Scientific Computing & Modelling

Analysis Programs

ADF Program System
Release 2010

Scientific Computing & Modelling NV
Vrije Universiteit, Theoretical Chemistry
De Boelelaan 1083; 1081 HV Amsterdam; The Netherlands
E-mail: support@scm.com

Copyright © 1993-2010: SCM / Vrije Universiteit, Theoretical Chemistry, Amsterdam, The Netherlands
All rights reserved

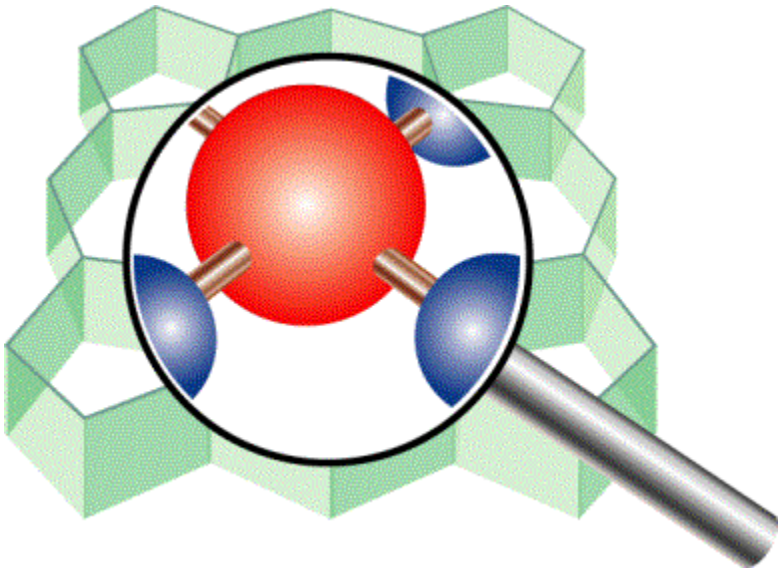

Table of Contents

Analysis Programs **1**
Table of Contents **2**
Introduction **3**
Densf: Volume Maps **4**
Input **4**
Input/Output files 4
Grid 5
Inline Grid 6
Units 6
Density 7
Kinetic Energy Density and Electron Localization Function (ELF) 7
Laplacian of the Density 7
Gradient of the Density 8
Hessian of the Density 8
Potential 8
Orbitals 9
NOCV 10
Generic orbitals 11
Result: TAPE41 ..... **12**
Sections on TAPE41 12
Notes 13
Contents of TAPE41 13
Cntrs: Contour Plots **16**
Input **16**
Result **17**
ADFPLT: GUI for volume Maps **18**
Dos: Density of States **19**
Introduction **19**
Mulliken population analysis **20**
Density of states analyses based on Mulliken population analysis **21**
Generalizations of OPDOS, GPDOS, PDOS **21**
Input **22**
Energy scan values 22
Peak widening 23
Result files 23
Densities of States 23
Adfnbo, gennbo: NBO analysis **25**
NBO analysis of EFG, NMR chemical shifts, NMR spin-spin coupling **25**
EFG 26
NMR Chemical shift 26
NMR spin-spin coupling (J-coupling) 28
Generation of NBOs 29
Adf2aim: make input for Xaim (Bader's analysis) **30**

Introduction

The ADF package contains two main programs; ADF and BAND for calculations on, respectively, molecules and periodic systems and a suite of auxiliary utility and property programs, which are used to prepare special input data or to process results. This manual describes the usage of some analysis programs. Consult also the *User's Guide*. You should be familiar with the *User's Guide* anyway before reading this *Analysis* document because some concepts from the *User's Guide* will be assumed known here. Among these are the use, format, and general types of keywords applied in the input files as well as in the input for some of the analysis programs. Most of the analysis programs relate only to ADF, in the current version of the package.

The analysis programs are installed automatically when the ADF package is installed. See the *Installation Manual*.

Some of the files that will be mentioned are KF files. A KF file (Keyed-File) is a special type of Direct-Access Fortran file used in programs of the ADF suite. They have a keyword driven organization and can easily be processed with the KF-utilities that come with the ADF package.

Overview

The analysis programs discussed in this manual are:

- *densf*: a program to generate values of the charge density, potential, molecular orbitals in a user-specified regular grid (2-D or 3-D).
- *cntrs*: a program to generate contours for data computed by *densf*.
- *adfplt*: a program to graphically display orbitals, densities or potentials computed in a 2D or 3D grid.
- *dos*: a program to create density-of-states (DOS) type info (Total DOS, Projected DOS, ...), to be plotted or printed.
- *adfnbo*: a program that generates an input file for gennbo.
- *gennbo*: a program that performs NBO (Natural Bond Orbital) analysis.
- *adf2aim*: a program that generates an input file for Xaim (Bader's analysis). Xaim is third party software.

Densf: Volume Maps

densf is an auxiliary program to generate values of molecular orbitals, charge densities and potentials in a user-specified grid, to be used typically for plotting or graphical display. The TAPE41 result file can be used directly by the ADFview program to visualize these properties.

densf requires an ascii input file where the user specifies the grid and the items that he/she wishes to see calculated on the grid, plus the standard result file TAPE21 from an *adf* calculation. *densf* writes a summary of the items that have been requested to standard output, together with some general information.

densf produces a (binary) KF file TAPE41, see OUTPUTFILE keyword below. TAPE41 is a kf file and all kf utilities can be used to inspect and process its data.

Furthermore, TAPE41 can be processed by *cntrs* to generate contour plot data. TAPE41 can also be processed by *adfplt* to display orbitals or densities on your screen, or to get a picture printed. The ADFview program can be used to view the data available in TAPE41 in a variety of ways. *Cntrs* and *adfplt* are separate utility programs, see later sections in this documentation. For the ADFview program separate documentation is available.

Starting from ADF2007, *densf* can also read and write cube files. See the CUBINPUT and CUBOUTPUT input options for details.

Examples of using *densf* are contained in the set of sample runs; see the *Examples* document.

Input

The input for *densf* is keyword oriented. The keywords may be specified in any order with one exception: INPUTFILE, if present, must be specified before any other option. Reading input by *densf* ends when it encounters the record EndInput or the end-of-file, whichever comes first.

The current version of *densf* does have reasonable defaults for all input. That means that in many cases you probably will not need to specify any input at all.

Below follows a list of the allowed keywords with their description.

Input/Output files

```
INPUTFILE {file}
```

INPUTFILE keyword specifies path to the TAPE21 file from which *densf* reads the input data. Absence of the keyword is treated as if **INPUTFILE TAPE21** has been specified.

```
OUTPUTFILE {file}
```

OUTPUTFILE keyword specifies path to the (possibly existing) TAPE41 file. If the file exists, *densf* will read grid specifications from it ignoring GRID keyword in the input. Computed quantities are saved in the file overwriting existing data with the same name, if any.

```
VTKFILE {file}
```

VTKFILE keyword specifies path to a file in the format readable by VTK directly. This option exists primarily for better integration with ADF-GUI and the user should not specify it.

```
CUBINPUT {file}
```

If the CUBINPUT keyword is present then the grid as specified in the **file** is used to calculate all requested quantities. Any volume data found in the cube file is also saved in the output file. NOTE: CUBINPUT option cannot be used with a pre-existing TAPE41 file because they both specify the grid, which may lead to a conflict.

```
CUBOUTPUT {file}
```

Presence of the CUBOUTPUT keyword tells densf to save all computed quantities as cube files using **file** as filename prefix. The prefix can also contain a complete path including directories. For example, specifying the following in the densf input

```
CUBOUTPUT /home/myhome/H2O
Density SCF
```

will result in a file /home/myhome/H2O%SCF%Density.cub being created containing volume data for the total SCF density. One file per requested quantity is created.

The OUTPUTFILE, CUBOUTPUT and VTKFILE options are mutually exclusive. Absence of any of these options is treated as if **OUTPUTFILE TAPE41** has been specified.

Grid

The Grid key is available either as simple key, or as block key.

The simple key options are as follows:

```
GRID {save} {coarse|medium|fine}
```

If the word save is specified, the program will store all grid points on TAPE41 (in addition to the specification of the grid that is always stored). The default is NOT to store all grid points.

Either coarse, medium or fine may be specified. This instructs the program to generate the grid automatically within a box enclosing all atoms of the molecule. The distance between grid points is 0.5, 0.2 or 0.1 bohr for respectively a coarse, medium or fine grid. Evidently the size of the result file TAPE41 depends strongly on this specification. The default value (used when the user does not specify the grid) is to generate a coarse grid.

If GRID is used as a block key it must be followed by the word end in a later record. The records until the end are the data for the Grid keyword:

```
Grid {save}
 x0, y0, z0
 n1, n2, n3
 v1x, v1y, v1z, length1
 v2x, v2y, v2z, length2
 v3x, v3y, v3z, length3
END
```

If the word save is specified, the program will store all grid points on TAPE41 (in addition to the specification of the grid that is always stored). The default is NOT to store all grid points.

The records in the data block must contain (*in the order specified below!*):

- 1 Three coordinates for the 'origin' (lower-left corner) of the grid.
- 2 Three integers: the numbers of points in three independent directions.
  If fewer integers are supplied the grid will accordingly be less-dimensional.

- 3 Three records each containing the coordinates for the direction of the independent vector (size irrelevant)
  and the total length of the grid in that direction.
  If a lower-dimensional grid is requested (see item #2), then fewer such direction-records are read and
  the redundant ones, if any, are ignored.
  The unit of length in which the grid size is input is by default Angstrom.
  The default can be overridden by using the input key UNITS, see below.

Notes:

- The second record ('three integers...') specifies the number of grid *points* in the different directions.
  The corresponding number of steps or intervals is one less!
- If the TAPE41 result file is to be used by the contour generating program *cntrs*,
  the grid used in the *densf* calculation must be *two*-dimensional.
- If the TAPE41 result file is to be used by ADFview, the grid used must be an three-dimensional orthogonal grid,
  with a single step size for all three dimensions.
- If the output TAPE41 file already exists and it contains valid grid data or if CUBINPUT is specified then the GRID input is ignored.
- The unit of length used in the input file has no relation to how the data are stored on the result file and how the program processes the data internally.
  Internal processing and storage on file is in bohr (atomic units).

Inline Grid

DENSF can now read grid as list of points. When specifying inline grid the GRID keyword should look as follows:

```
Grid Inline
  x1 y1 z1
  x2 y2 z2
  ...
  xN yN zN
End
```

Here, x#, y#, and z# are coordinates of points at which requested properties will be calculated. This feature may be used, for example, by external programs to calculate various properties at a number of points exactly and avoid interpolation with its inaccuracy. This feature should be used only when the output file has a TAPE41 format.

Units

As in the ADF main program, the unit of length can be set with the block type key

```
UNITS
 Length unit_of_length
END
```

In *densf* the only item that can be specified in the UNITS block is the length, so it seems a bit pointless to make UNITS a *block* type key rather than a simple key. However, to make its usage identical to the application in the *adf* main program the block form has been chosen to apply also here. The unit-of-length will apply to the grid specification in the input file. Default is angstrom.

Density

Generates the charge density in the grid. It is a simple keyword (not block-type).

```
density {fit} {frag} {ortho} {scf} {trans}
```

Occurrence of the word fit specifies that all densities specified in this record will be computed from the fit functions (an approximation to the exact density), rather than from the occupied molecular orbitals.

frag, ortho, scf, and trans causes each of the corresponding densities to be computed. frag stands for the sum-of-fragments (i.e. the initial) density, scf for the final result of the *adf* calculation, ortho for the orthogonalized fragments (orthogonalization to account for the Pauli repulsion, see the ADF User's Guide), and trans for excitation transition density.

Transition density is a product of initial and final states of an excitation. In the simplest case when initial and final states consist of one molecular orbital each, in this case the corresponding transition density is a product of the two MOs. To otain transition densities one needs to perform an excitations calculation with ADF, see EXCITATIONS keyword in ADF User's Guide. Transition densities for all excitations found in the input TAPE21 file will be calculated. The transition densities are always fit-densities.

If both the exact and the fit-densities are required the density keyword must be repeated, once with and once without the fit option specified.

The default (when the DENSITY key does not occur in the input file) is to calculate the final SCF density and the sum-of-fragments density.

The frozen core density is calculated with:

```
density core
```

Kinetic Energy Density and Electron Localization Function (ELF)

```
KinDens {frag} {orth} {scf}
```

Generates the Kinetic energy density and electron localization function on the grid.

Occurrence of any of the words requests calculation of the two quantities (KinDens and ELF) based on the corresponding density: sum-of-fragments, orthogonalized fragments, or SCF, respectively. If none of the options is present, scf is assumed.

Laplacian of the Density

The Laplacian of the exact SCF density is calculated with:

```
Laplacian
```

The Laplacian of the fitted SCF density is calculated with:

```
Laplacian fit
```

The LAPLACIAN key can occur multiple times. The LAPLACIAN feature is also supported by ADFview.

Gradient of the Density

The gradient of the exact SCF density is calculated with:

```
DenGrad
```

The gradient of the fitted SCF density is calculated with:

```
DenGrad fit
```

The gradient of the frozen core density is calculated with:

```
DenGrad core
```

The DENGRAD key can occur multiple times. This feature should be used only when the output file has a TAPE41 format.

Hessian of the Density

The hessian of the exact SCF density is calculated with:

```
DenHess
```

The Hessian of the fitted SCF density is calculated with:

```
DenHess fit
```

The Hessian of the frozen core density is calculated with:

```
DenHess core
```

The DENHESS key can occur multiple times. This feature should be used only when the output file has a TAPE41 format.

Potential

Generates the coulomb and/or exchange-correlation potential in the grid.

```
potential {coul / XC} {frag} {ortho} {scf}
```

frag, ortho, and scf are as for the density: at least one must be specified.

coul and XC specify that the Coulomb potential, respectively the exchange-correlation potential must be computed. Precisely one of these options must be specified in the record. If both potential types are required, another input record with the potential key must be used.

In the present release the xc option is not yet operational.

The default (when the POTENTIAL key does not occur in the input) is to calculate the SCF Coulomb potential.

Orbitals

A block type key in which the required molecular orbitals are specified. The key can be repeated in input any number of times; all occurrences are read and applied.

```
Orbitals type
 (data)
END
```

The argument of the orbitals key (type) must be scf (for the scf orbitals) or frag (for the fragment orbitals) or loc (for the localized molecular orbitals, see the ADF User's Guide).

The frag option is not operational in the present release.

In many data records in the ORBITALS block, as noted in the description of these data records, you may specify a HOMOLUMO range.

A HOMOLUMO range is the following:

```
   {HOMO{{-}n}} {LUMO{{+}n}}
```

HOMO: the highest occupied orbital
HOMO-n, with n an integer: the highest (n+1) occupied orbitals
LUMO: the lowest virtual orbital
LUMO+n, with n an integer: the lowest (n+1) virtual orbitals.

The HOMO part, or the LUMO part, or both must be specified. The integer n with sign is always optional, and the sign is always optional (and has no meaning, it is intended to enhance readability).

Thus, as an example,

```
    HOMO-1 LUMO+1
```

means a range of 4 orbitals: the two highest occupied ones, and the two lowest virtuals.

Each data record in the orbitals block must have either of the following formats:

1. the word alpha or beta.
This specifies that subsequent records refer to spin-alpha or spin-beta orbitals respectively. In a restricted calculation this has no meaning and beta must not be specified.
alpha and/or beta may occur any number of times in the orbitals block. All records until the first occurrence of alpha or beta are assumed to refer to spin-alpha orbitals.
2. label n1, n2, n3, ...
label is one of the subspecies of the point group symmetry used in the *adf* calculation and n1 etc. are indices of the molecular orbitals (in that subspecies) that are to be computed. This format is meaningless and must not be used for the loc orbitals type, because *localized* orbitals do not (necessarily) belong anymore to a particular symmetry representation.
3. label HOMOLUMO
label is one of the subspecies of the point group symmetry used in the calculation, the orbitals follow from the HOMOLUMO range.
4. label occ or **label virt**
occ specifies all orbitals (in that symmetry representation) up to and including the highest occupied one. virt specifies all orbitals above the highest occupied one. In this context *partially* occupied orbitals are considered occupied. Note carefully that if in a particular symmetry representation an empty orbital is computed below the highest occupied one in that same representation (excited state), that particular empty one is included in the list of occ.
Again, this format is meaningless and must therefore not be used for the loc type of orbitals.

5. all occ or **all virt** or **all HOMOLUMO**
Specifies *for each symmetry representation:*

- all orbitals up to and including the highest occupied one (in that symmetry), or
- all orbitals above the highest occupied one, or
- all orbitals defined by the HOMOLUMO range.

This form is not to be used for the LOC type of orbitals. However, using this for LOC will not result in an error but will be interpreted as identical to the following format.
6. all
This format must be used only for the LOC type of orbitals and simply means: all computed localized orbitals (irrespective of occupation numbers).
7. n1, n2, ...
a simple list of integer indices. This format must be used only for the loc type of orbitals since no reference is made to any symmetry representation. The indices refer of course to the list of localized orbitals as computed by *adf*, see the *User's Guide*.

The default value used when the ORBITALS key is not present is:

```
Orbitals SCF
 All HOMO-1 LUMO+1
End
```

NOCV

In ADF2009.01 it is possible to use DENSF to calculate $\varepsilon^*\varphi^2$ values of Natural Orbitals for Chemical Valence (NOCVs). Additional information on NOCVs is available in this paper.

The relevant part of the DENSF input is as follows:

For spin-unrestricted:

```
NOCV
  Alpha
   N1α
   N2α
   ...
  Beta
   N1β
   N2β
   ...
END
```

For spin-restricted:

```
NOCV
   N1
   N2
   ...
END
```

N1, N2, etc. specify sequential numbers of the orbitals for which $\varepsilon^*\varphi^2$ is to be calculated.

Alpha and *Beta* specify that the numbers that follow refer to spin α and β, respectively. Both *Alpha* and *Beta* are optional, *Alpha* being assumed if omitted. The NOCV input block must be closed with "END".

Alternatively, one can specify to calculate all (alpha- or beta-) NOCV's:

For spin-unrestricted:

```
NOCV
  Alpha
   ALL
  Beta
   ALL
END
```

For both spin-restricted and spin-unrestricted:

```
NOCV
   ALL
END
```

The last and probably the most convenient form of the NOCV input blocks lets one to specify an NOCV eigenvalue threshold as a criterion for selecting orbitals:

For spin-unrestricted:

```
NOCV
  Alpha
   THRESH threshold
  Beta
   THRESH threshold
END
```

For both spin-restricted and spin-unrestricted:

```
NOCV
   THRESH threshold
END
```

When this form of the input is used, only those NOCVs will be included whose absolute eigenvalue is equal to or larger than the given *threshold*.

Generic orbitals

There is also a possibility to calculate any orbital as long as it is present in the t21 file in the BAS representation. The input syntax is as follows:

```
Generic
 section1%variable1
 section2%variable2
End
```

In the example above, each line contains the section and variable name of the orbital in the input t21 file. The length of the variable should be equal to the number of atomic functions (naos) and it is supposed to contain expansion coefficients of the orbital on the basis of atomic (primitive) functions.

The calculation results are stored in the output file in sections and variables with exactly the same names as specified in the input. The section and variable names may contain spaces although the leading and training spaces are discarded.

Result: TAPE41

Follows a description of the contents of TAPE41. We start with a brief discussion of the sections. At the end you can find an uncommented list of all variables and sections. Note that some data are only generated when certain keywords are provided.

Sections on TAPE41

Grid

This is a general section. It contains the grid data and some more general info.

The grid characteristics are stored as:

- The 'origin' of the grid.
- The numbers of points in three independent directions.
- Three vectors, called 'x-vector', 'y-vector' and 'z-vector'.
  They are the *steps* in the three independent directions that define the grid.

If the save option was used in input (key grid) also all grid coordinates are stored: for each point three coordinates (xyz), also if only a 2-dimensional or 1-dimensional grid has been generated (a 2D grid does not necessarily lie in the xy-plane).

Note that the grid values are now stored in a simpler manner than in previous (prior to 2004) versions of densf, because the 'x values', 'y values', and 'z values' now each have their own, separate sections.

The remaining (general) data in this section comprises:

- The number of subspecies ('symmetries') for which data such as Molecular Orbitals may be present.
- The names of the subspecies.
- A logical with the name 'unrestricted', which flags whether the data pertain to an unrestricted calculation.
- The total number. of grid points.

SumFrag

Contains grid data of the Sum-of-fragments (charge density, coulomb potential, kinetic energy density, ELF, etc.).

Ortho

Contains similar data for the orthogonalized-fragments.

SCF

Contains the (spin) density, potential, etc. of the final (scf) solution.

Core

Contains grid data of the frozen core (charge density, gradients, Hessian).

TransDens_L1_L2

Contains grid data for electron transition densities. L1 is either SS or ST, and L2 is a symmetry label for all transitions in the section. Here SS and ST stand for Singlet-Singlet and Singlet-Triplet, respectively.

Variables in each section are Fitdensity_N and Coulpot_N for the density and Coulomb potential for excitation N within this spin and symmetry.

SCF_label

'Label' is one of the symmetry subspecies.

Each such section contains the total number of orbitals in that subspecies (as used in the *adf* calculation), with their occupation numbers and energy eigenvalues.

In addition it contains the grid-values of the (user-specified subset of) MOs in that subspecies. The variable name corresponding to an orbital is simply its index in the energy-ordered list of all orbitals (in that subspecies): '1', '2', etc.

LocOrb

Values of the localized orbitals.

NOCV

Values related to the NOCVs.

Geometry

Some general geometric information: the number of atoms (not counting any dummy atoms that may have been used in the *adf* calculation), their Cartesian coordinates (in bohr) and nuclear charges.

Note: the *order* of the atoms here is not necessarily identical to the input list of atoms: they are grouped by atom type.

Notes

- In an unrestricted calculation the section SCF_label is replaced by SCF_label_A and SCF_label_B for the spin-alpha and spin-beta data, respectively, and similarly for LocOrb: LocOrb_A and LocOrb_B.
- One or more subspecies may not have been used in the *adf* calculation.
  This happens when the basis set used in that calculation does not contain the necessary functions to span symmetry-adapted combinations of basis functions for that subspecies.
  In such a case the corresponding section on TAPE41 will not be created by *densf*.
- If you want to verify the contents of TAPE41, use the *pkf* utility to obtain a survey or *dmpkf* to get a complete ascii printout.

Contents of TAPE41

The information is presented in three columns. In the left-most column, section and variable names are printed, variable names being indented. In the middle column, variable's type and size is given. If the type is omitted, double precision floating point is assumed. The right-most column contains comments, if any.

Note that the name of a section of variable may consist of more than one word and that blanks in such names are significant. Furthermore, they are case-sensitive. Each line below contains the name of only one section or variable.

```
      NAME            length            Comment
Grid
  Start-point           (3)
```

nr of points x *(one integer)*
nr of points y *(idem)*
nr of points z *(idem)*
total nr of points *(idem)*
x-vector (3)
y-vector (3)
z-vector (3)
nr of symmetries *(one integer)*
labels *(nr of symmetries*160 characters)*
unrestricted *(one logical)*

SumFrag

CoulPot *(total nr of points)*
XCPot_A *(idem)* *spin-restricted:* XCPot
XCPOt_B *(idem)*
Density_A *(idem)* *spin-restricted:* Density
Density_B *(idem)*
Fitdensity_A *(idem)* *spin-restricted:* Fitdensity
Fitdensity_B *(idem)*
Kinetic Energy Density_A *(idem)* *spin-restricted:* Kinetic Energy Density
Kinetic Energy Density_B *(idem)*
ELF_A *(idem)* *spin-restricted:* ELF
ELF_B *(idem)*

Ortho

Same variables as in SumFrag

SCF

Same variables as in SumFrag and Ortho, and:

DensityLap_A *(idem)* *spin-restricted:* DensityLap
DensityLap_B *(idem)*
DensityGradX_A *(idem)* *spin-restricted:* DensityGradX
DensityGradX_B *(idem)*
DensityGradY_A *(idem)* *spin-restricted:* DensityGradY
DensityGradY_B *(idem)*
DensityGradZ_A *(idem)* *spin-restricted:* DensityGradZ
DensityGradZ_B *(idem)*
DensityHessXX_A *(idem)* *spin-restricted:* DensityHessXX
DensityHessXX_B *(idem)*
DensityHessXY_A *(idem)* *spin-restricted:* DensityHessXY
DensityHessXY_B *(idem)*
DensityHessXZ_A *(idem)* *spin-restricted:* DensityHessXZ
DensityHessXZ_B *(idem)*
DensityHessYY_A *(idem)* *spin-restricted:* DensityHessYY
DensityHessYY_B *(idem)*
DensityHessYZ_A *(idem)* *spin-restricted:* DensityHessYZ
DensityHessYZ_B *(idem)*
DensityHessZZ_A *(idem)* *spin-restricted:* DensityHessZZ
DensityHessZZ_B *(idem)*

Core

Density *(total nr. of points)*
DensityGradX *(idem)*
DensityGradY *(idem)*
DensityGradZ *(idem)*
DensityHessXX *(idem)*
DensityHessXY *(idem)*

```
  DensityHessXZ          (idem)
  DensityHessYY          (idem)
  DensityHessYZ          (idem)
  DensityHessZZ          (idem)
TransDens_L1_L2                        L1: SS or ST; L2 is excitation's symmetry
  Fitdensity_1           (total nr. of points)
  Fitdensity_2           (idem)
  Fitdensity_3           (idem)
  Coulpot_1              (idem)
  Coulpot_2              (idem)
  Coulpot_3              (idem)
SCF_label_A
     (label is a symmetry subspecies.
      Spin-restricted: SCF_label)
  nr of orbitals         (one integer)
  Occupations            (nr of orbitals)
  Eigenvalues            (idem)
  1                      (total nr of points)
  2                      (idem)
  3                      (idem)
     (as many as there are Molecular Orbitals in that
      symmetry representation for the indicated spin)
SCF_label_B
     (only if spin-unrestricted same variable as
      in SCF_label_A)
LocOrb_A                               if unrestricted, otherwise
                                       LocOrb
  nr of orbitals         (one integer)
  1                      (total nr. of points)
  2                      (idem)
  (etc)
NOCV
  Dens_A number*(occupation number)      (total nr. of points)
  Dens_B number*(occupation number)      (idem)
  (etc)
Geometry
  nnuc                   (one integer)
     (nr of nuclei, omitting dummy atoms)
  xyznuc                 (nnuc times 3)
     (the atoms are not in the same order as in the adf input
      file. Rather they are grouped by atomtype.)
  qtch                   (nnuc)          Atomic charges
x values
      x values           (total nr. of points)
y values
      y values           (idem)
z values
      z values           (idem)
```

Cntrs: Contour Plots

cntrs is an auxiliary program to generate plot data from TAPE41 produced by *densf*. In the future *cntrs* will be superseded by the ADF-GUI, which will offer both two-dimensional and three-dimensional visualization possibilities.

cntrs requires an ascii input file where the user specifies which items should be plotted and what scan values are to be used, plus the standard result file TAPE41 from *densf*. TAPE41 must be present as a local file in the directory where *contrs* executed. For usage by *contrs* TAPE41 must have been generated in a *densf* run using a *two*-dimensional grid.

cntrs produces as result one or more ascii files with plot data.

An example of using *cntrs* is contained in the set of sample runs (NO_2), see the *Examples* document.

Input

The (ascii) input for *cntrs* is keyword oriented. *The order of keywords in input is relevant for* cntrs.

Scan

```
scan
 scanvalues
END
```

With the scan key you read in values for which contours are to be generated for the items that are specified subsequently in input. Up to a maximum number of 20 scan values can be supplied. scan may occur any number of times in input. Each occurrence resets the scan values for the subsequent items. The initial values, which apply until the first occurrence, if at all, of scan are the eleven values 0, ±2e-2, ±5e-2, ±1e-1, ±2e-1, ±5e-1.

Dash

```
DASH length
```

contours corresponding to positive scan values are plotted as solid lines, the zero-contour is plotted by a dash-dot-dash line, and negative contours are dash-lines.

The dash key defines the length of a dash. Default: 0.2 bohr. dash may occur any number of times in input, each occurrence resets the dash-length for the items that follow.

Items to be plotted

The remaining part of input has the format:

```
FILE filename
item {factor}
item {factor}
...
FILE filename
item {factor}
...
(etc.)
END INPUT
```

Each FILE key requires the name of a file. This file must not yet exist and will be created by *cntrs* as an ascii file on which to write plot data. All 'items' until the next occurrence of dash will be combined into one quantity for the contours of which the plot data are generated.

Each item must be of the form Section%Variable and must in this way correspond to one of the variables on TAPE41 (*case-sensitive!*), see the description in the chapter about *densf*.. All items that belong to one file will be added up, each one multiplied by its factor (default: 1.0), to the quantity to be plotted. In this way you can generate contours for instance of density differences or a summation of densities.

Result

Each of the ascii result files, the names of which are defined in input (key file), consists of a sequence of data blocks.

Each block consists of a number of records that contain two values ('x' and 'y') and it ends with a blank line.

Each block defines a contour by plot instructions as follows:

- for the first {x,y}: start to plot at that point.
- for each next {x,y}: (continue to) draw the contour to that point.
- the blank line signals the end of the contour.

The last block does not correspond to a contour, but draws a rectangle around the whole picture.

ADFPLT: GUI for volume Maps

Adfplt is a tool [Autschbach, 1999 #1110] to graphically display orbitals, densities or potentials computed in a 2D or 3D grid. You can view the picture on your screen, or put it in a postscript file for printing, or send it to another device. This auxiliary program is installed with the other programs of the ADF package, but only if you have previously installed a few graphical libraries and you have correctly assigned the appropriate Environment Variable GKSGR_LIB to point to the directory where these libraries reside.

Environment variable: only for ADFPLT

variable	typical value	comments
GKSGR_LIB	/usr/local/lib	(full path) directory with GKSGR lib

The graphical libraries GKS and GR are required to install and operate the *adfplt* utility. These (graphical) libraries are not part of ADF, but are available for free from their developers in Jülich, Germany. The libraries may not be available for all platforms, which then implies that cannot use *adfplt* on such machines. Consequently we cannot guarantee the continuity of their availability in any respect. For instructions concerning the GKS and GR libraries, please check the ADFPLT link on our Contributed Software page.

The ADFview module of the ADF-GUI is the suggested alternative for ADFPLT that is supported by SCM.

Adfplt requires a TAPE41 file (with that name) computed by *densf*, using a 3D grid. You start the program by invoking its executable with as argument the data you want to display.

```
$ADFBIN/adfplt Sec%Var
```

Sec%Var must reference existing data on your TAPE41 file. To get a survey of all data on TAPE41, use the *pkf* facility.

```
$ADFBIN/pkf TAPE41 > t41_ascii
```

You can then inspect the ascii dump of the TAPE41 contents to determine what you can display.

After having typed the command, you get a list of available output devices to send the picture to. Select one of them by typing the indicated number, for instance 211 to get the picture on your screen. (The appropriate choice may depend on your system settings, but a little try-and-error will soon help you on your way).

You may add options in the call of *adfplt*.

```
$ADFBIN/adfplt -opt1 -opt2 [...] Sec%Var
```

You get a list of all available options by typing

```
$ADFBIN/adfplt -help
```

In the current implementation, each run is one-time shot. To get another picture you have to start the program again.

Dos: Density of States

The auxiliary program *dos* computes various types of densities-of-states (DOS) for a user-specified energy interval.

dos requires an ascii input file where the user specifies the items to be calculated and computational details, plus the standard result file TAPE21 from an *adf* calculation. The latter file must be present as a local file with name TAPE21 in the directory where *dos* is executed.

dos produces as result one or more ascii files with the density-of-states values. Error messages and computational info (if any) are written to standard output.

Introduction

The program *dos* gives information on the number and character of one-electron levels (molecular orbitals) as a function of the (orbital) energy. The total density of states $N(E)$ is a well known concept in electronic structure theory of infinite systems (crystals). $N(E)dE$ denotes the number of one-electron levels (orbitals) in the infinitesimal energy interval dE. The total density of states (TDOS) at energy E is usually written as

$$N(E) = \sum_i \delta(E-\varepsilon_i) \quad (3.3.1)$$

where the ε_i denote the one-electron energies. So the integral of $N(E)$ over an energy interval E_1 to E_2 gives the number of one-electron states in that interval. Usually the δ-functions are broadened to make a graphical representation possible.

When the δ-functions are multiplied by a weight factor that describes some property of the one-electron state φ_i at energy ε_i various types of densities-of-states are obtained that provide a graphical representation of the state character (orbital character) as a function of one-electron energy.

In calculations on finite molecules the total density of states as a function of (orbital) energy may also be useful, but the main use of various types of densities-of-states is to provide a pictorial representation of Mulliken populations. The weight factors employed are related to the orbital character determined by means of a Mulliken population analysis *per orbital* (see below). The program *dos,* therefore, provides the same information as can be generated by the ADF program (a population analysis per orbital) but *dos* enables an easy graphical representation and is particularly useful when there are many one-electron levels, for instance in calculations on clusters. You can obtain a simple view of the character of the orbitals in a certain energy range. You can also find out in which orbitals (at which energies) certain basis functions or fragment orbitals give a large contribution, and whether such contributions are bonding, nonbonding or antibonding with respect to particular bonds. Such information is provided by *dos* in the form of (weighted) density of states values over a user-specified energy range, which can for instance be plotted by *gnuplot*.

The following options are available for computations by *dos*:

- TDOS: Total Density of States
- GPDOS: Gross Population Density of States
- OPDOS: Overlap Population Density of States
- PDOS: Projected Density of States

The total density of states (TDOS) has large values at energies where there are many states per energy interval.

The GPDOS (Gross Population based Density Of States) of a function χ_μ (or a sum of such functions) has large values at energies where this function (these functions) occur(s) in the molecular orbitals.

The PDOS of a function χ_μ provides similar information, but with the projection of χ_μ onto the orbital φ_i as weight factor for the importance of χ_μ in the orbital φ_i.

The OPDOS (Overlap Population based Density Of States) between χ_μ and χ_ν has large positive values at energies where the interaction between them is bonding, and negative values where the interaction is antibonding. An example of the use of these plots is provided in [1].
[1] P.J. van den Hoek, E.J. Baerends, and R.A. van Santen, J. Phys. Chem. **93**, 6469 (1989).

We review below the Mulliken population analysis, and then describe the forms of density of states analysis performed by DOS. Finally an input description of DOS is given.

Mulliken population analysis

The orbitals φ_i with energies ε_i are expanded in basis functions χ_μ, which leads to the definition of density matrices P_i describing orbital densities, from which the total density matrix can be constructed:

$$\varphi_i(\mathbf{r}) = \sum_\mu \chi_\mu(\mathbf{r})\, C_{\mu i}$$
$$\rho_i(\mathbf{r}) = \int|\varphi_i(\mathbf{r})|^2 = \sum_{\mu\nu} P_{i,\mu\nu}\, \chi_\mu(\mathbf{r})\, \chi_\nu(\mathbf{r}); \quad P_{i,\mu\nu} = C_{\mu i} C_{\nu i}$$
$$\rho(\mathbf{r}) = \sum_i n_i \rho_i(\mathbf{r}) = \sum_{\mu\nu} P_{\mu\nu}\, \chi_\mu(\mathbf{r})\, \chi_\nu(\mathbf{r}); \quad P_{\mu\nu} = \sum_i n_i C_{\mu i} C_{\nu i} \quad (3.3.2)$$

Here μ and μ run over the basis functions, which may be either primitive functions, or combinations of primitive functions, for instance the SCF orbitals of atoms or larger fragments.

The Mulliken population analysis provides a partitioning of either the total charge density or an orbital density. The total density is written as

$$\rho(\mathbf{r}) = \sum_{\mu\nu} P_{\mu\nu}\, \chi_\mu(\mathbf{r})\, \chi_\nu(\mathbf{r}) = \sum_{A\le B} \sum_{\mu\in A} \sum_{\nu\in B} P_{\mu\nu}\, \chi_\mu\, \chi_\nu = \sum_{A\le B} \rho_{AB} \quad (3.3.3a)$$
$$\rho_{AB} = \sum_{\mu\in A} \sum_{\nu\in B} P_{\mu\nu}\, \chi_\mu\, \chi_\nu \quad (3.3.3b)$$

The total number of electrons, $N = \int \rho(\mathbf{r}) d(\mathbf{r})$, is now partitioned over the atoms by assigning an overlap population $P_{\mu\nu}S_{\mu\nu} + P_{\nu\mu}S_{\nu\mu}$ for one half to the atom A of χ_μ and one half to atom B of χ_ν,

$$N = \int \rho(\mathbf{r}) d(\mathbf{r}) = \sum_{\mu\nu} P_{\mu\nu} S_{\mu\nu} = \sum_\mu GP_\mu \quad (3.3.4a)$$
$$GP_\mu = \sum_\nu P_{\mu\nu} S_{\mu\nu} \quad (3.3.4b)$$

GP_μ is the gross population of χ_μ. It contains the net population $P_{\mu\mu}$ and half of each total overlap population $P_{\mu\nu}S_{\mu\nu} + P_{\nu\mu}S_{\nu\mu}$ between χ_μ and χ_ν. Summing the gross populations over the functions $\mu \in A$ yields the total number of electrons assigned to atom A, or the gross population of atom A, GP_A, and hence the gross charge Q_A of atom A,

$$GP_A = \sum_{\mu\in A} GP_\mu \quad (3.3.5a)$$
$$Q_A = Z_A - GP_A \quad (3.3.5b)$$

The overlap population $OP_{\mu\nu}$ between two functions and the overlap population Q_{AB} between two atoms are defined in an analogous manner,

$$OP_{\mu\nu} = P_{\mu\nu}S_{\mu\nu} + P_{\nu\mu}S_{\nu\mu} \quad (3.3.6a)$$
$$Q_{AB} = \sum_{\mu\in A} \sum_{\nu\in B} OP_{\mu\nu} \quad (3.3.6b)$$

These quantities can be evaluated for a single orbital density, $N = 1 = \int|\varphi_i(\mathbf{r})|^2 d\mathbf{r}$. The gross population $GP_{i,\mu}$ of a function in a specific orbital density $|\varphi_i(\mathbf{r})|^2$ is then associated with the fraction of the orbital density belonging to that function (or the percentage χ_μ character of orbital φ_i, and the overlap population $OP_{i,\mu\nu}$ gives an indication of the strength of bonding or antibonding between χ_μ and χ_ν in orbital φ_i,

$$GP_{i\mu} = \sum_\nu P_{i,\mu\nu} S_{\mu\nu} = \sum_\nu C_{\mu i} C_{\nu i} S_{\mu\nu} \quad (3.3.7a)$$
$$OP_{i,\mu\nu} = P_{i,\mu\nu}S_{\mu\nu} + P_{i,\nu\mu}S_{\nu\mu} = 2\, C_{\mu i} C_{\nu i} S_{\mu\nu} \quad (3.3.7b)$$

Density of states analyses based on Mulliken population analysis

Total density of states

The total density of states TDOS at energy *E* is written as

TDOS: $N(E) = \sum_i \delta(E-\varepsilon_i)$ (3.3.8)

so the integral of $N(E)$ over an energy interval E_1 to E_2 gives the number of one-electron states in that interval. In practice the delta functions are approximated by Lorentzians,

TDOS: $N(E) = \sum_i L(E-\varepsilon_i) = \sum_i \{\sigma/\pi \cdot 1/[(E-\varepsilon_i)^2+\sigma^2]\}$ (3.3.9)

A plot of $N(E)$ *versus E* reveals energetic regions where many levels are located. The width parameter s determines of course the appearance of the plot. A typical value is 0.25 eV (used as default in *dos*).

Partial (gross population and projected) density of states

In order to find out if a given function χ_μ contributes strongly to one-electron levels at certain energies, one may weigh a one-electron level with the percentage χ_μ character. We usually determine the χ_μ character by the gross populations, obtaining the GPDOS form of the partial density of states,

GPDOS: $N_\mu(E) = \sum_i GP_{i,\mu} L(E-\varepsilon_i)$ (3.3.10)

If the weight factor is determined by projection of φ_i against χ_μ, we obtain the projected density of states PDOS,

PDOS: $N_\mu(E) = \sum_i |< \chi_\mu | \varphi_i >|^2 L(E-\varepsilon_i)$ (3.3.11)

One should not use the PDOS for d-type or f-type primitive basis functions ('BAS'). A d-type function consists of 6 Cartesian functions, while there can of course be only 5 true d-type functions among them: one (linear combination) of them is in fact an s-type function (x2+y2+z2). Similarly, there are 10 f-type Cartesian functions, 3 of which are in fact p-functions. The PDOS is calculated for the 6 d-type and 10 f-type Cartesian functions, which leads to undesired results. An PDOS for SFOs does not suffer from this problem.

Overlap population density of states (OPDOS)

If the delta function representing orbital φ_i is weighed with the overlap population between χ_μ and χ_ν in φ_i, the overlap population density of states OPDOS is obtained,

OPDOS: $N_{\mu\nu}(E) = \sum_i OP_{i,\mu\nu} L(E-\varepsilon_i)$ (3.3.12)

If an orbital φ_i at energy ε_i is strongly bonding between χ_μ and χ_ν the overlap population is strongly positive and OPDOS(e) will be large and positive around $E=\varepsilon_i$. Similarly, OPDOS(E) will be negative around energy ε_i when there is antibonding between χ_μ and χ_ν in φ_i.

The OPDOS(E) has been used under the name coop (crystal orbital overlap population) in Extended-Hückel solid state calculations by Hoffmann and coworkers [2].
[2] R. Hoffmann, *A chemist's view of bonding in extended structures* (VCH Publishers, New York, 1988).

Generalizations of OPDOS, GPDOS, PDOS

As observed above, the basis functions in the above expressions may be primitive basis functions ('Slater type orbitals'), but of course the formulas are equally applicable for other types of MO expansions. In *dos* the

user may select either the expansion in primitive basis functions ('BAS') or the expansion in SFOs (Symmetrized Fragment Orbitals) for the DOS analyses.

It is also possible in DOS to treat a *set* of basis functions simultaneously. For instance, the GPDOS for a set of basis functions μ_1, μ_2, ... is simply defined as the summation of the corresponding single-function GPDOS(E) values

$$N_{\mu\text{-set}}(E) = \sum_{\mu\in\mu\text{-set}} \sum_i GP_{i,\mu} L(E-\varepsilon_i) \quad (3.3.13)$$

In a similar fashion the OPDOS can be defined for *two sets* of basis functions μ_1, μ_2, ... and ν_1, ν_2, ... as

$$N_{\mu\text{-set},\nu\text{-set}}(E) = \sum_{\mu\in\mu\text{-set}} \sum_{\nu\in\nu\text{-set}} \sum_i OP_{i,\mu\nu} L(E-\varepsilon_i) \quad (3.3.14)$$

and finally for the PDOS we get in similar fashion

$$N_{\mu\text{-set}}(E) = \sum_{\mu\in\mu\text{-set}} \sum_i |\langle \chi_\mu | \varphi_i \rangle|^2 L(E-\varepsilon_i) \quad (3.3.15)$$

Input

The (ASCII) input for *dos* is keyword oriented. Reading input by *dos* terminates whenever it finds a line END INPUT or the end-of-file, whichever comes first.

Follows a list of keywords with their meaning. Generally keys may occur more than once and *the order in which they appear is relevant in some cases*. For instance the key energyrange (which defines for what energy values to compute densities-of-states, see below) applies to all items that come after it in input until the next occurrence of energyrange.

Energy scan values

```
ENERGYRANGE {Npoint=nr} {E-start=e1} {E-end=e2 / E-step=de}
```

This specifies for which energy values the densities-of-states are computed that are specified *after* it in the input file and *until* the next occurrence of ENERGYRANGE.

ENERGYRANGE specifies the lower bound, upper bound and number of equidistant energy values (including end-points). All items are optional with defaults applying for those omitted.

The E end and E-step values determine one another and must therefore not be specified both (or be consistent).

The initial defaults are:

```
nr=301
e1=-20
de=0.1
```

All energy data are in eV.

When values have been changed with the key ENERGYVALUE, the so-modified values are the defaults for the next occurrence of ENERGYVALUE.

Peak widening

The peaks in the DOS curves corresponding to the energies of the molecular orbitals are widened by a Lorentzian curve, the width of which can be adjusted.

```
LORENTZIAN width=width
```

Initial default width is 0.25 (eV).

As for ENERGYRANGE, the key LORENTZIAN may occur more than once and each occurrence sets the width for all items after it.

Result files

The computed densities-of-states are stored on one or more ascii files, which have to be specified in input.

```
FILE file
```

The key FILE may occur any number of times in input. Each time it occurs the specified file is opened by *dos*. The file must not yet exist and the new file will accumulate (ascii) the densities-of-states data of all DOS items subsequently specified, until the next occurrence of FILE. The first occurrence of the key FILE must be given before any DOS specification (by the keys TDOS, OPDOS, GPDOS, PDOS, see below).

The format of the result file is such that it can be fed directly into *gnuplot*.

Densities of States

```
TDOS { title }
```

TDOS
instructs the program to compute the *total* density of states.
title (optional)
will appear as title to the section of corresponding Density-of-States data in the result file.

The other types of densities-of-states require block-type keyword input.

```
OPDOS { title }
Ftype numbers
Ftype numbers
...
SUBEND
Ftype numbers
Ftype numbers
...
END
```

Ftype
Specifies the type of basis functions to use in the MO expansions. If the primitive basis functions are to be used Ftype must be bas. For the SFO representation Ftype must be one of the irreducible representations of the pointgroup symmetry. All Ftype values in the data block must be consistent: either all are bas or all are irrep labels. The scope of this consistency requirement is the data block of the current key: in a next OPDOS data block, for instance, a different choice may be made.
numbers
Must be a sequence of integers referring to the basis functions to be selected, i.e. the 'μ-set' and 'v-set'

in (3.3.13) etc.
If bas-type basis functions are selected the numbers refer to the overall list of all basis functions as printed in the output file of the *adf* run. If SFOs are selected the numbers refer to the SFO list of the pertaining symmetry representation *without the core functions*, see the *adf* output file.
SUBEND
Must be typed as such and separates the 'μ-set' and the 'ν-set': all records before subend specify together the 'μ-set' and all records below subend comprise the 'ν-set'. Each of these two sections may consist of any number of records.

The input for GPDOS and PDOS are similar, but simpler because only one set of functions ('μ-set') has to be specified, so there is no subend in the data blocks for these keys.

```
GPDOS { title }
Ftype numbers
Ftype numbers
...
END
PDOS { title }
Ftype numbers
Ftype numbers
...
END
```

The keys GPDOS, OPDOS, PDOS and (TDOS) may occur any number of times in input and in any order. Each time the DOS key occurs the current energyrange and lorentzian settings apply and the results are written to the current file.

Adfnbo, gennbo: NBO analysis

Dr. Autschbach, SCM, and Prof. Weinhold have collaborated to prepare a simple in put file generator, called adfnbo, for the GENNBO program of Prof. Weinholds Natural Bond Orbital (NBO) 5.0 package:
http://www.chem.wisc.edu/~nbo5
The GENNBO executable is included in the ADF distribution and can be enabled via the license file for all those who buy an NBO manual from either the NBO authors or from SCM (info@scm.com). An extensive documentation of GENNBO is part of the NBO manual. ADFNBO is currently suitable only for spin-restricted calculations and its application to frozen-core basis sets also needs to be further tested.

Next a brief summary of the capabilities of GENNBO is given (by Prof. Weinhold).
GENNBO implements most capabilities of the full NBO 5.0 program suite as described on the NBO website:
http://www.chem.wisc.edu/~nbo5
These include determination of natural atomic orbitals (NAOs), bond orbitals (NBOs), and localized MOs (NLMOs), as well as the associated NPA (atomic charges and orbital populations) and NRT (resonance structures, weightings, bond orders) valence descriptors, for a wide variety of uncorrelated and correlated (variational, perturbative, or density functional) theoretical levels. GENNBO-supported options include all keywords except those explicitly requiring interactive communication with the host electronic structure system (viz., $DEL deletions, NEDA, NCS, NJC). The GENNBO program typically sits conveniently on the PC desktop, ready to analyze (or re-analyze at will, with altered options) the final results of a complex ADF calculation performed on a remote cluster.

GENNBO "communicates" with the original ADF calculation through an archive file (JOB.47 file, preserving all necessary details of the final density) that is initially generated by ADF and subsequently becomes the input file for GENNBO. The .47 file contains a standard $NBO ... $END keylist that can be edited with a standard word processor or text editor to include chosen NBO keyword options, just as though they might have appeared in the original input stream of an interactive ADFNBO run. The stand-alone GENNBO program therefore allows many alternative NBO analysis options to be explored at leisure, without costly re-calculation of the wave function.

NBO analysis of EFG, NMR chemical shifts, NMR spin-spin coupling

For certain molecular properties it is possible to perform detailed analyses in terms of Natural Bond Orbitals (NBOs) and Natural Localized Molecular Orbitals (NLMOs). These features generally require a sequence of ADF and/or property code runs. An initial nonrelativistic or scalar relativistic ADF run, followed by the generation of NBO and NLMO data, is required, and the resulting data files need to be present in subsequent property calculations, along with a keyword indicating that the NBO analysis is requested in the property module.

We have noted in the past some slight loss of numerical accuracy of the results after going through the various orbital transformations in the NBO - NLMO sequence. It is important that the user verifies in each case that the total contributions from the analysis are in agreement with the total calculated property, within the numerical integration accuracy limits. In order to assist the user with this, the analysis program always print the total analysis contributions, including small nonprinted values.

Moreover, there appears to be a problem with the analysis of the Fock matrix in the NBO program in conjunction with ADF calculations. Therefore please do NOT use the Fock matrix second order perturbation theory analysis in NBO at this time. We will remove this disclaimer once the issue has been fixed. Applications of the NBO-NLMO property analysis codes have so far given no indication that the Fock matrix issue interferes with the analysis.

Important note: If properties are analyzed from within spin-orbit relativistic computations, the NBO/NLMO analysis is performed in terms of scalar (spin-free) relativistic orbitals, as detailed in the technical references.

The results from these analyses are exact in the sense that they fully reproduce the final spin-orbit property result, and they allow to dissect the property in terms of more intuitive one component real scalar relativistic localized orbitals. Typically, the property analysis in a spin-orbit calculation involves contributions from unoccupied scalar NLMOs, whereas there are no such contributions if a nonrelativistic or scalar relativistic property is analyzed.

Available properties for NBO analysis: EFG, NMR chemical shifts and NMR spin-spin coupling.

EFG

EFGs: nonrelativistic and scalar ZORA, in ADF/AORresponse. Requires initial ADF run with

```
AOresponse
 donothing
End
```

in order to generate orbitals that re equivalent to those generated in the subsequent ADF run where the EFG is calculated. Alternatively, simply calculate the EFG twice, once before the NBO generation step, and once afterward.

The next step (see below) is to create the NBOs and the required data files for the analysis. Afterward, in the second ADF run, use

```
Aoresponse
 efg NUC nbo
end
```

`efg NUC nbo`

> Here NUC is the number of the nucleus at which the EFG is to be computed (ADF internal atom ordering). Example: efg 1 nbo.

In addition to the optional NBO analysis, the EFG program in AOResponse prints a Mulliken type analysis of the EFG principal components, and an analysis in terms of canonical MOs.

WARNING: the ordering of the principal components is lowest to highest including the sign. That is, we have V11 ≤ V22 ≤ V33. This does not conform to the usual convention of |V11| ≤ |V22| ≤ |V33|. Please make sure you select the right component for your analysis.

Example job: $ADFHOME/examples/adf/AlCl3_efgnbo

For an explanation of the output and a general usage tutorial, see
J. Autschbach, S. Zheng, and R.W. Schurko, *Analysis of Electric Field Gradient Tensors at Quadrupolar Nuclei in Common Structural Motifs*, Concepts in Magnetic Resonance Part A **36A**, 84 (2010)

Further references and recommended citations:
A.J. Rossini, R.W. Mills, G.A. Briscoe, E.L. Norton, S.J. Geier, I. Hung, S. Zheng, J. Autschbach, and R.W. Schurko, *Solid-State Chlorine NMR of Group IV Transition Metal Organometallic Complexes*, Journal of the American Chemical Society **131**, 3317 (2009)

NMR Chemical shift

An implementation is currently available for spin-orbit ZORA computations. If scalar ZORA calculations are to be analyzed, provide the input keyword

```
FAKESO
```

in the NMR input (outside of the 'nmr' keyword). A native scalar ZORA implementation that does not require this keyword is under development. If this feature is requested one should restrict the calculation to a single shielding tensor per NMR run. It would be good practice to check the results against regular NMR calculations where the analysis feature is not requested. No ZORA scaling is applied in the analysis results. The data should be equivalent to a regular computation in the NMR input with

```
NMR
 u1k best
 calc all noscl
END
```

Depending on whether scalar or spin-orbit calculations are to be analyzed, the sequence of calculations is different:

scalar:
1. ADF, scalar ZORA
2. generate NBOs and required data files for analysis
3. NMR with FAKESO and analysis keywords, use TAPE21, TAPE10 from step 1.

spin-orbit:
1. ADF, scalar ZORA
2. generate NBOs and required data files for analysis
3. delete TAPE21, TAPE10, TAPE15
4. ADF, spin-orbit ZORA
5. NMR with analysis keywords, using TAPE21, TAPE10 from step 4

In the NMR run, in addition to the NMR keyword, provide the following

```
analysis
 print 0.01
 canonical
 nbo
 components
end
```

The optional canonical keyword can be used independently from the NBO analysis features. It enables an analysis of the shielding in terms of the canonical MOs. The components keyword is optional and enables an analysis not only of the isotropic shielding but also of each principal component of the tensor. The print keyword selects printout of contributions relative to the total diamagnetic, paramagnetic. In the example, only contributions greater than 1% are printed. Set to zero to print ALL contributions.

Example job: $ADFHOME/examples/adf/CH4_nmrnbo

References:

J. Autschbach, *Analyzing NMR shielding tensors calculated with two-component relativistic methods using spin-free localized molecular orbitals*, Journal of Chemical Physics **128**, 164112 (2008)

J. Autschbach and S. Zheng, *Analyzing Pt chemical shifts calculated from relativistic density functional theory using localized orbitals: The role of Pt 5d lone pairs*, Magnetic Resonance in Chemistry **46**, S45 (2008)

J. Autschbach and S. Zheng, *Relativistic computations of NMR parameters from first principles: Theory and applications*, Annual Reports on NMR Spectroscopy **67**, 1 (2009)

NMR spin-spin coupling (J-coupling)

Nonrelativistic, scalar ZORA, spin-orbit ZORA

The sequence of jobs is similar to those in the NMR section.

scalar or **nonrelativistic**:
1. ADF, scalar ZORA or nonrel.
2. generate NBOs and required data files for analysis
3. CPL with analysis keyword, use TAPE21, TAPE10 from step 1.

spin-orbit:
1. ADF, scalar ZORA
2. generate NBOs and required data files for analysis
3. delete TAPE21, TAPE10, TAPE15
4. ADF, spin-orbit ZORA
5. CPL with analysis keyword, using TAPE21, TAPE10 from step 4

In the CPL run provide the following 'contributions' keyword to enable the analysis

```
nmrcoupling
 ... other options
 contributions 1E19 nbo
end
```

The numerical value selects a print threshold in SI units of T**2/J for the analysis. Increase the value to obtain less detail in the analysis. By default, 'contributions' triggers an analysis of the J-coupling in terms of canonical MOs. The nbo keyword enables in addition the NBO-NLMO analysis.

Please note that due to the history of how the program was developed the output from the scalar/nrel. analysis and from the spin-orbit calculations differs somewhat. The qualitative content is the same.

In scalar ZORA or nonrelativistic CPL calculations without the SD term an orbital based analysis is only performed for the Fermi-contact mechanism. If you also need an analysis for the PSO and SD mechanisms but do not want to run a spin-orbit calculation with ADF please use the SD or NOSD keywords which will cause the spin-orbit branch of the CPL code to be used. In ZORA spin-orbit calculations the FC, SD, PSO, and cross terms are analyzed together by default. You can selectively switch them on or off in order to get individual mechanism analyses. The DSO mechanism is often negligible. An analysis tool for this mechanism has therefore not yet been developed.

Example job: $ADFHOME/examples/adf/CPL_CH3OH_NBO

References:

NMR spin-spin couplings with NBO analysis

J. Autschbach, *Analyzing molecular properties calculated with two-component relativistic methods using spin-free Natural Bond Orbitals: NMR spin-spin coupling constants* Journal of Chemical Physics **127**, 124106 (2007)

J. Autschbach and B. Le Guennic, *Analyzing and interpreting NMR spin-spin coupling constants from molecular orbital calculations*, Journal of Chemical Education **84**, 156 (2007)

A.M.A. Boshaala, S.J. Simpson, J. Autschbach and S. Zheng, *Synthesis and Characterization of the Trihalophosphine Compounds of Ruthenium [RuX$_2$(η^6-cymene)(PY$_3$)] (X = Cl, Br, Y = F, Cl, Br) and the Related PF$_2$(NMe$_2$) and P(NMe$_2$)$_3$ Compounds; Multinuclear NMR Spectroscopy and the X-ray Single*

Crystal Structures of [$RuBr_2(\eta^6$-cymene)(PF_3)], [$RuBr_2(\eta^6$-cymene)(PF_2{NMe_2})], and [$RuI_2(\eta^6$-cymene)(P{NMe_2}$_3$)], Inorganic Chemistry **47**, 9279 (2008)

J. Autschbach and S. Zheng, *Relativistic computations of NMR parameters from first principles: Theory and applications*, Annual Reports on NMR Spectroscopy **67**, 1 (2009)

Generation of NBOs

How to generate the NBOs, NLMOs, and the data files needed for these calculations (step 2 below is step 2 in the examples above):

1. run ADF with scalar ZORA or nonrelativistic options, and keep TAPE21 and TAPE15.

2.

```
# run adfnbo in WRITE  mode to create the gennbo input file FILE47
# and one of the required property analysis files, adfnbo.kf

$ADFBIN/adfnbo << eor
write
spherical
end input
eor

rm -f adfnbo.37 adfnbo.39 adfnbo.49 adfnbo.48
$ADFBIN/gennbo < FILE47

# run adfnbo in COPY mode to create the second property analysis
# file, adfnbo2.kf

$ADFBIN/adfnbo << eor
 spherical
 copy
end input
eor

# run adfnbo in READ mode: prepare locorb on TAPE21

$ADFBIN/adfnbo << eor
 spherical
 read
end input
eor

rm -f adfnbo.37 adfnbo.39 adfnbo.49 adfnbo.48

# keep the TAPE21 after this sequence in order to
# be able to plot the NBOs and NLMOs with adfview:

mv TAPE21 nbonlmo.t21

# clean up, keep adfnbo*.kf for any NBO property analyses.
```

Adf2aim: make input for Xaim (Bader's analysis)

ADF utility adf2aim (original name rdt21) developed by Xavi López, Engelber Sans and Carles Bo (see http://www.quimica.urv.es/ADF_UTIL): convert an ADF TAPE21 to WFN format (for Bader analysis) The program rdt21 is now called adf2aim and is part of the ADF package, starting from ADF2004.01.

The WFN file is an input file for the third party program Xaim (see http://www.quimica.urv.es/XAIM for details), which is a graphical user interface to programs that can perform the Bader analysis.