 -0.000   N1    0.00   -
----------------------------------------------------------------------------
TOT     561.881 1549.8644
```

The reason for this change is that now the precision parameter is valid for the integral of just two basins, i.e., the relative precision is lower. For the integral of the whole box-region the precision is roughly 0.01 of 0.24 (saddle point density), i.e., ca. 4% error, whereas for the chosen basins (recovering around 45% of the saddle point density) it corresponds to 0.01 of ca. 0.11, i.e., around 9% error. The results are safe by using the precision parameter of 0.001 for both routes. Of course, this increases the amount of computed points.

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

4.7 ICL graph creation

Performing the search for critical points without the keyword **icl_graph**, cf. Sec. 3.5, yields a file (with the extension '.cp.str') with the coordinates of the critical points.

Using this file, the interconnection line (ICL) graph can be created using the command:

```
dgrid filename.cp.str <icl>
```

where <icl> is one of ‘saddle’, ‘ring’, or ‘full’. With ‘saddle’ the interconnection lines from saddle points to the attractors will be created. Using ‘ring’ creates the ICLs from ring critical points to the minima (cages), whereas ‘full’ creates both graphs in one run. The resulting data (the coordinates of the critical points as well as the steps along the ICLs) are written in the STR format to a file with ‘.graph.str’ appended.

4.8 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
  /         ˆ             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 ‘$+$, $-$, $\times$, $/$, ˆ, **over**’ need a number as a parameter in the input line. For instance the command ‘$\times$ 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 36. It is also possible to use this utility to convert files written with older DGrid versions into the current 4.5 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 [33].

4.9 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 possible operations <oper> are summarized in Table 4.1.

Table 4.1 Operations on two grids

<fmt>		Format	
$+$	$\mapsto$	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.

4.10 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 bohr^3 density 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 basin 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 with the information about the cropping 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. Sec. 3.8.6. The utility can be used if one do not want to perform a new basin search for different cropping isovalues.

4.11 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 [38]. Of course, not all basins of the second set will intersect the chosen basins. 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 with the basins of the second set are 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). Of course all the intersections together have the same volume as the intersected basin.

For example, for the C_3H_3NO molecule two grids for the electron density and ELI-D, respective, will be computed using the following *control* file (with the Gaussian03 *basis* file *C3H3NO_HF_6-31G.g03* from the *example* directory):

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

:KEYWORDS
:----------------------------------------------
 basis=C3H3NO_HF_6-31G.g03

 compute=rho
 compute=ELI-D     triplet

 mesh=0.05  4.0

END
```

Two grid files will be written, namely *C3H3NO_HF_6-31G.g03.rho_r* for the electron density and *C3H3NO_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):

```
::C3H3NO basins
:--------------------------------------------------|

:KEYWORDS
:---------------------------------------------------
 property=C3H3NO_HF_6-31G.g03.rho_r
 crop    =C3H3NO_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 *C3H3NO_HF_6-31G.g03.rho_r.bsn* with 8 electron density basins, respectively the file *C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn* describing 15 ELI-D basins.

According to the QTAIM approach the electron density basins, cf. Fig. 3.1 on page 47, describe the atoms in molecule [9]. Because non-nuclear maxima are not present in case of the C_3H_3NO 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 atoms (which can be assumed as bond descriptors) and around the oxygen and nitrogen core basins (lone-pair descriptors), cf. Fig. 3.2 on page 55.

There are always two possibilities for the intersection analysis depending on the choice of the intersecting set (*A* intersect *B*, or *B* intersect *A*). 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  6
          ↪  C3H6_HF_6-31G.g03.rho_r.bsn
```

the ELI-D basin number 6, which corresponds to the ‘bond’ basin between the oxygen and carbon C3, will be intersected by the electron density basins. 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. 6 (O1-C5):

 rho[r]
 basin  descriptor      volume           %
-------------------------------------------
   6           C3       0.939        17.19
   7           O1       4.523        82.82
-------------------------------------------
                        5.462       100.00

Basin intersections written to file
-> C3H3NO_HF_6-31G.g03.rho_r.bsn.isect_C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn-6
```

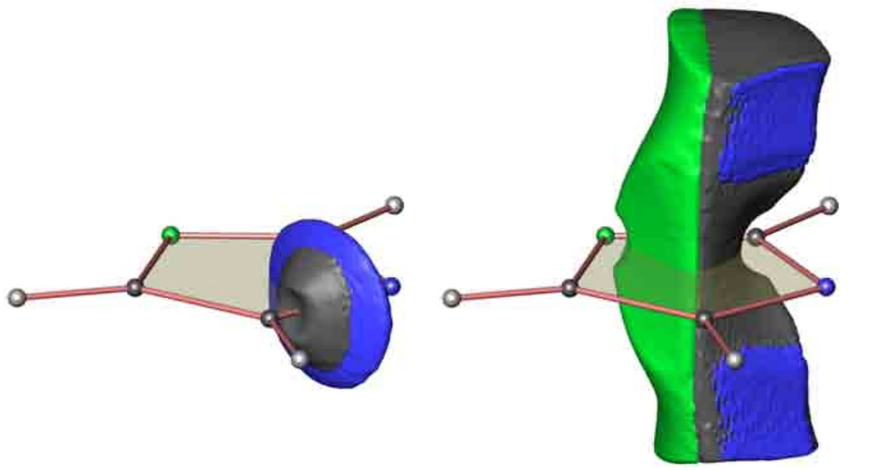

Fig. 4.1 ELI-D basins (cropped by the 0.001 bohr^{-3} isosurface of electron density) of C_3H_3NO intersected by electron density basins. Left: intersection of ELI-D bond basin O1-C3. Right: intersection of ELI-D bond basin N1-C1. The color coding is blue (oxygen), green (nitrogen), black (carbon), gray (hydrogen)

The result shows that the ELI-D basin number 6 (O1-C3 bond) is intersected by two density basins, namely the ones corresponding to the atoms O1 and C3, respectively, cf. the left diagram in Fig. 4.1. The density basin of the oxygen (the oxygen 'atom') participate to almost 83% on the ELI-D basin volume. The resulting basin intersections are saved in a *basin* file (with the long name documenting that the density basins intersected the ELI-D basin number 6), which actually contains the density basins within the region of ELI-D basin nr. 6. The file can be used for the integration of the electron density to yield the charges within each intersection. The intersection charges can be connected with the bond polarity [38]. In the case of the above intersection the oxygen contribute with 84.8% to the charge in the O1-C3 ELI-D bond basin, whereas the C3 contribution amounts to 15.2% only ($1.01 + 0.18 = 1.19$ electrons total). Thus, the bond can be regarded as polar, with the oxygen as the more negative bonding partner.

Intersecting the ELI-D basin number 9, which can be assigned to the N1-C1 bond, with the electron density basins of the oxazole yields somewhat different picture:

```
Intersections of ELI-D[r] basin No. 9 (N1-C1):

 rho[r]
 basin  descriptor      volume           %
-------------------------------------------
   7            O1       1.139        2.80
   8            N1      15.341       37.70
   4            C1      24.217       59.51
-------------------------------------------
                        40.697      100.00

Basin intersections written to file
```

```
  -> C3H3NO_HF_6-31G.g03.rho_r.bsn.isect_C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn-9
```

Now, cf. also the right diagram in Fig. 4.1, the larger volume contribution of 59.5% is given for the carbon density basin. Nevertheless, the integration of the electron density within the intersections yields more charge (58%) for the nitrogen density basin then for the carbon basin (1.64 electrons for N and 1.15 electrons for C). The N-C bond can be regarded as slightly polar with the nitrogen as the more negative participant.

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

`dgrid` *C3H3NO_HF_6-31G.g03.rho_r.bsn* `intersect` 7
$\hookrightarrow$ *C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn*

the electron density basin number 7, which corresponds to the basin of the oxygen atom, is intersected by the ELI-D basins. On the console the following output appears:

```
Intersections of rho[r] basin No. 7 (O1):

 ELI-D[r]
 basin  descriptor      volume           %
-------------------------------------------
   1            O1       0.268        0.27
  10            H3       0.706        0.71
  12            H1       0.909        0.91
   9         N1-C1       1.139        1.14
  15         C3-C2       1.716        1.72
   6         O1-C3       4.523        4.54
   7         O1-C1       4.689        4.70
  13         LP_O1      85.722       86.00
-------------------------------------------
                        99.674      100.00

Basin intersections written to file
  -> C3H3NO_HF_6-31G.g03.elid_r_t_tr.bsn.isect_C3H3NO_HF_6-31G.g03.rho_r.bsn-7
```

The oxygen density basin (QTAIM basin) is intersected mainly by 3 ELI-D basins, with the largest contribution of the ELI-D oxygen lone-pair (86%, marked LP). The two other contributions of roughly 5% are the intersections of the density basin with the respective O-C ELI-D bond basins. Observe that the ELI-D core basin of the oxygen take up just a small fraction (ca. 0.3%) of the QTAIM basin volume (however, containing 2.1 electrons). The 3 intersections are shown in Fig. 4.2.

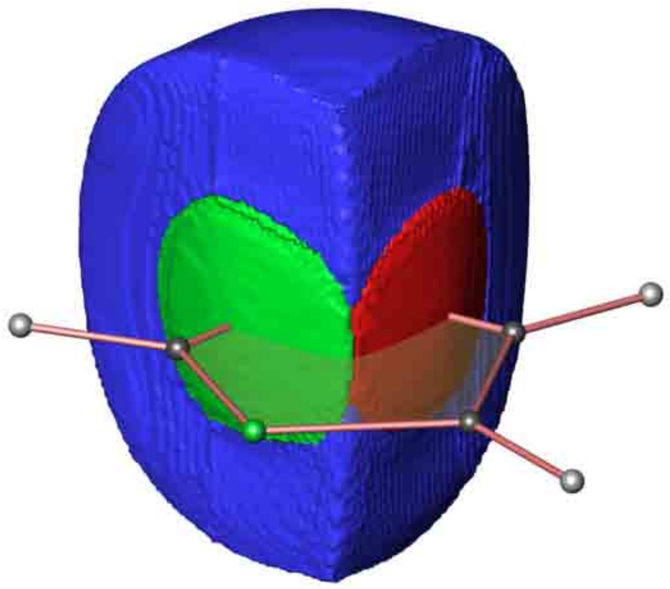

Fig. 4.2 Electron density basin (cropped by the 0.001 bohr^{-3} isosurface of electron density) for the oxygen in the C_3H_3NO molecule intersected by ELI-D basins. The blue intersection correspond to oxygen lone-pair, the red and green intersections are due the respective C-O bond basins

Appendix A
Comments to property calculations

A.1 Evaluation of Elk calculations

To evaluate Elk [6] results the input file ‘elk.in’ (the DGrid conversion routine must be called with this file name) needs to be converted. The name for the converted file (in wide sense corresponding to a *basis* file) should be given as a second argument, for instance:

```
dgrid   elk.in   Al.mte
```

The generated file (‘.mte’ file) contains the expansions of charge density and kinetic energy density together with their derivatives. This allows fast evaluation of many properties so that the ‘.mte’ file is sufficient to run the tasks. However, some tasks, like the evaluation of delocalization indices or projected properties require an access to the wavefunction which is not saved in the ‘.mte’ file. In this case additional files from the Elk calculation should be available.

A.1.1 Overlap integrals and delocalization indexes

For the calculation of overlap integrals over basins (needed e.g. for the examination of delocalization indexes) the expansion size for the multipole (<lmax>, integer) and the Fourier expansions (<gmax>, float) of basin shape can be specified with the keyword **basin_mte**= <lmax> <gmax> in the *control* file. If not given then the default values <lmax>=30 and <gmax>=50.0 will be used. For QTAIM partitioning where the MT-spheres lie completely inside basins, <lmax> can be reduced to `lmaxapw` used in Elk.

A.1.2 *Projected properties*

For properties computed from k-states within certain energy window the energy range be specified in the *control* file using the keyword **energy_window**= <emin> <emax>. The energy values must be given in a.u. relative to the Fermi level. Note that only the valence states in terms of Elk can be projected, not the core states.

The choice of particular k-state is done by the number of k-point with the appended *k*-point index as a state symbol (e.g., 'k3'). State index is then the number of the band chosen.

A.2 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 numerical 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 4.6. This approach is possible only if the *basis* file is present (i.e., not possible for solid state calculations, except with Elk[6], see Sec. A.1). For Gauss-type basis set analytical charge integration can be used, cf. Sec. 3.7.

A.3 Spin density

For the spin density, computed with the assignment **compute**=rho-spin, cf. Table 2 on page 17, both spin parts are needed. Thus, the spin density will be not computed if a particular spin part is chosen. Note that with natural occupation the spin density cannot be determined as there is no spin information (the occupation is between 0-2). To compute the spin density the corresponding natural spinorbitals are needed.

A.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 [29]. In each respective cases the electron density of corresponding spin will be sampled.

ELI-D for triplet coupled electron pairs 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. [33].

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. [33], which is just a way to show an ELI-D value.

The other variant of ELI, namely ELI-q (cf. Ref. [33]), 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. [30].

A.5 ELF, LOL

The assignments ‘**compute**=ELF alpha’, respectively ‘**compute**=ELF beta’, result in the calculation of ELF according the formula of Becke and Edgecombe [15]. 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. [31]. 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. [31]. 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 [33].

With the assignment ‘**compute**=ELF-Kirzhnits’ (with or without the spin descriptor) the ELF variant of Tsirelson [44] is computed. It is based on kinetic energy density approximated by the Kirzhnits 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 [41].

A.6 STR format

The STR format is used to support structure information for visualization. The data are written in sections into a file. The data start with the sections, termed ‘UNITS’, ‘GRID_DESCRIPTION’, and ‘SCALE’:

```
UNITS
BOHR             1

GRID_DESCRIPTION
FILENAME         C3H3NO_HF_6-31G.g03.elid_r_t_tr
ELI_CORE         1

SCALE
ATOMIC_RADIUS    1.0
BOND_DIAMETER    0.5
BOND_MIN         0.1
BOND_MAX         5.0
PATH_DIAMETER    0.5
```

The ‘GRID_DESCRIPTION’ section is used for the ICL graphs. The data can be evaluated by chosen visualization tool (and are actually used by the module written for the programs Avizo and OpenDX). This header is followed by one or more sections called:

• ATOMS	$\mapsto$	spheres
• CONNECTIONS	$\mapsto$	cylinders
• UNIT_CELL	$\mapsto$	unit cell
• POLYHEDRA	$\mapsto$	polyhedra and polygons
• PATHS	$\mapsto$	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’.

A.6.1 ATOMS

In this section the data for spheres representing atoms, maxima, saddle points, etc. are given. Following is an example for the C_3H_3NO molecule, written out by the Avizo visualization program:

```
ATOMS
:atom      radius      x          y          z     vis
:-----------------------------------------------------
 O_1        0.551   -2.0618    0.5748    0.0000   1
 C_2        0.451    0.0000    2.1211   -0.0000   1
 N_3        0.503    2.0869    0.8955   -0.0000   1
 C_4        0.451    1.4135   -1.6748   -0.0000   1
 C_5        0.451   -1.1101   -1.8596    0.0000   1
 H_6        0.095   -0.3077    4.1025   -0.0000   1
 H_7        0.095    2.8049   -3.1219   -0.0000   1
 H_8        0.095   -2.4310   -3.3680    0.0000   1

ATOM_BUTTONS
:atom      r       g       b
:----------------------------
 O      0.2300  0.2300  0.9700
 C      0.3300  0.3300  0.3300
 N      0.0000  0.8400  0.1700
 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, which is the case if the structure file is created with the command '`dgrid` *basisfile* `str`' (cf. Sec. 4.3) then the visualization tool must choose its own default values.

A.6.2 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
:-----------------------------------------------------------------------------------
 O-C_1-2      0.100  -2.0618    0.5748   0.0000    0.0000   2.1211  -0.0000   1  1
 O-C_1-5      0.100  -2.0618    0.5748   0.0000   -1.1101  -1.8596   0.0000   1  1
 C-N_2-3      0.100   0.0000    2.1211  -0.0000    2.0869   0.8955  -0.0000   1  1
 C-H_2-6      0.100   0.0000    2.1211  -0.0000   -0.3077   4.1025  -0.0000   1  1
 N-C_3-4      0.100   2.0869    0.8955  -0.0000    1.4135  -1.6748  -0.0000   1  1
 C-C_4-5      0.100   1.4135   -1.6748  -0.0000   -1.1101  -1.8596   0.0000   1  1
 C-H_4-7      0.100   1.4135   -1.6748  -0.0000    2.8049  -3.1219  -0.0000   1  1
 C-H_5-8      0.100  -1.1101   -1.8596   0.0000   -2.4310  -3.3680   0.0000   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.

A.6.3 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.

A.6.4 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    0.6000

  FACE vertices:   5
     -2.0618    0.5748    0.0000
      0.0000    2.1211   -0.0000
      2.0869    0.8955   -0.0000
      1.4135   -1.6748   -0.0000
     -1.1101   -1.8596    0.0000

POLYHEDRA_BUTTONS
:col      r       g       b
:-----------------------------
  1:   0.9220  0.8980  0.4750
```

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 shininess 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 pentagon formed by the ring atoms of the oxazole molecule).

A.6.5 PATHS

In this section the data for the ICL trajectories running from the saddle point to the corresponding attractor (respective from ring critical points to the minima) are given (cf. Sec. 3.5):

```
PATHS
:        name     points   vis col  diameter
:-------------------------------------------
Path     B1-A2      36      1   1      0.02

:         x         y         z
:---------------------------------
        0.7730  1.624593  0.000000
        0.7908  1.615388  0.000000
                  ...
        2.0833  0.897110  0.000000

:        name     points   vis col  diameter
:-------------------------------------------
Path     B1-A4      21      1   1      0.02

:         x         y         z
:---------------------------------
        0.7730  1.624593  0.000000
        0.7553  1.633799  0.000000
                  ...
        0.0045  2.117934  0.000000


PATH_BUTTONS
:col      r       g       b
:----------------------------
  1:   1.0000  0.5000  0.0000
  2:   0.0000  0.7500  1.0000
  3:   0.7500  1.0000  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 ‘B1-A2’ stands for the path from the saddle point number 1 (which is between the nitrogen and carbon C2 ρ-basins, cf. the table on page 57) to the attractor number 2 (the nitrogen, cf. the table on page 57). If ICL graph for the ring points is present, then the path descriptors start with ‘R’ instead of ‘B’. Additionally, color number 2 is chosen for this paths.

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.

A.6.6 END

The STR format is terminated by the string ‘END’.

And this is also the end of the User’s Guide

References

1. ADF (density functional program). 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
2. Cube format. http://astronomy.swin.edu.au/∼pbourke/geomformats/cube/
3. GAMESS, general atomic and molecular electronic structure system. Schmidt MW, Baldridge KK, Boatz JA, Elbert ST, Gordon MS, Jensen JH, Koseki S, Matsunaga N, Nguyen KA, Su S, Windus TL, Dupuis M, Montgomery J (1993) J. Comput. Chem. 14: 1347
4. Grace. http://plasma-gate.weizmann.ac.il/Grace/
5. Turbomole. Ahlrichs R, Bär M, Häser M, Horn H, Kölmel C (1989) Chem. Phys. Lett. 162: 165
6. Elk 1.2.20 (2011). http://elk.sourceforge.net
7. Andersson, K., Barysz, M., Bernhardsson, A., Blomberg, M.R.A., Cooper, D.L., Fülscher, M.P., de Graaf, C., Hess, B.A., Karlström, G., Lindh, R., Malmqvist, P.A., Nakajima, T., Neogrády, P., Olsen, J., Roos, B.O., Schimmelpfennig, B., Schütz, M., Seijo, L., Serrano-Andr'es, L., Siegbahn, P.E.M., Stålring, J., Thorsteinsson, T., Veryazov, V., Widmark, P.O.: MOLCAS. Lund University, Sweden
8. Ángyán, J.G., Loos, M., Mayer, I.: Covalent bond orders and atomic valence indexes in the topological theory of atoms in molecules. J. Phys. Chem. **98**, 5244–5248 (1994)
9. Bader, R.F.W.: Atoms in molecules. Oxford University Press, Oxford (1990)
10. Bader, R.F.W., Gatti, C.: A Green's function for the density. Chem. Phys. Lett. **287**, 233 (1998)
11. Bader, R.F.W., Stephens, M.E.: Fluctuation and correlation of electrons in molecular systems. Chem. Phys. Lett. **25**, 445–449 (1974)
12. Bader, R.F.W., Stephens, M.E.: Spatial localization of the electronic pair and number distributions in molecules. J. A. Chem. Soc. **97**, 7391–7399 (1975)
13. Baranow, A.I., Kohout, M.: Topological analysis of real space properties for the solid-state full-potential apw dft method. J. Phys. Chem. Solids **71**, 1350–1356 (2010)
14. Baranow, A.I., Kohout, M.: Localization and delocalization indices in solids. J. Comput. Chem. **–**, 1–1 (2011)
15. Becke, A.D., Edgecombe, K.E.: A simple measure of electron localization in atomic and molecular systems. J. Chem. Phys. **92**, 5397 (1990)
16. Cioslowski, J.: Isopycnic orbital transformations and localization of natural orbitals. Int. J. Quantum Chem. Symp. **24**, 15–28 (1990)
17. Clementi, E., Roetti, C.: At. Data. Nucl. Data Tables **14**, 218 (1974)
18. Cooper, D.L., Ponec, R.: A one-electron approximation to domain-averaged fermi hole analysis. Phys. Chem. Chem. Phys. **10**, 1319–1329 (2008)
19. Cremer, D., Kraka, E.: A description of the chemical bond in terms of local properties of electron density and energy. Croat. Chem. Acta **57**, 1259–1281 (1984)

20. Fradera, X., Austen, M.A., Bader, R.F.W.: The lewis model and beyond. J. Phys. Chem. A **103**, 304–314 (1999)
21. Fradera, X., Poater, J., Simon, S., Duran, M., Solà, M.: Electron-pairing analysis from localization and delocalization indices in the framework of the atom-in-molecules theory. Theor. Chem. Acc. **108**, 214–224 (2002)
22. Frisch, M.J., Trucks, G.W., Schlegel, H.B., Scuseria, G.E., Robb, M.A., Cheeseman, J.R., Zakrzewski, V.G., Montgomery, J.A.J., Stratmann, R.E., Burant, J.C., Dapprich, S., Millam, J.M., Daniels, A.D., Kudin, K.N., Strain, M.C., Farkas, O., Tomasi, J., Barone, V., Cossi, M., Cammi, R., Mennucci, B., Pomelli, C., Adamo, C., Clifford, S., Ochterski, J., Petersson, G.A., Ayala, P.Y., Cui, Q., Morokuma, K., Malick, D.K., Rabuck, A.D., Raghavachari, K., Foresman, J.B., Cioslowski, J., Ortiz, J.V., Baboul, A.G., Stefanov, B.B., Liu, G., Liashenko, A., Piskorz, P., Komaromi, I., Gomperts, R., Martin, R.L., Fox, D.J., Keith, T., Al-Laham, M.A., Peng, C.Y., Nanayakkara, A., Gonzalez, C., Challacombe, M., Gill, P.M.W., Johnson, B., Chen, W., Wong, M.W., Andres, J.L., Gonzalez, C., Head-Gordon, M., Replogle, E.S., Pople, J.A.: Gaussian 03. Gaussian Inc., Wallingford CT (2004)
23. Fulton, R.L.: Sharing of electrons in molecules. J. Phys. Chem. **97**, 7516–7529 (1993)
24. Gatti, C., Cargnoni, F., Bertini, L.: Chemical information from the source function. J. Comp. Chem. **24**, 422 (2002)
25. Hunter, G.: The exact one-electron model of molecular structure. Int. J. Quantum Chem. **29**, 197 (1986)
26. Jepsen, O., Andersen, O.K.: The stuttgart TB-LMTO-ASA program, version 4.7. Max-Planck-Institut für Festkörperforschung, Stuttgart (2000)
27. Kohout, M.: A measure of electron localizability. Int. J. Quantum Chem. **97**, 651–658 (2004)
28. Kohout, M.: Bonding indicators from electron pair density. Faraday Discuss. **135**, 43–54 (2007)
29. Kohout, M., Pernal, K., Wagner, F.R., Grin, Y.: Electron localizability indicator for correlated wavefunctions. I parallel-spin pairs. Theor. Chem. Acc. **112**, 453–459 (2004)
30. Kohout, M., Pernal, K., Wagner, F.R., Grin, Y.: Electron localizability indicator for correlated wavefunctions. II antiparallel-spin pairs. Theor. Chem. Acc. **113**, 287–293 (2005)
31. Kohout, M., Savin, A.: Atomic shell structure and electron numbers. Int. J. Quantum Chem. **60**, 875–882 (1996)
32. Kohout, M., Wagner, F.R., Grin, Y.: Electron localization function for transition-metal compounds. Theor. Chem. Acc. **108**, 150–156 (2002)
33. Kohout, M., Wagner, F.R., Grin, Y.: Electron localizability indicator for correlated wavefunctions. III: singlet and triplet pairs. Theor. Chem. Acc. **119**, 413–420 (2008)
34. Ponec, R.: Electron pairing and chemical bonds. chemical structure, valences and structural similarities from the analysis of the fermi holes. J. Math. Chem. **21**, 323–333 (1997)
35. Ponec, R.: Electron pairing and chemical bonds. molecular structure from the analysis of pair densities and related quantities. J. Math. Chem. **23**, 85–103 (1998)
36. Ponec, R., Duben, A.J.: Electron pairing and chemical bonds: bonding in hypervalent molecules from analysis of fermi holes. J. Comput. Chem. **20**, 760–771 (1999)
37. Ponec, R., Feixas, F.: Domain averaged fermi hole analysis for open-shell systems. J. Phys. Chem. A **113**, 5773–5779 (2009)
38. Raub, S., Jansen, G.: A quantitative measure of bond polarity from the electron localization function and the theory of atoms in molecules. Theor. Chem. Acc. **106**, 223–232 (2001)
39. Savin, A., Jepsen, O., Flad, J., Andersen, O.K., H., P., von Schnering, H.G.: Electron localization in solid-state structures of the elements: the diamond structure. Angew. Chem. Int. Ed. Engl. **31**, 187 (1992)
40. Schaftenaar, G., Noordik, J.H.: Molden: a pre- and post-processing program for molecular and electronic structures. J. Comput.-Aided Mol. Design **14**, 123 (2000)
41. Schmider, H.L., Becke, A.D.: Chemical content of the kinetic energy density. J. Mol. Struct. (Theochem) **527**, 51 (2000)
42. Shaffer, A., Wierschke, S.G.: Comparison of computational methods applied to oxazole, thiazole, and other heterocyclic compounds. J. Comp. Chem. **14**, 75–88 (1993)

43. Silvi, B., Savin, A.: Classification of chemical bonds based on topological analysis of electron localization function. Nature **371**, 683 (1994)
44. Tsirelson, V., Stash, A.: Determination of the electron localization function from electron density. Chem. Phys. Lett. **351**, 142 (2002)
45. Wagner, F.R., Bezugly, V., Kohout, M., Grin, Y.: Charge decomposition analysis of the electron localizability indicator: A bridge between the orbital and direct space representation of the chemical bond. Chem. Eur. J. **13**, 5724–5741 (2007)
46. Wagner, F.R., Kohout, M., Grin, Y.: Direct space decomposition of eli-d: Interplay of charge density and pair-volume function for different bonding situations. J. Phys. Chem. A **112**, 9814–9828 (2008)
47. Werner, H.J., Knowles, P.J., Amos, R.D., Bernhardsson, A., Berning, A., Celani, P., Cooper, D.L., Deegan, M.J.O., Dobbyn, A.J., Eckert, F., Hampel, C., Hetzer, G., Korona, T., Lindh, R., Lloyd, A.W., McNicholas, S.J., Manby, F.R., Meyer, W., Mura, M.E., Nicklass, A., Palmieri, P., Pitzer, R., Rauhut, G., Schütz, M., Schumann, U., Stoll, H., Stone, A.J., Tarroni, R., Thorsteinsson, T.: MOLPRO, a package of ab initio programs

Index