BAND Manual

Release 2017

www.scm.com

Apr 12, 2018

CONTENTS

GENERAL

1.1 Introduction

The Amsterdam Density Functional Band-structure program - **BAND** - can be used for calculations on periodic systems, i.e. polymers, slabs and crystals, and is supplemental to the molecular ADF program for non-periodic systems. It employs density functional theory in the Kohn-Sham approach. BAND is very similar to ADF in the chosen algorithms, although important differences remain. Like ADF, BAND makes use of atomic orbitals, it can handle elements throughout the periodic table, and has several analysis options available. Unlike ADF, BAND can use numerical atomic orbitals, so that the core is described very accurately. Because of the numerical orbitals BAND can calculate accurate total energies. Furthermore it can handle basis functions for arbitrary l-values.

The installation of BAND is explained in the Installation manual. There you can also find information about the license file, which you need to run the program.

This User's Guide describes how to use the program, how input is structured, what files are produced, and so on. The Examples section explains the most popular features in detail, by commenting on the input and output files in the $ADFHOME/examples/band directory.

Where references are made to the operating system (OS) and to the file system on your computer the terminology of UNIX type OSs is used.

This guide and other documentation is available at http://www.scm.com. As mentioned in the license agreement, it is mandatory, for publications in which BAND has been used, to cite the lead references.

1.2 Characterization of BAND

Functionality

- Automatic geometry optimization

- Formation energy with respect to isolated atoms which are computed with a fully numerical Herman-Skillman type subprogram

- A choice of density functionals, including LDA (Local Density Approximation), GGA (Generalized Gradient Approximation), meta-GGA (with kinetic energy density dependence), and Hybrid (including exact exchange) functionals

- Time-dependent Current-DFT (TD-CDFT) for calculation of frequency-dependent dielectric functions of systems periodic in one, two and three dimensions.

- The ZORA method for scalar relativistic effects is available (also for the TD-CDFT option). Spin-orbit coupling can be taken into account, including non-collinear magnetization.

- Phonon spectrum.

- Electric field perpendicular to the periodic direction(s).
- Non-Equilibrium Green's Function (NEGF) method for calculation of transmission and current plots.

Analysis

- Mulliken populations for basis functions, overlap populations between atoms or between basis functions.
- Hirshfeld charge analysis
- Densities-of-States: DOS, PDOS and OPWDOS/COOP
- Local Densities-of-States: LDOS (STM images)
- Form factors (X-ray structures)
- Charge analysis using Voronoi cells (yielding Voronoi Deformation Charges)
- Orbital plots
- Deformation density plots
- Band structure plots along the edges of the Brillouin zone
- One-electron energies and orbitals at the Brillouin Zone sample points
- Fragment orbitals and a Mulliken type population analysis in terms of the fragment orbitals
- Quantum Theory of Atoms In Molecules (QT-AIM). Atomic charges and critical points.
- Electron Localization Function (ELF).
- Fragment based Periodic Energy Decomposition Analysis (PEDA).
- PEDA combined with Natural Orbitals for Chemical Valency (NOCV) to decompose the orbital relaxation (PEDA-NOCV).

Technical

- Linear scaling techniques are used to speed up calculations on large unit cells
- SCF convergence based on a Direct Inversion of Iterative Subspace (DIIS) method. For problematic systems also LIST is available.
- The implementation is built upon a highly optimized numerical integration scheme for the evaluation of matrix elements of the Hamiltonian, property integrals involving the charge density, etc. This is the same numerical integration scheme as used in ADF.
- Basis functions are Slater-Type Orbitals (STOs) and/or Numerical Orbitals (NOs).
- The ZlmFit is used to fit the deformation density, which is the difference between the final density and the startup density. The deformation density has zero charge and will in general be small. The fitted deformation density is used for the calculation of the Coulomb potential and the derivatives of the total density (needed for the gradient corrections in the exchange-correlation functionals). In both cases the main part, due to the startup density, is calculated accurately by a numerical procedure, and only the small part from the deformation density is obtained via the fit.
- A frozen core facility is provided to allow efficient treatment of the inner atomic shells.
- Space group symmetry is used to reduce the computational effort in the integrals over the Brillouin zone.

1.3 Release 2017

In comparison to BAND 2016, the BAND 2017 release offers the following new functionality:

- NEGF
    - Transport through a molecular device[*68* (page 209)]. See *Electronic Transport (NEGF)* (page 63).

- Response for 2D (for *NEWRESPONSE* (page 56))
    - Dielectric function for thin films.

- Response for 1D and 3D (for *NEWRESPONSE* (page 56))
    - Enhanced performance for non-symmetric systems.
    - Improved stability for systems with many bands.

- Restart from a smaller basis set
    - Restart a difficult to converge TZP calculation from a SZ one.
    - Restart from the density matrix, useful for NEGF.

- *OEP* (page 24) for meta-GGA
    - Option for the purists.

- Basis sets
    - 1s frozen core basis sets added for the elements: Al, Si, P, S, Cl, and Ar

- Band structure:
    - Fat-Bands for band structure analysis
    - Labels for high-symmetry k-points
    - New BandStructure visualization module

- Phonons
    - Implemented isotopic shift (Atomic masses)
    - Adjusted automatic K-space integration: significant speeds-up for phonons calculations with large super-cells
    - Two-sided numerical differentiation improves the precision of phonon calculations

- Default settings changed
    - *Band Structure* (page 75) and *DOS* (page 73) no longer computed by default
    - element 113 is named Nihonium (Nh) instead of Ununtrium (Uut),
    - element 115 is named Moscovium (Mc) instead of Ununpentium (Uup),
    - element 117 is named Tennessine (Ts) instead of Ununseptium (Uus), and
    - element 118 is named Oganesson (Og) instead of Ununoctium (Uuo), following IUPAC

Apart from this new functionality and performance improvements, certain bugs have been fixed.

INPUT AND OUTPUT

When the program has been installed properly on your machine, you should be able to run it by supplying appropriate input.

In the Examples section you can find sample input files covering most basic options.

Delimiters

An input record may contain several items. The general rule is that each sequence of characters, that does not contain a delimiter, is an entity. Delimiters in this context are:

1. the blank or space character ' '
2. the comma ',' and
3. the equal sign '='.

It is assumed throughout that only characters of the Fortran character set are used.

> **Warning:** Do not use tabs in the input file! The program may *not* see them as delimiters and the effects are hard to predict.

Uppercase and lowercase

Virtually all input items are case-insensitive, but take notice of the obvious exceptions: names of files and directories are case-sensitive.

2.1 Keywords

Input is structured by keywords, or keys. A key is a string of letters, dollar signs ($), and digits. It must not start with a digit. It must not contain any other character, in particular not any blank.

Key-controlled input occurs in two forms: either one record (which contains both the key and/or associated data) or a sequence of records, collectively denoted as a key *block*. The first record of the block specifies the key (and may supply additional information); the block is closed by the 'end-key' code: a record containing only the word 'End'. The other records in the block provide data associated with the key.

The form to be used for a key is not optional: each admissible key corresponds to a particular form. The block form is used for keys which relate to 'lists' of data, such as atomic coordinates or basis functions.

As is the case in the molecular code, the 'title' must be specified explicitly with the key `Title` and may be put on any record of the input file. The input file should end with a record 'End Input', that is, the program reads input until such a record is encountered or until the Fortran end-of-file condition becomes true, whichever comes first.

'End Input' is not a key.

Summarizing, the input file must have the following format:

2.2 Format of the Input file

```
TITLE jobname
   key-controlled data
   key-controlled data
   (et cetera...)
End Input
```

The ordering of keys in input is free and has no consequences.

All numerical information is 'free format': the absolute positions of numbers and the format of reals is irrelevant.

In most cases lowercase characters and capitals can both be used and the program will not discriminate between them, except in filenames.

2.3 Setting up an Input File

We now give a brief guide how to set up input.

You first specify the minimally required information: the definition of the geometry and the basis sets.

- Define the geometry with the key blocks `Atoms` and `Lattice`.
- Specify the `BasisDefaults` key block. With this you can set the quality of the basis set to be used, and the size of the frozen core.
- Terminate the input file by typing 'End Input'

```
Atoms
   Atomnames and fractional/Cartesian coordinates
End

Lattice
   Lattice vectors of the unit cell
End

BasisDefaults
   BasisType ...
   Core      ...
End
```

Then you add keys for all aspects for which you don't want to use the defaults.

- Specify a title using the key `Title`.
- Other keys: `Unrestricted` (if you want to do a spin-unrestricted calculation), `NumericalQuality` (general precision parameter), `KSpace` (parameter for numerical integration over the Brillouin Zone), `DOS` (generation of the density-of-states), and so on.
- Comments: everything behind a exclamation mark "!" is treated as comment. If you put it behind a key, leave some space between it, otherwise it will be seen as part of the key.

Consult the sample runs and description below, to see how you may use the various keys (`Lattice`, `Atoms`, `BasisDefaults`).

2.4 General Input Features

The program supports the following features to enhance user-friendliness of input:

- Arithmetic expressions can be used (wherever numbers are required) involving the standard arithmetic operands in Fortran (`+ - * / **`), together with parentheses where necessary or convenient. Blanks are allowed in expressions and ignored, but they are interpreted as separators, i.e. as denoting the end of an expression, whenever the part before the blank can be evaluated as a correct expression. For instance `3*  4` will be interpreted as 12, but `3   *4` will be interpreted as 3, followed by a character `*`, followed in turn by the number 4. All numbers and results are interpreted and handled as being of type real, but whenever the result is a whole number (allowing for small round-off deviations), it will be recognized as possibly denoting an integer.

- (Single) quotes can be used to designate strings, i.e. (parts of) records which are not to be parsed for expressions, but which are to be taken as they are. The quotes themselves are ignored. Double quotes inside a string are understood to denote the single quote character (as part of the string).

- Empty records and starting blanks in records are allowed, and can be used to enhance clarity and readability of the input file for human readers by structuring its layout.

- A double colon `::` is interpreted as an end of line character, hence the double colon and anything after it is ignored

2.5 Title, Comments and Ignore

Title Specifies the title of this run. The title is used for identification of the result files.

Comment (block-type), and line oriented The content of this key is a text that will be copied to the output header. There, general program information is printed, too. The exclamation mark can also be used to add comments to you input that should not interpreted by BAND. Example:

```
Comment ! here are my comments
   Description of the calculation
   Some more description
End

Lattice ! FCC lattice
   0 a a
   a 0 a
   a a 0
End

! Here are my defines
Define
   a = 3.75
End
```

Finally it is possible to let BAND ignore parts of the input

Ignore (block-type) Suppress reading of input until the next end-key code (END).

2.6 Defining variables and functions

The user may define *variables* and *functions* in the input file, and apply them subsequently in expressions. The input file is read sequentially and variables and functions must be defined before they can be used. Note carefully that

replacement of a variable (or function) by its value will occur wherever possible (textually), even if it leads to nonsense input. A frequently occurring mistake is that the user defines a variable 'C' in the input and then gets it corrupted because of subsequent isolated C characters being replaced by the defined numerical value. *Therefore: avoid single-character variables and function names.* Always check carefully that the identifier you introduce is not 'used' already in the input file. A few variables and functions are pre-defined:

- variables: pi = 3.1415....

- functions: sin, cos, tan, asin, acos, atan, exp, log, sqrt, nint.

The argument list of a function must be enclosed in parentheses, and the arguments (in case of more than one) must be separated by commas. Defining variables and/or functions is done with the block-type key `define`.

Define (block-type) Definition of user-supplied functions and variables that can subsequently be used in the input file. (see the note on auxiliary input features)

Example (part of input):

```
DEFINE
   ab = sin(pi/3)
   s13 = 14*sqrt(2)
   func(x,y,z) = x*ab+y**2-y*z
End
AKEY = FUNC (S13/5, S13/7, SIN(PI/6))
```

Here a function func and variables ab and s13 are defined, using the pre-defined functions sin and sqrt, as well as the pre-defined variable pi. These are then applied to assign a value to the (hypothetical) key `AKEY`.

Note 1: the variable ab is also used in the definition of func! This is allowed because ab is defined before func.

Note 2: variable- and function *names* must have the same form as keywords, i.e. only certain characters are allowed.

Note 3: in the definition of variables and functions blanks are ignored altogether (in the value part) and will not be interpreted as possible separators of the expression that defines the variable or function.

2.7 Execution flow

There are a few ways to alter the standard execution flow, which may be useful to developers or expert users.

StopAfter Specifies that the program is stopped after execution of a specified program-part (subroutine). The specified name should be one of a pre-defined list. The most relevant ones are **gemtry** (all geometrical aspects are checked, symmetry analysis is carried out, and numbers of (symmetry-unique) integration points in real space and in k-space are determined; this part takes only little cpu-time) and **atomic** (in addition to the geometry-part all radial parts of basis- and fit functions are generated, and the spherically symmetric atoms are computed and inserted in the crystal; the initial charge density is defined and integrated (check on integration-precision), and the electrostatic interaction is computed between the unrelaxed free atoms).

SKIP followed by any number of strings (separated by blanks or commas) tells the program to skip certain parts. Should only be used by those who know what they're doing. Recognized are certain pre-defined strings. Useful argument may be **DOS** when that part of the program takes a long time (usually not), or **eigenvalues** (to suppress printing the eigenvalues at the (first and last) SCF cycles).

ALLOW debugging feature to let the program continue even when intermediate results seem to be wrong or very inaccurate. Currently there are only few places in the program where this key is used. Argument to the key is what should be allowed. Possible arguments: **BadIntegration**, **OCC**, **CHARGEERROR**, **DEPENDBASIS**.

2.8 Printed Output

The amount of output in different stages of the program is controlled by print keys, that can be toggled with the key `Print` followed by a key (see description of Print key).

Print One or more strings (separated by blanks or comma's) from a pre-defined set may be typed after the key. This induces printing of various kinds of information, usually only used for debugging and checking. The set of recognized strings frequently changes (mainly expands) in the course of software-developments. Useful arguments may be symmetry, and fit. A list of all important arguments to this key follows:

  Eigens Prints the (complex) coefficients in the form (norm, phase factor) of the eigenvectors with respect to the valence basis. Coefficients with a norm smaller than eigthreshold will be skipped. This threshold can be set with the option eigthreshold x, default x=.01.

  MullikenOverlapPopulations Prints overlap population of all basis functions.

  OrbitalLabels Prints the labels of the orbitals. If you are interested in the labels of an old calculation and you don't want to repeat it completely, you can add the option StopAfter gemtry to your input file.

  OrbPop Prints the Mulliken population per orbital, for all eigenstates. This is the most detailed population analysis that you can get, one for all k-points and for each band. Populations below a certain threshold are ignored. This threshold can be set with the option popthreshold x. By default x =.01.

  BLCKAT Print the information about the distance effects used in the numerical integrals.

  Fit Print more details about the fitting procedure, such as the fit coefficients at each SCF cycle.

  Symmetry Print the symmetry operators.

  OrbLabels Print the list of basis functions

2.8.1 Thresholds

POPTHRESHOLD Threshold for printing Mulliken population terms. Default 1e-2. Works with print orbpop.

EIGTHRESHOLD Components smaller (absolute value) than this parameter (default 1e-2) are not printed in the output of the DOS section, where the breakdown of crystal orbitals in the primitive basis is printed. Works with print eigens.

2.8.2 More options

The following print options only make sense when debugging the code.

Print One or more strings (separated by blanks or comma's) from a pre-defined set may be typed after the key. The following names replace the IRPNT[IPRSE] keys of the previous versions of BAND:

- `PrepNone, PrepMore, PrepDetail` replace IPRNTP 0, 2, 5

- `IntNone, IntMore, IntDetail` replace IPRNTI 0, 2, 5

- `FrmNone, FrmMore, FrmDetail` replace IPRNTR 0, 2, 5

- `SCFNone, SCFMore, SCFDetail` replace IPRNTS 0, 2, 5

- `EigNone, EigMore, EigDetail` replace IPRNTE 0, 2, 5

2.8.3 Debug key

(programmers only)

DEBUG The argument is usually a (list of) subroutine names that you want to debug. (The programmer can check for this in the same way as `PRINT` or `ALLOW` keys.)

GEOMETRY

Let us start with a few examples showing how to define the geometry of your system in BAND.

Sodium chloride crystal:

```
Units
  Length Angstrom
End

Lattice
  0.00 2.82 2.82
  2.82 0.00 2.82
  2.82 2.82 0.00
End

Atoms
  Na  0.00 0.00 0.00
  Cl  2.82 2.82 2.82
End
```

Graphene (2D slab):

```
Units
  Length Angstrom
End

Lattice
  2.46 0.0     0.0
  1.23 2.13042 0.0
End

Atoms
  C  0.0  0.0     0.0
  C  1.23 0.71014 0.0
End
```

Water molecule:

```
Atoms
  O  0.0  0.0   0.0
  H  0.0  1.446 1.137
  H  0.0 -1.446 1.137
End
```

Notes:

- If you leave out the `Lattice` key block, BAND will not use periodic boundary condition
- The default unit of length is Bohr

Geometry from an external file:

```
GeometryFile /SomePath/diamond.xyz
```

3.1 Units

The **default** unit for geometric data (atomic positions and lattice vectors) is **Bohr (atomic units)**. You can change the input units via the optional block key `Units`:

```
Units
  {Length [Bohr|Angstrom]}
  {Angle [Degree|Radian]}
End
```

Length (**Default: Bohr**) Units of length.

Angle (**Default: Degree**) Units of angle.

3.2 Lattice vectors

Lattice (block-type) Vectors (Cartesian coordinates x, y, z) defining the Bravais lattice, one vector per line. The number of lines defines the periodicity of the system: 1 - polymer, 2 - slab, 3 - bulk crystal. If the `Lattice` block is omitted, no periodic boundary conditions will be used.

For 1D periodic systems the lattice vector must be aligned with the x-axis

```
! This is OK (lattice vector aligned with the x-axis):

Lattice
  2.0 0.0 0.0
End
```

```
! This is WRONG (lattice vector not aligned with the x-axis):

Lattice
  1.0 1.0 0.0
End
```

For 2D periodic systems (slabs) the lattice vectors must be on the xy-plane:

```
! This is OK (lattice vectors in the xy-plane):

Lattice
  2.0 0.0 0.0
  1.0 1.0 0.0
End
```

```
! This is WRONG (lattice vector not in the xy-plane):

Lattice
```

```
   2.0 0.0 0.0
   0.0 0.0 2.0
End
```

3.3 Coordinates of atoms in the unit cell

The chemical symbols and coordinates of the atoms in the unit cell must be defined in the `Atoms` block key:

```
Atoms
  O  x1 y1 z1
  H  x2 y2 z2
  H  x3 y3 z3
End
```

The chemical symbol, preceding the coordinates, defines the *atom type*. Be aware that internally BAND will reorder the atoms, so that atoms of the same type are consecutive.

You may want atoms of the same element to belong to different atom types. This can be done by appending a user defined name to the element symbol, using `.` as a separator. Example:

```
Atoms
  O           x1 y1 z1
  H.my_name_1 x2 y2 z2
  H.my_name_2 x3 y3 z3
End
```

This can be useful if you want to:

- Define different basis sets and numerical accuracy for different types, see the *Basis set* (page 31) section and the examples *BasisDefaults* (page 121) and *Multiresolution* (page 123)

- Manipulate the spin, see the *BetaIron* (page 113) example

- Define different *isotopes* (page 14) for the same element

- Mark fragments in *PEDA* (page 78) and *PEDANOCV* (page 79) calculations (see also the *PEDA* (page 181) and *PEDANOCV* (page 184) examples)

3.4 Fractional coordinates

Coordinates Fractional Including this key in the input specifies that nuclear coordinates (key atoms) are given as expansions in the Bravais lattice vectors, rather than in a Cartesian representation. See the *NaCl example* (page 146).

3.5 Selected atoms

SelectedAtoms With this key you can select atoms. This has an effect on a couple of options, like *Gradients* (page 47), *Hessian (FREQUENCIES)* (page 50), *NMR* (page 61), and *EFG* (page 60) options. In these cases, the properties will only be calculated for the selected atoms, which can make the job faster.

```
SelectedAtoms n1, n2, n...
```

The numbers of the atoms are as on input. If the selection breaks the symmetry the program will stop. Leaving out this key in general selects *all* atoms.

3.6 Atomic masses (AtomProps)

The default atomic masses used in BAND are taken from Nuclear Physics A 565, 1-65 (1993) (http://dx.doi.org/10.1016/0375-9474(93)90024-R). The atomic masses of selected atom types can be redefined via the `AtomProps` block key:

```
AtomProps
   Symbol m=mass
   ...
End
```

Symbol Symbol defining the atom type, as defined in the *Atoms block key* (page 13). Here, *mass* defines the new atomic mass (in *unified atomic mass unit*).

Example with three different hydrogen isotopes:

```
Atoms
   N         0.000000    0.000000    0.010272
   H        -0.471582   -0.816803    0.407861
   H.D       0.943163    0.000000    0.407861
   H.T      -0.471582    0.816803    0.407861
End
AtomProps
   H.D m=2.01410
   H.T m=3.01605
End
```

Only the following properties depend on the atomic masses:

- *Frequencies* (page 50)
- *Phonons and thermodynamic properties* (page 53)

3.7 Ghost atoms

A ghost atom represents only the basis set of an atom, there is no nucleus at that position. Ghost atoms are defined by prepending "Gh." to the element name. In the case of a frozen core, the core orbitals of the ghost atoms are added to the valence space.

Example The dimer N-H including basis functions of two hydrogens:

```
Atoms
   N         0.000000    0.000000    0.010272
   H        -0.471582   -0.816803    0.407861
   Gh.H      0.943163    0.000000    0.407861
   Gh.H     -0.471582    0.816803    0.407861
End
```

3.8 Geometry from file

Instead of specifying the geometry of your system in the input file via the *Atoms* (page 13), *Lattice* (page 12) and *Units* (page 12) block keys, the geometry can be read from an external file

```
GeometryFile file
```

file The absolute path of the file containing the geometry information of the system. The only format currently supported is .xyz (https://en.wikipedia.org/wiki/XYZ_file_format). The default unit is Angstrom.

Notes

- The standard .xyz file format does not include lattice vectors. However, we support lattice vectors in .xyz files; our convention is to specify the lattice vectors at the end of the .xyz file via the keys `VEC1`, `VEC2` and `VEC3`. For 1D periodic systems only VEC1 is needed. For 2D periodic systems only `VEC1` and `VEC2` are needed. Example: .xyz file for graphene:

```
2

C    0.0  0.0     0.0
C    1.23 0.71014 0.0
VEC1 2.46 0.0     0.0
VEC2 1.23 2.13042 0.0
```

- If you specify the geometry via the `GeometryFile` key, the following keys must not appear in the input: *Atoms* (page 13), *Lattice* (page 12) and *Units* (page 12).

MODEL HAMILTONIANS

The program calculates by default the non-relativistic, self-consistent field (SCF) solution from the potential in the spin-restricted local density approximation (LDA) to the exchange-correlation (XC) potential. The post-SCF generalized gradient (GGA) corrections for the exchange and the correlation energies (each for a few different functionals) are calculated and printed on output. The program can be instructed to use the GGA corrections in the potential during the SCF as well. This will in general have little effect on the formation energy, but it may affect the Density of States (DOS).

4.1 Density Functional

The starting point for the XC functional is usually the result for the homogeneous electron gas, after which the so called non-local or generalized gradient approximation (GGA) can be added.

4.1.1 XC functionals

The density functional approximation is controlled by the XC key. `XC (block-type)`

Three classes of XC functionals are supported: LDA, GGA, meta-GGA, and range-separated hybrid functionals. There is also the option to add an empirical dispersion correction. The only ingredient of the LDA energy density is the (local) density, the GGA depends additionally on the gradient of the density, and the meta-GGA has an extra dependency on the kinetic energy density. The range-separated hybrids are explained below in the section *Range-Separated Hybrids* (page 22).

In principle you may specify different functionals to be used for the *potential,* which determines the self-consistent charge density, and for the *energy* expression that is used to evaluate the (XC part of the) energy of the charge density. The *energy* functional is used for the nuclear gradients (geometry optimization), too. To be consistent, one should generally apply the same functional to evaluate the potential and energy respectively. Two reasons, however, may lead one to do otherwise:

1. The evaluation of the GGA part (especially for meta-GGAs) in the *potential* is rather time-consuming. The effect of the GGA term in the potential on the self-consistent charge density is often not very large. From the point of view of computational efficiency it may, therefore, be attractive to solve the SCF equations at the LDA level (i.e. not including GGA terms in the potential), and to apply the full expression, including GGA terms, to the energy evaluation *a posteriori*: post-SCF.

2. A particular XC functional may have only an implementation for the potential, but not for the energy (or vice versa). This is a rather special case, intended primarily for fundamental research of Density Functional Theory, rather than for run-of-the-mill production runs.

All subkeys of `XC` are optional and may occur twice in the data block: if one wants to specify different functionals for potential and energy evaluations respectively, see above.

```
XC
   {LDA  {Apply}   LDA {Stoll}}
   {GGA  {Apply}   GGA}
   {DiracGGA   GGA}
   {MetaGGA  {Apply}   GGA}
   {Dispersion {s6scaling} {RSCALE=r0scaling} {Grimme3} {BJDAMP} {PAR1=par1}
↪{PAR2=par2} {PAR3=par3} {PAR4=par4}}
   {Model [LB94|TB-mBJ|KTB-mBJ|JTS-MTB-MBJ|GLLB-SC]}
   {SpinOrbitMagnetization [None|NonCollinear|CollinearX|CollinearY|CollinearZ]}
   {LibXC {Functional}}
End
```

The common use is to specify either an LDA or a (meta)GGA line. (Technically it is possible to have an LDA line *and* a GGA line, in which case the LDA part of the GGA functional (if applicable) is replaced by what is specified by the LDA line.)

Apply States whether the functional defined on the pertaining line will be used self-consistently (in the SCF-potential), or only post-SCF, i.e. to evaluate the XC energy corresponding to the charge density. The value of apply must be **SCF** or **POSTSCF**. (**default=SCF**)

LDA/GGA/metaGGA

LDA Defines the LDA part of the XC functional and can be any of the following:

Xonly: The pure-exchange electron gas formula. Technically this is identical to the Xalpha form with a value 2/3 for the X-alpha parameter.

Xalpha: the scaled (parameterized) exchange-only formula. When this option is used you may (optionally) specify the X-alpha *parameter* by typing a numerical value after the string Xalpha (**Default: 0.7**).

VWN: the parameterization of electron gas data given by Vosko, Wilk and Nusair (ref [*1* (page 205)], formula version V). Among the available LDA options this is the more advanced one, including correlation effects to a fair extent.

Stoll: For the VWN or GL variety of the LDA form you may include Stoll's correction [*2* (page 205)] by typing Stoll on the same line, after the main LDA specification. You must not use Stoll's correction in combination with the Xonly or the Xalpha form for the Local Density functional.

GGA Specifies the GGA part of the XC Functional. It uses derivatives (gradients) of the charge density. Separate choices can be made for the GGA exchange correction and the GGA correlation correction respectively. Both specifications must be typed (if at all) on the same line, after the GGA subkey.

For the exchange part the options are:

- **Becke**: the gradient correction proposed in 1988 by Becke [*3* (page 205)]

- **PW86x**: the correction advocated in 1986 by Perdew-Wang [*4* (page 205)]

- **PW91x**: the exchange correction proposed in 1991 by Perdew-Wang [*5* (page 205)]

- **mPWx**: the modified PW91 exchange correction proposed in 1998 by Adamo-Barone [*27* (page 206)]

- **PBEx**: the exchange correction proposed in 1996 by Perdew-Burke-Ernzerhof [*12* (page 205)]

- **HTBSx**: the HTBS exchange functional [*43* (page 207)]

- **RPBEx**: the revised PBE exchange correction proposed in 1999 by Hammer-Hansen-Norskov [*13* (page 205)]

- **revPBEx**: the revised PBE exchange correction proposed in 1998 by Zhang-Yang [*28* (page 206)]

- **mPBEx**: the modified PBE exchange correction proposed in 2002 by Adamo-Barone [*29* (page 206)]

- **OPTX**: the OPTX exchange correction proposed in 2001 by Handy-Cohen [*30* (page 206)]

For the correlation part the options are:

- **Perdew**: the correlation term presented in 1986 by Perdew [*6* (page 205)]

- **PBEc**: the correlation term presented in 1996 by Perdew-Burke-Ernzerhof [*12* (page 205)]

- **PW91c**: the correlation correction of Perdew-Wang (1991), see [*5* (page 205)], [*8* (page 205)], [*9* (page 205)]

- **LYP**: the Lee-Yang-Parr 1988 correlation correction [*7* (page 205)]

Some GGA options define the exchange and correlation parts in one stroke. These are:

- **BP86**: this is equivalent to **Becke + Perdew** together

- **PW91**: this is equivalent to **pw91x + pw91c** together

- **mPW**: this is equivalent to **mPWx + pw91c** together

- **PBE**: this is equivalent to **PBEx + PBEc** together

- **HTBS**: this is equivalent to **HTBSx + PBEc** together

- **RPBE**: this is equivalent to **RPBEx + PBEc** together

- **revPBE**: this is equivalent to **revPBEx + PBEc** together

- **mPBE**: this is equivalent to **mPBEx + PBEc** together

- **BLYP**: this is equivalent to **Becke** (exchange) + **LYP** (correlation)

- **OLYP**: this is equivalent to **OPTX** (exchange) + **LYP** (correlation)

- **OPBE**: this is equivalent to **OPTX** (exchange) + **PBEc** (correlation) [*31* (page 206)]

DiracGGA (Expert option!) This key handles which XC functional is used during the Dirac calculations of the reference atoms. A string is expected which is not restricted to names of GGAs but can be LDA-like functionals, too.

Note: In some cases using a GGA functional leads to slow convergence of matrix elements of the kinetic energy operator w. r. t. the `Accuracy` parameter. Then one can use the LDA potential for the calculation of the reference atom instead.

MetaGGA Key to select the evaluation of a meta-GGA. A byproduct of this option is that the bonding energies of all known functionals are printed (using the same density). Meta-GGA calculations can be time consuming, especially when active during the SCF.

Self consistency of the meta-GGA is implemented as suggested by Neuman, Nobes, and Handy.[*11* (page 205)]

The available functionals of this type are:

- **TPSS**: The 2003 meta-GGA [*15* (page 205)]

- **M06L**: The meta-GGA as developed by the Minesota group [*16* (page 206)]

- **revTPSS**: The 2009 revised meta-GGA [*26* (page 206)]

Dispersion Correction

In BAND parameters for *Grimme3* and *Grimme3 BJDAMP* can be used according to version 3.1 (Rev. 1) of the coefficients, published on the Bonn .

`DISPERSION Grimme3 BJDAMP {PAR1=par1 PAR2=par2 PAR3=par3 PAR4=par4}` If this key is present a dispersion correction (DFT-D3-BJ) by Grimme [*42* (page 207)] will be added to the total bonding energy, gradient and second derivatives, where applicable. Parametrizations are implemented e.g. for B3LYP, TPSS, BP86, BLYP, PBE, PBEsol [*14* (page 205)] , and RPBE. It has four parameters. One can override these using *PAR1=.. PAR2=..*, etc. In the table the relation is shown between the parameters and the real parameters in the dispersion correction.

variable	variable on Bonn
PAR1	s6
PAR2	a1
PAR3	s8
PAR4	a2

`DISPERSION Grimme3 {PAR1=par1 PAR2=par2 PAR3=par3}` If this key is present a dispersion correction (DFT-D3) by Grimme [*41* (page 207)] will be added to the total bonding energy, gradient and second derivatives, where applicable. Parametrizations are available e.g. for B3LYP, TPSS, BP86, BLYP, revPBE, PBE, PBEsol [*14* (page 205)], and RPBE, and will be automatically set if one of these functionals is used. For all other functionals, PBE-D3 parameters are used as default. You can explicitly specify the three parameters.

variable	variable on Bonn
PAR1	s6
PAR2	sr,6
PAR3	s8

`Dispersion {s6scaling RSCALE=r0scaling}` If the DISPERSION keyword is present a dispersion correction will be added to the total bonding energy, where applicable. By default the correction of Grimme is applied.[*32* (page 207)] The term is added to the bonding energies of all printed functionals, here the LDA and a couple of GGAs are meant. The global scaling factor, with which the correction is added, depends on the XC functional used for SCF but it can be modified using the *s6scaling* parameter. The following scaling factors are used (with the XC functional in parentheses): 1.20 (BLYP), 1.05 (BP), 0.75 (PBE), 1.05 (B3LYP). In all other cases a factor 1.0 is used unless modified via the s6scaling parameter. The van der Waals radii, used in this implementation, are hard-coded. However, it is possible to modify the global scaling parameter for them using the *RSCALE=r0scaling* argument. The default value is 1.1 as proposed by Grimme.[*32* (page 207)]

Model Potentials

`Model` Some functionals give only a potential and have no energy expression. We call such functionals model potentials. In BAND the following model potentials are available:

LB94 With this model the asymptotically correct potential of van Leeuwen and Baerends is invoked.[*10* (page 205)]

TB-mBJ This model potential can be used to correct for the band gap problem with GGAs for bulk systems.[*44* (page 207)] This potential depends on a c-factor for which there is a density dependent automatic expression. However you can override the automatic value by specifying XC%TB_mBJCFactor cfac. In principle: the bigger the value the larger the gap. **KTB-mBJ/JTS-mTB-mBJ** are variations of **TB-mBJ**. The formula for C contains three parameters: A,B, and E. The logic is as follows

potential	A	B	E
TB-mBJ[*44* (page 207)]	-0.012	1.023	0.5
KTB-mBJ[*54* (page 208)]	0.267	0.656	1.0
JTS-mTB-mBJ[*55* (page 208)]	0.4	1.0	0.5

The three parameters (A,B, and E) can be user-defined set as follows:

```
XC
    Model TB_mBJ
```

```
    TB_mBJAFactor valA
    TB_mBJBFactor valB
    TB_mBJEFactor valE
End
```

GLLB-SC This parameter-free functional models the derivative discontinuity.[*49* (page 208)]

Non-Collinear Approach

SpinOrbitMagnetization (**Default=CollinearZ**) Most XC functionals have as one ingredient the spin polarization. Normally the direction of the spin quantization axis is arbitrary and conveniently chosen to be the *z*-axis. However, in a *spin-orbit* (page 25) calculation the direction matters, and it is arbitrary to put the z-component of the magnetization vector into the XC functional. It is also possible to plug the size of the magnetization vector into the XC functional. This is called the non-collinear approach. There is also the exotic option to choose the quantization axis along the *x* or *y* axis. To summarize, the value **NonCollinear** invokes the non-collinear method. The other three option **CollinearX**, **CollinearY** and **CollinearZ** causes either the x, y, or z component to be used as spin polarization for the XC functional.

LibXC Library Integration

LibXC functional LibXC is a library of approximate XC functionals, see Ref. [*63* (page 208)]. The development version 3 of LibXC is used. See the LibXC website for the complete list of functionals: http://www.tddft.org/programs/Libxc.

The following functionals can be evaluated with LibXC (incomplete list):

- **LDA:** LDA, PW92, TETER93

- **GGA:** AM05, BGCP, B97-GGA1, B97-K, BLYP, BP86, EDF1, GAM, HCTH-93, HCTH-120, HCTH-147, HCTH-407, HCTH-407P, HCTH-P14, PBEINT, HTBS, KT2, MOHLYP, MOHLYP2, MPBE, MPW, N12, OLYP, PBE, PBEINT, PBESOL, PW91, Q2D, SOGGA, SOGGA11, TH-FL, TH-FC, TH-FCFO, TH-FCO, TH1, TH2, TH3, TH4, VV10, XLYP, XPBE

- **MetaGGA:** B97M-V, M06-L, M11-L, MN12-L, MS0, MS1, MS2, MVS, PKZB, TPSS

- **Hybrids** (**only for non-periodic systems**): B1LYP, B1PW91, B1WC, B3LYP, B3LYP*, B3LYP5, B3LYP5, B3P86, B3PW91, B97, B97-1 B97-2, B97-3, BHANDH, BHANDHLYP, EDF2, MB3LYP-RC04, MPW1K, MPW1PW, MPW3LYP, MPW3PW, MPWLYP1M, O3LYP, OPBE, PBE0, PBE0-13, REVB3LYP, REVPBE, RPBE, SB98-1A, SB98-1B, SB98-1C, SB98-2A, SB98-2B, SB98-2C, SOGGA11-X, SSB, SSB-D, X3LYP

- **MetaHybrids** (**only for non-periodic systems**): B86B95, B88B95, BB1K, M05, M05-2X, M06, M06-2X, M06-HF, M08-HX, M08-SO, MPW1B95, MPWB1K, MS2H, MVSH, PW6B95, PW86B95, PWB6K, REVTPSSH, TPSSH, X1B95, XB1K

- **Range-separated** (**for periodic systems, only short range-separated functionals can be used**, see *Range-separated hybrid functionals* (page 22)): CAM-B3LYP, CAMY-B3LYP, HJS-PBE, HJS-PBESOL, HJS-B97X, HSE03, HSE06, LRC_WPBE, LRC_WPBEH, LC-VV10, LCY-BLYP, LCY-PBE, M11, MN12-SX, N12-SX, TUNED-CAM-B3LYP, WB97, WB97X, WB97X-V

Example usage for the MVS functional:

```
XC
  LibXC MVS
End
```

Notes:

- **All electron basis sets should be used** (see `CORE NONE` in section *Basis set* (page 31)).

- For periodic systems only short range-separated functionals can be used (see *Range-separated hybrid functionals* (page 22))

- In case of LibXC the output of the BAND calculation will give the reference for the used functional, see also the LibXC website http://www.tddft.org/programs/Libxc.

- Do not use any of the subkeys LDA, GGA, METAGGA, MODEL in combination with the subkey LIBXC.

- One can use the DISPERSION key icw LIBXC. For a selected number of functionals the optimized dispersion parameters will be used automatically, please check the output in that case.

Range-separated hybrid functionals

Short range-separated hybrid functionals, like the **HSE03** functional [*62* (page 208)], can be useful for prediction of more accurate band gaps compared to GGAs. These must be specified via the *LibXC* (page 21) key

```
XC
   LibXC functional {omega=value}
End
```

functional The functional to be used. (Incomplete) list of available functionals: **HSE06**, **HSE03**, **HJS-B97X**, **HJS-PBE** and **HJS-PBESOL** (See the LibXC website (http://www.tddft.org/programs/octopus/wiki/index.php/Libxc_functionals) for a complete list of available functionals).

omega *Optional*. You can optionally specify the switching parameter omega of the range-separated hybrid. Only possible for the **HSE03** and **HSE06** functionals (See [*62* (page 208)]).

Notes:

- Hybrid functionals can only be used in combination with all-electron basis sets (see `CORE NONE` in section *Basis set* (page 31)).

- The Hartree-Fock exchange matrix is calculated through a procedure known as Resolution of the Identity (RI). See *RIHartreeFock* (page 44) key.

- Regular hybrids (such as B3LYP) and long range-separated hybrids (such as CAM-B3LYP) **cannot** be used in periodic boundary conditions calculations (they can only be used for non-periodic systems).

- There is some confusion in the scientific literature about the value of the switching parameter ω for the HSE functionals. In LibXC, and therefore in BAND, the HSE03 functional uses $\omega = 0.106066$ while the HSE06 functional uses $\omega = 0.11$.

Usage example:

```
XC
   LibXC HSE06 omega=0.1
End
```

Defaults and special cases

- If the `XC` key is not used, the program will apply only the Local Density Approximation (no GGA terms). The chosen LDA form is then VWN.

- If only a GGA part is specified, omitting the *LDA* subkey, the LDA part defaults to VWN, except when the LYP correlation correction is used: in that case the LDA default is Xonly: pure exchange.

- The reason for this is that the LYP formulas assume the pure-exchange LDA form, while for instance the Perdew-86 correlation correction is a correction to a *correlated* LDA form. The precise form of this correlated LDA form assumed in the Perdew-86 correlation correction is not available as an option in ADF but the VWN formulas are fairly close to it.

- Be aware that typing only the subkey *LDA*, without an argument, will activate the VWN form (also if LYP is specified in the GGA part).

4.1.2 GGA+U

A special way to treat correlation is with so-called LDA+U, or GGA+U calculations. It is intended to solve the band gap problem of traditional DFT, the problem being an underestimation of band gaps for transition-metal complexes. A Hubbard like term is added to the normal Hamiltonian, to model on-site interactions. In its very simplest form it depends on only one parameter, U, and this is the way it has been implemented in BAND. The energy expression is equation (11) in the work of Cococcioni.[*47* (page 207)] See also the review article [*46* (page 207)].

HubbardU (block-type) Key to control the GGA+U calculation:

```
HubbardU
   Enabled          [false | true]
   UValue           u1 u2 ...
   lValue           l1 l2 ...
   {PrintOccupations [true | false]}
End
```

Enabled (**Default: false**) Whether or not to apply the Hubbard Hamiltonian

UValue For each atom type specify the U value [atomic units]. A value of 0.0 is interpreted as no U.

lValue For each atom type specify the l value (0 - s orbitals, 1 - p orbitals, 2 - d orbitals). A negative value is interpreted as no l-value.

PrintOccupations (**Default: true**) Whether or not to print the occupations during the SCF.

An example to apply LDA+U to the d-orbitals of NiO looks like

```
HubbardU
   printOccupations true
   Enabled          true
   uvalue           0.3 0.0
   lvalue           2   -1
End

ATOMS Ni
   0.000  0.000  0.000
END

ATOMS O
   2.085  2.085  2.085
END
```

4.1.3 Spin polarization

By default BAND calculations are spin-restricted. To perform a spin-unrestricted calculation you should include the `Unrestricted` key:

```
UNRESTRICTED
```

Be aware that spin-unrestricted calculations are computationally roughly twice as expensive as spin-restricted.

4.1.4 OEP

(Expert options) When you are using a meta-GGA you are by default using a generalized Kohn-Sham method. However, it is possible to calculate a local potential, as is required for a strict Kohn-Sham calculation, via OEP, (see [*67* (page 209)]).

The main options are controlled with the `MetaGGA` subkey of the XC block if `OEP` is present.

```
XC
    [...]
    MetaGGA GGA OEP {approximation} {Fit} {Potential}
    [...]
End
```

GGA specifies the name of the used meta-GGA. In combination with OEP only **PBE**, **TPSS**, **MVS**, **MS0**, **MS1**, **MS2**, and **SCAN** can be used!

approximation (**Default: KLI**) There are three flavors to approximate the OEP: **KLI**, **Slater**, and **ELP**

Fit By adding the string **Fit** on this line, one uses the fitted density instead of the exact density for the evaluation.

Potential If not specified, only the tau-dependent part of the OEP is evaluated and used. By adding the string **Potential** in addition the tau-independent part is added to the XC potential. (This is needed e.g. for plotting the 'vxc')

With the following subkeys of the `XC` blockkey you have extra control over the iterative OEP evaluation:

MGGAOEPMaxIter (**Default: 30**) defines the maximum number of cycles for the iterative OEP evaluation.

MGGAOEPConvergence (**Default: 1E-6**) defines convergence criterion for OEP evaluation.

MGGAOEPWaitIter (**Default: 0**) defines the number of SCF cycles with the regular meta-GGA before switching to the OEP scheme.

MGGAOEPMaxAbortIter (**Default: 0**) defines number of cycles for which the error is allowed to increase before the calculation is aborted. Here, zero means: do never abort.

MGGAOEPMaxErrorIncrease (**Default: 0.0**) defines the maximum rate of increasing error before the calculation is aborted. Here, zero means: do never abort.

An example for an OEP metaGGA calculation

```
XC
    MetaGGA MVS OEP
End
```

Note that a very fine Becke grid is needed.

```
BeckeGrid
    Quality USER
    UserRadMulFactor 20.0
    UserCoreL 11
    UserInter1L 13
    UserInter2L 21
    UserExterL 31
```

```
   UserExterLBoost 35
 End
```

Note also: the gaps are typically not closer to experiment, and the calculations are more expensive. This option is mainly about academic interest.

4.2 Relativistic Effects

Relativistic effects are treated with the accurate and efficient ZORA approach [*17* (page 206), *18* (page 206)], controlled by the `Relativistic` keyword. Relativistic effects are negligible for light atoms, but grow to dramatic changes for heavy elements. A rule of thumb is: Relativistic effects are quite small for elements of row 4, but very large for row 6 elements (and later).

```
Relativistic [None|ZORA|ZORA Spin]
```

This key instructs BAND to take relativistic effects into account. (**Default: none**)

None No relativistic effects

ZORA Scalar relativistic ZORA. This option comes at very little cost.

ZORA Spin Spin-orbit coupled ZORA. This is the best level of theory, but it is (4-8 times) more expensive than a normal calculation. Spin-orbit effects are generally quite small, unless there are very heavy atoms in your system, especially with p valence electrons (like Pb). See also the *SpinOrbitMagnetization* (page 21) key.

4.3 Solvation

4.3.1 COSMO: Conductor like Screening Model and the Solvation-key

You can study chemistry in solution, as contrasted to the gas phase, with the implementation in BAND of the Conductor like Screening Model (COSMO) of solvation.[*36* (page 207)]

In the COSMO model all solvents are roughly the same, and approximated by an enveloping metal sheet. One explicit dependency on the solvent is that the solvation energy is scaled by

$$f(\epsilon) = \frac{\epsilon - 1}{\epsilon + \chi}$$

and this depends on the dielectric constant of the solvent, and an empirical factor χ. The other is that the shape of the surface is influenced by the *Rad* parameter, see below. SOLVATION (block type)

```
SOLVATION
   {Surf [Delley|Wsurf|Asurf|Esurf|Klamt]}
   {Solv {Name=solvent} {Eps=78.4} {Rad=1.4} {Del=1.4} {Emp=0.0}}
   {Div {Ndiv=3} {NFdiv=1} {Min=0.5} {OFAC=0.8} {leb1=23} {leb2=29} {rleb=1.5}}
   {RADII
     name1=value1
     name2=value2
   subend}
   {Charge {Method=meth} {Conv=1e-10} (Iter=1000} {NoCorr}}
   {Cvec [Exact|FitPot]}
   {SCF [Var|Pert]}
End
```

Presence of the `SOLVATION` key block triggers the solvent calculation and does not require additional data. With subkeys you can customize various aspects of the model, for instance to specify the type of solute. None of the subkeys is obligatory

See also the subkey *StaticCosmoSurface* (page 48) in the `GEOOPT` block and the subkey *StaticCosmoSurface* (page 50) in the `FREQUENCIES` block.

Below follows a description of the subkeys:

Surf [Delley|Wsurf|Asurf|Esurf|Klamt] (**Default: Delley**) Five different cavity types are available. The Wsurf, Asurf, and Esurf surfaces are constructed with the GEPOL93 algorithm.[*38* (page 207)]

   Wsurf Wsurf triggers the Van der Waals surface (VdW), which consists of the union of all atomic spheres.

   Asurf Asurf gives the Solvent-Accessible-Surface (SAS). This is similar to VdW but consists of the path traced by the center of a spherical solvent molecule rolling about the VdW surface or, equivalently, a VdW surface created by atomic spheres to which the solvent radius has been added. These two surface types contain cusps at the intersection of spheres.

   Esurf Esurf gives the Solvent-Excluding-Surface (SES), which consists of the path traced by the *surface* of a spherical solvent molecule rolling about the VdW surface. Primarily, this consists of the VdW surface but in the regions where the spheres would intersect, the concave part of the solvent sphere replaces the cusp. This SES surface is the default.

   Klamt The fourth surface option is Klamt as described in [*36* (page 207)]. It excludes the cusp regions also.

   Delley The fifth surface is the so called Delley surface.[*39* (page 207)]

Solv Solvent details.

   Del Del is the value of Klamt's delta_sol parameter, only relevant in case of Klamt surface.

   Emp Emp addresses the empirical scaling factor x for the energy scaling.

   Name, Eps, Rad Eps specifies the dielectric constant (the default relates to water). An infinite value for Eps is chosen if Eps is specified to be lower than 1.0. Rad specifies the radius of the (rigid sphere) solvent molecules, in Angstrom. Instead of specifying Eps and Rad one can specify a solvent name or formula after 'name='. The following table lists names and formulas that are recognized with the corresponding values for Eps and Rad. The names and formulas are case-insensitive.

Name	Formula	Eps	Rad
AceticAcid	CH3COOH	6.19	2.83
Acetone	CH3COCH3	20.7	3.08
Acetonitrile	CH3CN	37.5	2.76
Ammonia	NH3	16.9	2.24
Aniline	C6H5NH2	6.8	3.31
Benzene	C6H6	2.3	3.28
BenzylAlcohol	C6H5CH2OH	13.1	3.45
Bromoform	CHBr3	4.3	3.26
Butanol	C4H9OH	17.5	3.31
isoButanol	(CH3)2CHCH2OH	17.9	3.33
tertButanol	(CH3)3COH	12.4	3.35
CarbonDisulfide	CS2	2.6	2.88
CarbonTetrachloride	CCl4	2.2	3.37
Chloroform	CHCl3	4.8	3.17
Cyclohexane	C6H12	2	3.5
Cyclohexanone	C6H10O	15	3.46
Dichlorobenzene	C6H4Cl2	9.8	3.54
Continued on next page			

Table 4.1 – continued from previous page

DiethylEther	(CH3CH2)2O	4.34	3.46
Dioxane	C4H8O2	2.2	3.24
DMFA	(CH3)2NCHO	37	3.13
DMSO	(CH3)2SO	46.7	3.04
Ethanol	CH3CH2OH	24.55	2.85
EthylAcetate	CH3COOCH2CH3	6.02	3.39
Dichloroethane	ClCH2CH2Cl	10.66	3.15
EthyleneGlycol	HOCH2CH2OH	37.7	2.81
Formamide	HCONH2	109.5	2.51
FormicAcid	HCOOH	58.5	2.47
Glycerol	C3H8O3	42.5	3.07
HexamethylPhosphoramide	C6H18N3OP	43.3	4.1
Hexane	C6H14	1.88	3.74
Hydrazine	N2H4	51.7	2.33
Methanol	CH3OH	32.6	2.53
MethylEthylKetone	CH3CH2COCH3	18.5	3.3
Dichloromethane	CH2Cl2	8.9	2.94
Methylformamide	HCONHCH3	182.4	2.86
Methypyrrolidinone	C5H9NO	33	3.36
Nitrobenzene	C6H5NO2	34.8	3.44
Nitrogen	N2	1.45	2.36
Nitromethane	CH3NO2	35.87	2.77
PhosphorylChloride	POCl3	13.9	3.33
IsoPropanol	(CH3)2CHOH	19.9	3.12
Pyridine	C5H5N	12.4	3.18
Sulfolane	C4H8SO2	43.3	3.35
Tetrahydrofuran	C4H8O	7.58	3.18
Toluene	C6H5CH3	2.38	3.48
Triethylamine	(CH3CH2)3N	2.44	3.81
TrifluoroaceticAcid	CF3COOH	42.1	3.12
Water	H2O	78.39	1.93

DIV

Ndiv, NFdiv Ndiv controls how fine the spheres, that in fact describe the surface, are partitioned in small surface triangles, each containing one point charge to represent the polarization of the cavity surface. Default division level for triangles Ndiv=3. Default final division level for triangles NFdiv=1 (NFdiv $\leq$ Ndiv).

Min (**Default: 0.5**) Min specifies the size, in Angstrom, of the smallest sphere that may be constructed by the SES surface. For VdW and SAS surfaces it has no meaning.

Ofac (**Default: 0.8**) Ofac is a maximum allowed overlap of new created spheres, in the construction procedure.

leb1, leb2, rleb For the Delley type of construction one needs to set the variables leb1 (**Default: 23**), leb2 (**Default: 29**), and rleb (**Default: 1.5**) to set the number of surface points. If the cavity radius of an atom is lower than rleb use leb1, otherwise use leb2. These values can be changed: using a higher value for leb1 and leb2 gives more surface points (maximal value leb1, leb2 is 29). A value of 23 means 194 surface points in case of a single atom, and 29 means 302 surface points in case of a single atom Typically one could use leb1 for the surface point of H, and leb2 for the surface points of other elements.

Radii In the Radii data block you give a list of atom types and values:

```
SOLVATION
   . . .
   Radii
       H 0.7
       C 1.3
       . . .
   Subend
   . . .
End
```

The values are the radii of the atomic spheres, in the default unit of length (Angstrom or Bohr). If not specified the default values are those by Allinger.[*40* (page 207)]

Charge This addresses the determination of the (point) charges that model the cavity surface polarization. In COSMO calculations you compute the surface point charges q by solving the equation $A \cdot q = -f$, where f is the molecular potential at the location of the surface charges q and A is the self-interaction matrix of the charges. The number of charges can be substantial and the matrix A hence very large. A direct method, i.e. inversion of A, may be very cumbersome or even impossible due to memory limitations, in which case you have to resort to an iterative method. *Meth* (**Default: INVER**) specifies the equation-solving algorithm. The option *INVER* requests direct inversion. The option *CONJ* uses the preconditioned biconjugate gradient method. This is guaranteed to converge and does not require huge amounts of memory. CONV and ITER are the convergence criterion and the maximum number of iterations for the iterative method. In periodic calculations the unit cell needs to be neutral. Therefore, by default the COSMO charges are forced to sum to zero. When NOCORR is present, this constraint is not applied.

CVec (**Default: Exact**) There are two ways to calculate the molecule-(COSMO)surface interaction. EXACT: Consider the interactions between the electron density and the potential due to the COSMO points. FITPOT: Use the COSMO point charges in the potential of the molecule (obtained by density fitting).

The EXACT option is exact on paper, but in reality numerical integration is used to get the energy. The standard grid is formally not suitable to handle the cusps at the COSMO points. Still this option works well because the singularity is softened by the smallness of the electronic density at the COSMO points. The EXACT option is much faster when doing a geometry optimization.

SCF (**Default: VAR**) The option VAR is to apply the external potential of the COSMO surface during the SCF. To apply COSMO only post SCF you can use the PERT option.

4.3.2 Additional keys for periodic systems

PeriodicSolvation (block-type) For the simulation of periodic structures additional information might be necessary. Therefore, the `PeriodicSolvation` block can be used.

```
PeriodicSolvation
   {RemovePointsWithNegativeZ [True|False]}
   {Nstar integer}
End
```

RemovePointsWithNegativeZ (**Default: false**) This option, expecting a boolean value (true or false), handles whether a COSMO surface is constructed on both sides of a surface. If one is only interested in the solvation effect on the upper side of a surface then this option should be set to true.

Nstar (**Default: 4**) This option, expecting an integer number (>2), handles the accuracy for the construction of the COMSO surface. The larger the given number the more accurate the construction.

General remarks: The accuracy of the result and the calculation time is influenced by the screening radius `SCREENING%RMADEL` (see *Screening* (page 45)). If the calculation does take too long, defining a smaller radius does help. **But:** too small radii, especially smaller than the lattice constants, will give unphysical results.

4.4 Electric and Magnetic Static Fields

EField Include a homogeneous, static, electric field in the z-direction (only possible for 0D, 1D or 2D periodic systems):

```
EField
   Ez rval
   {Unit [Volt/Angstrom | a.u. | Volt/Bohr | Volt/Meter]}
End
```

Ez This option expects a real value which is the strength of the electric field.

Unit (**Default: Volt/Angstrom**) This options specifies the unit of the electric field.

The static field is explicitly handled in the determination of the orbital coefficients, energy, and gradients. If you apply it to any other property, such as the NMR shielding tensor, dielectric function, solvation energy, etc., the result is probably not entirely correct. In case of doubt contact the SCM staff.

FuzzyPotential Atomic (fuzzy cell) based, external, electric potential

```
FuzzyPotential
   scale $scale
   a1 v1   ! atom with index a1 gets potential coefficient v1 (a.u.)
   a2 v2   ! atom a2 gets potential v2
   ...
End
```

scale Overall scaling factor to be applied.

If an atom is not in the list it gets a coefficient of zero. The potential of an atom is its number (v_i) as specified on input times its fuzzy cell

$$V(r) = \sum_i^{\text{atoms}} v_i \mathcal{P}_{i,U}(r)$$

using the same partition function $\mathcal{P}$ as for the *BeckeGrid* (page 37). A partition function (or fuzzy cell) of an atom is close to one in the neighborhood of this atom.

The sign convention is: negative is favorable for electrons. (Unit: a.u.)

BField The effect of a magnetic filed can be **approximated** by the following potential: $\mu_B \vec{\sigma}_i \vec{B}$, where μ_B is the Bohr magneton, $\vec{\sigma}_i$ are the Pauli matrices and $\vec{B}$ is the magnetic field. For *Spin-unrestricted collinear* (page 23) calculations, the spin is assumed to be aligned with the z-axis.

```
BField
   Bx rval
   By rval
   Bz rval
End
```

Bz,By,Bz (**Units: Tesla**) X,Y and Z components of the magnetic field.

4.5 Nuclear model

Normally the nucleus is modeled as a point charge.

NuclearModel (**Default: PointCharge**) Specify what model to use for the nucleus:

```
NuclearModel [PointCharge | Gaussian | Uniform]
```

One can choose between the standard point charge model and a Gaussian model. For the Gaussian model the nuclear radius is calculated according to the work of Visscher and Dyall.[*50* (page 208)]

The static field is explicitly handled in the determination of the orbital coefficients, energy, and gradients. If you apply it to any other property, such as the NMR shielding tensor, dielectric function, solvation energy, etc., the result is probably not entirely correct. In case of doubt contact the SCM staff.

ACCURACY AND EFFICIENCY

Given a chosen Model Hamiltonian, the three main issues that influence the accuracy of a calculation are: *integration in real space* (page 37), *integration in reciprocal space* (page 39), and the quality of the *basis set* (page 31).

General Numerical Quality

The key `NumericalQuality` sets the quality of several important technical aspects of a BAND calculation (with the notable exception of the *basis set* (page 31))

```
NumericalQuality [Basic | Normal | Good | VeryGood | Excellent]
```

NumericalQuality (**Default: Normal**) This key sets the default Quality of the *Becke grid* (page 37), *ZlmFit* (page 40), *k-space integration* (page 39), and *SoftConfinement* (page 36). Possible values are Basic, Normal, Good, VeryGood, and Excellent. For the user this is the most convenient way to influence the precision, and estimate the error. It is always possible to tweak things more, as is explained below.

Defaults Convention

There has been a big change in many default settings with the 2014 release. To switch between the two styles

```
DefaultsConvention [Pre2014 | 2014]
```

With the *pre2014* settings, you will get STO fitting, the Voronoi Grid, the old, symmetric *k*-point sampling, and no confinement, resembling the behavior of the 2013 version.

5.1 Basis set

With the key `BasisDefaults` you can control the quality of the basis set, and the level of frozen core approximation. Given your preferences the basis and fit sets are retrieved from the database. The output starts with a copy of your input, but with the basis and fit sets inserted from the database: at this point you can exactly see which values are used. You can always override the defaults by specifying the *AtomType* (page 34) keyword. There is no need for special basis sets for relativistic calculations.

5.1.1 Automatic mode: BasisDefaults

The simplest way of specifying a basis set is via the `BasisDefaults` block key. For example:

```
BasisDefaults
   BasisType TZP
   Core None
End
```

instructs BAND to use an all-electron (`Core None`) TZP basis set.

BasisDefaults (block-type) This key allows you to generate automatically the required AtomType blocks from the database. The basis sets database is located in "$ADFHOME/atomicdata/band". You can always see what has been generated at the beginning of the output file, where your input is echoed with the AtomType blocks expanded. It shows as a comment from which file each AtomType has been copied.

```
! --- basis from file /path/to/TZ2P/C.1s---

AtomType C
...
End

! --- basis from file /path/to/DZ/H---

AtomType H
...
End
```

The `BasisDefaults` block has the following subkeys.

BasisType The following basis sets are available.

- **DZ**: Double zeta. Adapted from the ADF basis set of the same name. The smallest basis set. The results are most likely not so accurate compared to larger basis sets.

- **DZP**: Double zeta plus polarization function. Is only available for *main group elements* up to Krypton. **Note:** For other elements a TZP basis set will be used **automatically**!

- **TZP**: Triple zeta plus polarization. Adapted from the ADF basis set of the same name.

- **TZ2P**: Triple zeta plus double polarization. Adapted from the ADF basis set of the same name. This is a good basis set.

- **QZ4P**: Quadruple zeta plus quadrupole polarization. The biggest basis set. For benchmark purposes only.

If you have a large unit cell, you could start with the DZ basis, to see if it will be feasible at all. For more reliable results, use a triple zeta basis set.

You can override the defaults per type as presented in the following example (compare to *BasisDefaults* (page 121) example):

```
Atoms
   C               0.0   0.0   0.0
   O.large_basis   0.0   2.13  0.0
   H.large_basis   4.0   0.0   0.0
   H               4.0   1.43  0.0
End

BasisDefaults
   ! Cheap defaults
   BasisType DZ
   Core Large
   C               Core=None        ! This C has no frozen core
   O.large_basis   BasisType=TZ2P   ! This O with a larger basis
   H.large_basis   BasisType=V      ! This one also with a larger basis
End
```

Furthermore, you can override the defaults per `Atoms` block by specifying `Atoms%Core` and `Atoms%BasisType`, or `Atoms%Path` (sub)keys. Following is an example:

```
Atoms :: block one
   BasisType TZ2P
   Core      Small
   Si   0.0  0.0  0.0
   ...
End

Atoms :: block two
   BasisType DZP
   Core      None
   H    1.4  1.4  1.4
   ...
End
```

Core (**Default: Large**) This influences the size of the frozen core used. Possible values are *None*, *Small*, *Medium*, *Large*. You should be aware that a change of this parameter does not necessarily mean that a different frozen core is used. You can again check in the output what file was really used. You can see it from the extension in the comment line preceding the `AtomType` key

```
! --- basis from file /path/to/TZ2P/C.1s---
```

or from the Dirac block in the `AtomType` key.

Note: For all elements up to Au the *Medium* and *Large* frozen core are identical. And starting with Thallium there are three different frozen core definitions available.

FitType The STO fit set is by default determined via the `BasisType`. With this key one can choose the fit set independently from the BasisType key. (See *obsolete density fitting* (page 41))

5.1.2 Available standard basis sets

The next table gives an idication which all electron (ae) and frozen core (fc) standard basis sets are available for the different elements in BAND.

Table 5.1: Available standard basis sets for non-relativistic and ZORA calculations H-Uuo (Z=1-118)

Element	ae	fc	SZ, DZ	DZP	TZP, TZ2P, QZ4P
H-He (Z=1-2)	ae		Yes	Yes	Yes
Li-Ne (Z=3-10)	ae	.1s	Yes	Yes	Yes
Na-Mg (Z=11-12)	ae	.1s .2p	Yes	Yes	Yes
Al-Ar (Z=13-18)	ae	.2p	Yes	Yes	Yes
K-Ca (Z=19-20)	ae	.2p .3p	Yes	Yes	Yes
Sc-Zn (Z=21-30)	ae	.2p .3p	Yes		Yes
Ga-Kr (Z=31-36)	ae	.3p .3d	Yes	Yes	Yes
Rb-Sr (Z=37-38)	ae	.3p .3d .4p	Yes		Yes
Y-Cd (Z=39-48)	ae	.3d .4p	Yes		Yes
In-Ba (Z=49-56)	ae	.4p .4d	Yes		Yes
La-Lu (Z=57-71)	ae	.4d .5p	Yes		Yes
Hf-Hg (Z=72-80)	ae	.4d .4f	Yes		Yes
Tl (Z=81)	ae	.4d .4f .5p	Yes		Yes
Pb-Rn (Z=82-86)	ae	.4d .4f .5p .5d	Yes		Yes
Fr-Ra (Z=87-88)	ae	.5p .5d	Yes		Yes
Ac-Lr (Z=89-103)	ae	.5d .6p	Yes		Yes
Rf-Uuo(Z=104-118)	ae	.5d .5f	Yes		Yes

- element name (without suffix): all electron (ae)

- .1s frozen: 1s

- .2p frozen: 1s 2s 2p

- .3p frozen: 1s 2s 2p 3s 3p

- .3d frozen: 1s 2s 2p 3s 3p 3d

- .4p frozen: 1s 2s 2p 3s 3p 3d 4s 4p

- .4d frozen: 1s 2s 2p 3s 3p 3d 4s 4p 4d

- .4f frozen: 1s 2s 2p 3s 3p 3d 4s 4p 4d 4f

- .5p frozen: 1s 2s 2p 3s 3p 3d 4s 4p 4d 5s 5p (La-Lu)

- .5p frozen: 1s 2s 2p 3s 3p 3d 4s 4p 4d 4f 5s 5p (other)

- .5d frozen: 1s 2s 2p 3s 3p 3d 4s 4p 4d 4f 5s 5p 5d

- .6p frozen: 1s 2s 2p 3s 3p 3d 4s 4p 4d 4f 5s 5p 5d 6s 6p (Ac-Lr)

- .5f frozen: 1s 2s 2p 3s 3p 3d 4s 4p 4d 4f 5s 5p 5d 5f 6s 6p

5.1.3 Manually specifying AtomTypes

AtomType (block-type) (*Expert Option*) Description of the atom type. Contains the block keys `Dirac`, `BasisFunctions` and `FitFunctions`. The key corresponds to one atom type. The ordering of the `AtomType` keys (in case of more than one atom type) is **NOT** arbitrary. It is interpreted as corresponding to the ordering of the `Atoms` keys. The n-th `AtomType` key supplies information for the numerical atom of the nth type, which in turn has atoms at positions defined by the nth `Atoms` key.

```
AtomType ElementSymbol
   Dirac ChemSym
      {option}
```

```
        ...
        shells cores
        shell_specification {occupation_number}
        ...
    SubEnd
    {BasisFunctions
        shell_specification STO_exponent
        ...
    SubEnd}
    FitFunctions
        shell_specification STO_exponent
        ...
    SubEnd
END
```

The argument *ElementSymbol* to `AtomType` is the symbol of the element that is referred to in the `Atoms` key block.

Dirac (block-type) Specification of the numerical ('Herman-Skillman') free atom, which defines the initial guess for the SCF density, and which also (optionally) supplies Numerical Atomic Orbitals (NOs) as basis functions, and/or as STO fit functions for the crystal calculation. The argument *ChemSym* of this option is the symbol of the element of the atom type. The data records of the `Dirac` key are:

1. the number of atomic shells (1s,2s,2p,etc.) and the nr. of core-shells (two integers on one line).

2. specification of the shell and its electronic occupation.

This specification can be done via quantum numbers or using the standard designation (e.g. '1 0' is equivalent to '1s'). Optionally one may insert anywhere in the Dirac block a record *Valence*, which signifies that all numerical valence orbitals will be used as basis functions (NOs) in the crystal calculation. You can also insert *NumericalFit* followed by a number (max. l-value) in the key block, which causes the program to use numerical STO fit functions. For example NumericalFit 2 means that the squares of all s,p, and d NOs will be used as STO fit functions with $l = 0$, since the NOs are spherically symmetric. If you insert *Spinor*, a spin-orbit relativistic calculation for the single-atom will be carried out.

The Herman-Skillman program generates all its functions (atomic potential, charge density, one-electron states) as tables of values in a logarithmic radial grid. The number of points in the grid, and the min. and max. r-value are defaulted at 3000, 0.000001, and 100.0 (a.u.) respectively. These defaults can be overwritten by specifying anywhere in the Dirac block the (sub)keys *radial*, *rmin* and *rmax*.

The program will do a spin-unrestricted calculation for the atoms in addition to the restricted one. The occupation of the spin-orbitals will be of maximum spin-multiplicity and cannot be controlled in the Dirac key-block.

BasisFunctions (block-type) Slater-type orbitals, specified by quantum numbers n,:math:*l* or by the letter designation (e.g. 2p) and one real (alpha) per STO. One STO per record. Use of this key is optional in the sense that Slater-type functions are not needed if other basis functions have been specified (i.e. the numerical atomic orbitals, see key Dirac).

FitFunctions (block-type) Slater-type fit functions, described in the same way as in `BasisFunctions`. Each `FitFunctions` key corresponds to one atom type, the type being the one of the preceding `Dirac` key. The selection choice of a 'good' fit set is a matter of experience. Fair quality sets are included in the database of the molecular program ADF.

Example:

```
AtomType C :: Carbon atom
    Dirac C
       3 1
```

```
    VALENCE
        1s
        2s
        2p 2.0
    SubEnd
    BasisFunctions
        1s 1.7
        ...
    SubEnd
    FitFunctions
        1s 13.5
        2s 11.0
        ...
    SubEndEnd
```

TestFunctions (block-type) An optional subkey of the `AtomType` key block is `TestFunctions` which has the same format as the `BasisFunctions` and `FitFunctions` blocks. The `TestFunctions` block specifies STOs to be used as test functions in the numerical integration package. For the time being the l value is ignored. A possible application is to include a very tight function, to increase the accuracy near a nucleus.

5.1.4 Confinement of basis functions

It is possible to alter the radial part of the basis functions in order to make them more compact.

SOFTCONFINEMENT (block-type) With soft confinement the radial part of the basis functions is multiplied with a Fermi-Dirac (FD) function. A FD function goes from one to zero, controlled by two parameters. It has a value 0.5 at `Radius`, and the decay width is `Delta`.

```
SoftConfinement
   Quality [Basic|Normal|Good|VeryGood|Excellent]
   Radius r
   Delta  d
End
```

This will set the confinement for all atoms. The most convenient way is to specify the `Quality` subkey. The resulting confinement parameters are

Quality	Radius	Delta
Basic	7.0	0.7
Normal	10.0	1.0
Good	20.0	2.0
VeryGood	/	/

Specifying the `Radius` or `Delta` subkey will override the defaults. A negative value means no confinement.

CONFINEMENT (block-type): subkey of atomtype You can also control the soft confinement at the atom type level. In a slab calculation this allows one to use different settings for surface atoms than for those in inner layers. You can specify the range and the decay speed of the FD function like:

```
Confinement
   Radius 7
   Delta 0.7
SubEnd
```

If the decay `Delta` is not specified, it defaults to 0.1*'Radius'. Relativistic effects are treated correctly. If the confinement option is used in combination with the `TAILS` option, the calculation may run significantly faster.

The `Confinement` option can also be useful without the tails option in cases where near linear dependency reduces the numerical reliability of the results (cf. `Dependency` keyword). This typically occurs for highly coordinated systems with large basis sets containing diffuse functions. For such cases, the confinement option (possibly in combination with the `Dependency` keyword) reduces the numerical problems.

NB: does not work with the `BasisDefaults` key (because it is a subkey of `AtomType`).

5.2 Basis Set Superposition Error (BSSE)

The Ghost Atom feature enables the calculation of Basis Set Superposition Errors (BSSE). Normally, if you want to know the bonding energy of system A with system B you calculate three energies

1. $E(A + B)$

2. $E(A)$

3. $E(B)$

The bond energy is then $E(A + B) - E(A) - E(B)$

The BSSE correction is about the idea that we can also calculate E(A) including basis functions from molecule B.

You can make a ghost atom by simply adding "Gh." in front of the element name, for instance "Gh.H" for a ghost hydrogen , "Gh.C" for a ghost Carbon atom.

You will get a better bonding energy, closer to the basis set limit by calculating

$E(A + B) - E(A/withBasghost) - E(B/withAasghost)$

The BSSE correction is

$E(A) - E(A/withBasghost) + E(B) - E(B/withAasghost)$

See also: *Example: BSSE correction* (page 125).

5.3 Numerical Integration

Many of the integrals are obtained by numerical integration. Two grids are available: the Becke grid (**default**) and the old Voronoi scheme (deprecated). One can switch between the two using:

```
IntegrationMethod [Becke | Voronoi]
```

5.3.1 Becke Grid

This numerical integration grid is a refined version of the fuzzy cells integration scheme developed by Becke.[*51* (page 208)] The implementation in BAND is described in Ref. [*52* (page 208)].

The quality of the Becke integration grid can be changed within the `BECKEGRID` block key

```
BECKEGRID
   Quality [basic|normal|good|verygood|excellent]
End
```

Quality (**Default: Normal**) For a description of the various "qualities" and the associated numerical accuracy see reference *52* (page 208). The integration grid quality defined in the `BECKEGRID` block key overrules the *NumericalQuality* (page 31).

Advanced options:

```
BECKEGRID
  {AtomDepQuality
     Ia1 [basic|normal|good|verygood|excellent]
     Ia2 [basic|normal|good|verygood|excellent]
     ...
  SubEnd}
  {RadialGridBoost boost}
End
```

AtomDepQuality One can define a different grid quality for each atom, with input numbers *Ia1*, *Ia2*, etc. If an atom is not present in the `AtomDepQuality` section, the quality defined in the `Quality` key will be used. *Example: Multiresolution* (page 123) illustrates how to use this option.

RadialGridBoost (**Default: 1.0** (or 3.0 if a numerically sensitive functional is used)) The number of radial integration points will be boosted by this factor. Some XC functionals require very accurate radial integration grids, so BAND will automatically *boost* the radial grid for the following numerically sensitive functionals: `LibXC M05`, `LibXC M05-2X`, `LibXC M06-2X`, `LibXC M06-HF`, `LibXC M06-L`, `LibXC M08-HX`, `LibXC M08-SO`, `LibXC M11-L`, `LibXC MS0`, `LibXC MS1`, `LibXC MS2`, `LibXC MS2H`, `LibXC MVS`, `LibXC MVSH`, `LibXC N12`, `LibXC N12-SX`, `LibXC SOGGA11`, `LibXC SOGGA11-X`, `LibXC TH1`, `LibXC TH2`, `LibXC WB97`, `LibXC WB97X`, `MetaGGA M06L`, `MetaHybrid M06-2X`, `MetaHybrid M06-HF`, `MetaGGA MVS`

Notes:

- The space-partition function used in BAND differs from the one described in Ref. [*52* (page 208)]. The unnormalized partition function used in the program is defined as (Ω_I is an element-dependent parameter: 0.1 Bohr for H, 0.3 Bohr for He-Xe and 0.6 Bohr for Cs-Uuo):

$$\mathcal{P}_{i,U} = \begin{cases} 1 & \text{if } r_{i,U} < \Omega_I \\ 0 & \text{if } \exists j : r_{j,U} < \Omega_J \\ \eta_i \frac{e^{-2(r_{i,U}-\Omega_I)/a_0}}{(r_{i,U}-\Omega_I)^2} & \text{elsewhere} \end{cases}$$

- A Becke grid of normal quality is roughly equivalent (in both absolute accuracy and computation time) to INTEGRATION 4 (Voronoi scheme), and a Becke grid of good quality is roughly equivalent to INTEGRATION 6 (Voronoi scheme).

- The Becke grid is not very well suited to calculate Voronoi deformation density (VDD) charges. For accurate calculation of VDD charges the Voronoi integration scheme is recommended.

5.3.2 Radial grid

With this keyword the radial grid part of the basis function evaluation with the subprogram Dirac can be controlled.

```
RadialDefaults
  NR nr
  RMin rmin
  RMax rmax
End
```

NR (**Default: 3000**) This key handles the number of radial points. With very high values like 30.000 the Dirac subprogram may not converge.

RMin, RMax These keys define the lower (**Default: 1e-6**) and upper (**Default: 100**) bound of the logarithmic grid.

5.3.3 Elliptic integrals

(*Expert Option*) The integration of the electrostatic interaction between spherical atoms is done very precisely in an elliptic grid, which depends on NUELSTAT and NVELSTAT

NUELSTAT (**Default: 50**) Electrostatic interaction integrals between spherical atomic densities are computed by numerical integration over an elliptic grid. *Nuelstat* is the outward (parabolic) coordinate number of integration points.

NVELSTAT (**Default: 80**) Electrostatic interaction integrals between spherical atomic densities are computed by numerical integration over an elliptic grid. *Nvelstat* is the angular (elliptic) coordinate number of integration points.

5.3.4 Voronoi grid (deprecated)

INTEGRATION (block-type) Parameter-specifications for the generation of numerical integration points and weights. Most data records must be of the form 'parameter value'. The most important parameter is accint, which is defaulted to the value of key ACCURACY. Unless one is very familiar with the details of the numerical integration package, we strongly recommend not to use the INTEGRATION key, and to specify only ACCURACY. More information can be found in the literature.

ACCURACY This key expects a real value, normally between 3 and 7, and will automatically set all necessary options for the numerical integration with the Voronoi grid. A value of 3 would be basic quality and a value of 7 would be good quality

5.4 Reciprocal Space Numerical Integration (KSpace)

The k-space integration can be controlled via the KSpace block key. Two different k-space integration methods are available: the *Regular Grid* (**default**) and the *Tetrahedron Method*.

5.4.1 Regular K-Space grid

By default BAND uses a regular grid to sample the Brillouin zone (BZ).

Automatic mode

The simplest way to adjust the quality of the k-space integration is via the Quality key:

```
KSpace
  Quality [Basic|Normal|Good|VeryGood|Excellent]
End
```

Quality (**Default: Normal**) With the automatic grid, the program will look at the size of a lattice vector. A big vector in real space, means a small one in reciprocal space. Thinking in real space lattice vectors the following intervals will be distinguished: 0-5 Bohr, 5-10 Bohr, 10-20 Bohr, 20-50 Bohr, and beyond. Here is the table explaining how many points will be used along a lattice vector.

range	Basic	Normal	Good	VeryGood	Excellent
0-5	5	9	13	17	21
5-10	3	5	9	13	17
10-20	1	3	5	9	13
20-50	1	1	3	5	9
50-...	1	1	1	3	5

By preferring odd-numbered values we can use a quadratic interpolation method, and have the Γ point in the grid. It is then reasonable to assume a decaying error when going to a better quality setting.

User-defined regular grid

It is possible to manually define the number of k-space points along each reciprocal lattice vector:

```
KSpace
  Grid n1 {n2} {n3}
End
```

For 1D periodic systems you should specify only `n1`, for 2D systems `n1` and `n2`, and for 3D systems `n1`, `n2` and `n3`.

5.4.2 Tetrahedron Method

The tetrahedron method can be useful when especially high symmetry points in the BZ are needed to capture the correct physics of the system, graphene being a notable example.

```
KSpace integer &
   Grid Symmetric
End
```

The parameter for numerical integration over the BZ is an integer value.

- 1: absolutely minimal (only the Γ point is used)

- 2/4/...: linear tetrahedron method

- 3/5/...: quadratic tetrahedron method

The linear tetrahedron method is usually inferior to the quadratic tetrahedron method. Try 3 for a reasonable result, or 5 for higher precision.

General Remark: The tetrahedron method samples the irreducible wedge of the first BZ, whereas the regular grid samples the whole, first BZ. As a rule of thumb you need to choose roughly twice the value for the regular grid. For example kspace 2 compares to grid 4 4 4, kspace 3 to grid 5 5 5, etc.. Sticking to this rule the number of unique k-points will be roughly similar.

5.5 Density fitting (Zlm Fit)

The default density fitting scheme in BAND is the so-called **Zlm Fit**, described in reference *53* (page 208). We do **not** recommend to use the STO Fit anymore.

5.5.1 Zlm Fit

The basic idea behind Zlm Fit can be described as follows: the total electron-density is split into atomic densities (in a similar way as the volume is partitioned for the Becke grid). These atomic densities are then approximated by a combination of radial spline functions and real spherical harmonics (Zlm).

```
ZlmFit (block-type)
```

```
ZlmFit
  Quality [Basic|Normal|Good|VeryGood|Excellent]
End
```

Quality (**Default: Normal**) controls the accuracy and quality of fitted properties.

Advanced options:

```
ZlmFit
  {AtomDepQuality
     Ia1 [basic|normal|good|verygood|excellent]
     Ia2 [basic|normal|good|verygood|excellent]
     ...
  SubEnd}
End
```

AtomDepQuality One can define a different fit quality for each atom, with input numbers *Ia1*, *Ia2*, etc. If an atom is not present in the `AtomDepQuality` section, the quality defined in the `Quality` key will be used. *Example: Multiresolution* (page 123) illustrates how to use this option.

5.5.2 Obsolete Method - STO Fit

In previous version of BAND this was the default option, which is now replaced by Zlm Fit. It is still used in the context of `NMR` and `OldResponse` calculations.

5.6 Self-consistency (SCF)

The SCF procedure searches for a self-consistent density. The self-consistent error is the square root of the integral of the squared difference between the input and output density of the cycle operator. When the SCF error is below a certain criterion, controlled by subkey `Criterion` of block key `Convergence`, convergence is reached. In case of bad convergence the SCF looks at the subkeys `Mixing`, and `Degenerate`, and the subkeys of block key `DIIS`.

See also:

Troubleshooting: *SCF does not converge* (page 102)

5.6.1 SCF key

The SCF key block controls technical SCF parameters

```
SCF
  {Mixing value}
  {Iterations n}
  {Eigenstates}
  {Pmatrix}
  {Rate value}
```

```
   {VSplit value}
End
```

Mixing (**Default: 0.075**) Initial 'damping' parameter in the SCF procedure, for the iterative update of the potential: new potential = old potential + mix (computed potential-old potential). Note: the program automatically adapts Mixing during the SCF iterations, in an attempt to find the optimal mixing value.

Iterations (**Default: 0**) The maximum number of SCF iterations to be performed. If zero, termination of the SCF procedure will depend on other aspects (convergence, time-out, insufficient progress towards convergence, ...).

EigenStates The program knows two alternative ways to evaluate the charge density iteratively in the SCF procedure: from the P-matrix, and directly from the squared occupied eigenstates. By default the program actually uses both at least one time and tries to take the most efficient. If present, `Eigenstates` turns off this comparison and lets the program stick to one method (from the eigenstates).

Pmatrix If present, evaluate the charge density from the P-matrix. See also the subkey Eigenstates.

Rate (**Default: 0.99**) Minimum rate of convergence for the SCF procedure. If progress is too slow the program will take measures (such as smearing out occupations around the Fermi level, see subkey `Degenerate` of block key `Convergence`) or, if everything seems to fail, it will stop.

VSplit (**Default: 5E-2**) To disturb degeneracy of alpha and beta spin MOs the value of this key is added to the beta spin potential at the startup.

5.6.2 Convergence key

All options and parameters related to the convergence behavior of the SCF procedure are defined in the `Convergence` block key. Also the finite temperature distribution is part of this

```
Convergence
  {Criterion value}
  {ElectronicTemperature value}
  {Degenerate value}
  {SpinFlip list}
  {StartWithMaxSpin [True|False]}
  {InitialDensity [RHO|PSI]}
  {Boltzmann n}
  {LessDegenerate [False|True]}
  {NoDegenerate}
End
```

Criterion Criterion for termination of the SCF procedure. The default depends on *NumericalQuality* (page 31). For Normal numerical quality it is 1E-6.

ElectronicTemperature (**Default: 0**) Requires a numerical argument, which is an energy width (in a.u.). It simulates a finite-temperature electronic distribution. The key may be used to achieve convergence in an otherwise problematically converging system. The energy of a finite-T distribution is different from the T=0 value, but for small T a fair approximation of the zero-T energy is obtained by extrapolation. The extrapolation energy correction term is printed with the survey of the bonding energy in the output file. Check that this value is not too large. Build experience yourself how different settings may affect the outcomes. **Note**: this key is meant to help you overcome convergence problems, not to do finite-temperature research! Only the electronic distribution is computed T-dependent, other aspects are not accounted for!

Degenerate (**Default: default**) Smooths (slightly) occupation numbers around the Fermi level, so as to insure that nearly-degenerate states get (nearly-) identical occupations. **Be aware**: In case of problematic SCF convergence the program will turn this key on automatically, unless the key `Nodegenerate` is set in input. The smoothing

depends on the argument to this key, which can be considered a 'degeneration width'. When the argument reads **default**, the program will use the value **1e-4 a.u.** for the energy width.

SpinFlip List here the atoms for which you want the initial spin polarization to be flipped. This way you can distinguish between ferromagnetic and anti ferromagnetic states. Currently, it is not allowed to give symmetry equivalent atoms a different spin orientation. To achieve that you have to break the symmetry.

StartWithMaxSpin (**Default: True**) To break the initial perfect symmetry of up and down densities there are two strategies. One is to occupy the numerical orbitals in a maximum spin configuration. The alternative is to add a constant to the potential. See also *Vsplit* (page 42) key.

InitialDensity (**Default: RHO**) The SCF is started with a guess of the density. There are the following choices **RHO**: the sum of atomic density. **PSI**: construct an initial eigensystem by occupying the atomic orbitals. The guessed eigensystem is orthonormalized, and from this the density is calculated.

LESSDEGENERATE If smoothing of occupations over nearly degenerate orbitals is applied (see *Degenerate* (page 42) key), then, if this key is set in the input file, the program will limit the smoothing energy range to 1e-4 a.u. as soon as the SCF has converged 'halfway', i.e. when the SCF error has decreased to the square root of its convergence criterion.

NODEGENERATE This key prevents any internal automatic setting of the key *DEGENERATE* (page 42).

5.6.3 DIIS key

The DIIS procedure to obtain the SCF solution depends on several parameters. Default values can be overruled with this key block.

```
DIIS
  {dimix rval}
  {adaptable lval}
  {ncycledamp ival}
  {nvctrx ival}
  {clarge rval}
  {chuge rval}
  {condition rval}
  {variant sval}
END
```

dimix (**Default: 0.2**) mixing parameter for the DIIS procedure.

adaptable (**Default: true**) change automatically the value of `dimix` during the SCF.

ncycledamp (**Default: 5**) number of initial iterations where damping is applied, before any DIIS is considered.

nvctrx (**Default: 20**) maximum number of DIIS expansion vectors.

clarge (**Default: 20**) when the largest DIIS coefficient exceeds this value, the oldest DIIS vector is removed and the procedure re-applied.

chuge (**Default: 50**) when the largest coefficient in the DIIS expansion exceeds this value, damping is applied.

condition (**Default: 1e+6**) the condition number of the DIIS matrix, the largest eigenvalue divided by the smallest, must not exceed this value. If this value is exceeded, this vector will be removed.

variant if specified, you can invoke one of the LIST methods, possible values are **LISTi**, **LISTb**, and **LISTd**. It appears that among these LISTi is the advised method. This can be tried if SCF convergence turns out to be problematic. As it requires an extra fitting step, it is more expensive per SCF iteration.

5.6.4 DIRIS key

DIRIS (block-type) Same options as `DIIS` key, except that this one applies to the DIIS procedure used in the Dirac subprogram, for numerical single atom calculations, which constructs the radial tables for the NAOs.

5.7 Hartree–Fock RI scheme

The Hartree-Fock exchange matrix is calculated through a procedure known as Resolution of the Identity (RI). The implementation of the RI scheme in BAND is loosely based on work by Ren *et al.* [*57* (page 208)]. For more information on hybrid functionals in BAND, see the *XC section* (page 17).

Technical aspects of the RI scheme can be tweaked in the `RIHartreeFock` key block:

```
RIHartreeFock
  Quality [basic|normal|good|verygood]
  FitSetType [Mirko|ET6|ET7|ET8]
  DependencyThreshold 1.0E-3
End
```

Quality (**Default: Normal**) Set the numerical quality of some internal technical procedures, including numerical integration and linear scaling parameters.

FitSetType (**Default: Mirko**) The auxiliary fit set employed in the RI scheme. ET8 is a fairly large fit set, and can be used for assessing the accuracy of the RI procedure.

DependencyThreshold (**Default: 1.0E-3**) To improve numerical stability, almost linearly-dependent combination of basis functions are removed from the Hartree-Fock exchange matrix. If you obtain unphysical large bond energy in an Hybrid calculation, you might try setting the `DependencyThreshold` to a larger value (*e.g.* 3.0E-3).

For efficiency and numerical stability reasons, it is advisable to include:

```
SoftConfinement
  Quality Basic
End
```

See the *Confinement of basis functions* (page 36) section for more info.

Notes: for periodic systems it is only possible to use short-range hybrid functionals (*e.g.* HSE06) in combination with all-electron basis sets.

5.8 Technical Settings

There are of course many other settings influencing the precision and performance. Usually the user does not need to care about them.

5.8.1 Linear Scaling

Tails Ignore function tails. By default no tails are ignored. Both CPU time and disk space can be saved by using the TAILS option. This option is most effective when combined with the `Confine` suboption.

```
TAILS {bas crbas} {core crcore} {fit crfit}
```

One real argument for keys `bas` and `core`, which should be a small value (**Default: crbas = 1e-6**). The core criterion defaults to the bas criterion (if set). The tail criterion specifies that tails of exponentially decaying (basis) functions are ignored, in the construction of Bloch functions, beyond the point where the remaining part of the function tail (radially) integrates to less than the criterion, relative to the integral of the function from zero to infinity. Also for the fit one can specify a cutoff, but this is turned off by default (**Default: crfit = 1e-15**). Here we advise to use a more strict criterion than for crbas (because the fit is usually much more dependent than the basis). This option has some refinements. For example

```
Tails confine=1e-2 bas=1e-5 Rosa
```

As you see there are two new elements (**confine** and **Rosa**). The first (**confine**) specifies that all basis functions are optimized for the tails option, by multiplying the tail of the function with a rapidly decaying function. This step affects the shape of all functions outside the radius where the relative norm of the function is smaller than 1e-2. The effects on the shape of the functions are typically very small. The second entry (**Rosa**) has the effect that the criterion becomes more strict for tight functions. For safer (but slower) calculations, specify smaller values for **confine** and **bas**, such as confine=1e-3 bas=1e-6.

The **Rosa** key works as follows. Determine the radius R where according to the normal criterion the function is negligible. Next substitute R -> R + 3*exp(-R/3). This means that for tight functions the tails radius is replaced by 3, and for diffuse functions this modifier has no effect.

Tentatively, based on the work by *Rosa Bulo*, we suggest to use

```
Tails bas=1e-3 Rosa
```

The `TAILS` keyword works most effectively when combined with the confinement keyword.

Note: The confine key described here should not be confused with the confinement option for each atom type. The confine option here affects functions in the region where they become very small, independent of the distance to the nucleus. The confinement option introduces a soft cut-off for all functions of a particular atom type, at a specific distance from the nucleus.

5.8.2 Dependency

DEPENDENCY Criterion for dependency of the basis and fit set:

```
DEPENDENCY {basis tolbas} {core tolcor} {fit tolfit} {corevalence tolovl}
```

basis (**Default: 1e-8**) Smallest eigenvalue of the overlap matrix of normalized Bloch functions. See also the discussion in *Recommendations* (page 99) about basis set dependency.

core (**Default: 0.98**) The program verifies that the frozen core approximation is reasonable, by checking the smallest value of the overlap matrix of the core (Bloch) orbitals against this criterion.

fit (**Default: 1e-6**) Criterion for dependency of the total set of fit functions. The value monitored is the smallest eigenvalue of the overlap matrix of normalized Bloch sums of symmetrized fit functions.

corevalence (**Default: 1e-5**) Criterion for dependency of the core functions on the valence basis. The maximum overlap between any two normalized functions in the two respective function spaces should not exceed 1.0-coreval(...).

5.8.3 Screening

The program BAND performs many lattice summations which are in practice truncated. The two prime examples are the construction of the Bloch basis and of the fit basis (*obsolete option*) and the construction of the Coulomb potential of the STO fit functions. The precision of the lattice summations is controlled by the `SCREENING` key

SCREENING (block) Parameters that influence the screening and tails of basis functions. Recognized options are

> **CUTOFF** Criterion for negligibility of tails in the construction of Bloch sums. Default depends on Accuracy.

> **DMADEL** One of the parameters that define the screening of Coulomb-potentials in lattice sums. Depends by default on Accuracy, rmadel, and rcelx. One should consult the literature for more information.

> **RCELX** Max. distance of lattice site from which tails of atomic functions will be taken into account for the Bloch sums. Default depends on Accuracy.

> **RMADEL** One of the parameters that define screening of the Coulomb potentials in lattice summations. Depends by default on Accuracy, dmadel, rcelx. One should consult the literature for more information.

> **NODIRECTIONALSCREENING** Real space lattice sums of slowly (or non-) convergent terms, such as the Coulomb potential, are computed by a screening technique. In previous releases, the screening was applied to all (long-range) Coulomb expressions. Starting from BAND98 screening is only applied in the periodicity directions. This key restores the original situation: screening in all directions.

5.8.4 Direct (on the fly) calculation of basis and fit

BAND usually calculates basis functions and theirs derivatives on the fly. However, for small bulk systems it can be faster to write the information to disk. Then one can set the `DirectBas` key to false. (**Default = true**)

```
Programmer
   {DirectBas [true | false]}
End
```

5.8.5 Fermi energy search

Fermi (block type) This key sets technical parameter used in the search for the Fermi energy, which is carried out at each cycle of the SCF procedure.

> **MaxTry** (**Default: 50**) Maximum number of attempts to locate the Fermi energy. The procedure is iterative in nature, narrowing the energy band in which the Fermi energy must lie, between an upper and a lower bound. If the procedure has not converged sufficiently within `MaxTry` iterations, the program takes a reasonable value and constructs the charge density by interpolation between the functions corresponding to the last used upper and lower bounds for the Fermi energy.

> **Delta** (**Default: 1e-4**) Convergence criterion: upper and lower bounds for the Fermi energy and the corresponding integrated charge volumes must be equal within delta.

> **Eps** (**Default: 1e-10**) After convergence of the Fermi energy search procedure, a final estimate is defined by interpolation and the corresponding integrated charge volume is tested. It should be exact, to machine precision. Tested is that it deviates not more than eps.

5.8.6 Block size

Efficiency and memory usage depend on how large the vector size of the program is

CPVECTOR The code is vectorized and this key can be used to set the vector length. Default depends on the machine and should be set at the installation of the program.

KGRPX is an absolute upper bound on the number of k-points processed together. Specifying this key and CPVector you can override bands defaults.

STRUCTURE AND REACTIVITY

To study structure and reactivity common tasks are **geometry optimization**, **transition state search** and a **frequency run**. A frequency run can be done to obtain a good initial Hessian for a geometry optimization, or to check whether a proper minimum or transition state has been found.

The basis for locating stationary points on the PES are the nuclear gradients. (The user may assume that the program does that automatically when needed.)

6.1 Nuclear energy gradients

If you want the program to calculate the gradient of the energy w. r. t. nuclear displacements you should add the `Gradients` keyword. If the `SelectedAtoms` key is present, only the gradient of the selected atoms will be evaluated, and the rest will be set to zero.

Details about the theory of the analytical gradients within the BAND program can be found in the work of *Kadantsev* et al. [*19* (page 206)].

Gradients (block-type) Key to control the geometry optimization:

```
Gradients
  {LatticeGradients [false | true]}
  {LatticeStepSize step}
  {UseRelativeLatticeStep  [true | false]}
End
```

> **LatticeGradients** (**Default: false**) This key handles whether the lattice gradients are evaluated numerically (false) or analytically (true).
>
> **latticeStepSize** (**Default: 0.012**) With this keyword the step size for the numerical differentiation can be changed. When `UseRelativeLatticeStep` is set to true the step will be relative to the size of the displacement. (Unit is Bohr)

The user rarely needs to specify the `Gradients` keyblock as it is automatically added for geometry optimizations and frequency runs.

This option honors the `SelectedAtoms` key, in which case only the forces will be calculated for the selected atoms.

6.2 Lattice gradients

The step size for numerical differentiation is controlled with *Gradients%latticeStepSize* (page 47), and *Gradients%UseRelativeLatticeStep* (page 47).

LatticeGradients (block-type) Key to handle special options for the calculation of analytical and numerical lattice gradients.

```
LatticeGradients
   {Analytical [true | false]}
   {AnalyticalPulay [true | false]}
   {AnalyticalKinetic [true | false]}
   {AnalyticalXC [true | false]}
   {AnalyticalElectrostatic [true | false]}
End
```

Analytical (**Default: false**) If set to true, the following options `AnalyticalElectrostatic`, `AnalyticalKinetic`, `AnalyticalPulay`, and `AnalyticalXC` will be set to true automatically.

AnalyticalElectrostatic (**Default: true**) If true the electrostatic energy contribution to the lattice gradients is calculated analytically, else it is calculated numerically.

AnalyticalKinetic (**Default: true**) If true the kinetic energy contribution to the lattice gradients is calculated analytically, else it is calculated numerically.

AnalyticalPulay (**Default: true**) If true the Pulay contribution to the lattice gradients is calculated analytically, else it is calculated numerically.

AnalyticalXC (**Default: true**) If true the exchange-correlation energy contribution to the lattice gradients is calculated analytically, else it is calculated numerically.

In case you want to calculate forces acting on lattice vectors without performing an geometry optimization, you should use `LatticeGradients%Analytical true`!

6.3 Geometry optimization (GeoOpt)

The geometry can be optimized by adding the `GeoOpt` keyblock. Within the block you can specify the maximum number of cycles and the convergence criterion. The optimizer works with Cartesian coordinates and a Quasi Newton stepper. The initial Hessian is by default the unit matrix, but can also be loaded from a previous geometry optimization or frequency run, also from faster methods as DFTB and UFF. Various constraints are possible such as bond distances and angles. The GDIIS convergence accelerator is available but is disabled by default. A geometry optimization can be restarted from a previous result file. The geometry can be optimized non-relativistically, with scalar relativistic ZORA, or with spin-orbit coupling.

See also:

Troubleshooting: *Geometry does not converge* (page 103)

GeoOpt (block-type) Key to control the automatic geometry optimization:

```
GeoOpt
   {OptimizeLattice [false | true]}
   {Iterations n}
   {Converge {Grad=tolgrad} {E=tolE} {Step=tolStep}}
   {TrustRadius radius}
   {UseVariableTrustRadius var}
   {InitialHessian [UnitMatrix | filename | Swart]}
   {RestartSCF [true | false]}
   {RestartFit [true | false]}
   {GDIIS [true | false]}
   {StaticCosmoSurface [true | false]}
   {TSMode mode}
End
```

OptimizeLattice (**Default: false**) Whether or not to optimize the lattice vectors. The lattice gradients are obtained by numerical or analytical differentiation.

Iterations (**Default: 50**) maximum number of cycles.

Converge {Grad=tolgrad} {E=tolE} {Step=tolStep} various criteria to determine the convergence can be specified on line. Here *tolGrad* is the maximum gradient allowed (**Default: 1.0e-2** Hartree per Angstrom), *tolE* is the maximum energy change allowed (**Default: 1.0e-3** Hartree), *tolStep* is the maximum step size (**Default: 3e-2** Bohr)

TrustRadius (**Default: 0.2** Bohr) The step sizes taken by the optimizer will be limited to this value. If the proposed step is larger, it will be scaled back.

UseVariableTrustRadius (**Default: false**) Automatic adjustment of the trust radius based on the observed energy changes.

InitialHessian (**Default: UnitMatrix**) The initial Hessian is read from a "RUNKF" file *filename* of a previous frequency calculation. Band can also read the Hessian from DFTB or UFF calculations (either a frequency run or a geometry optimization). Another option, *Swart*, is to use the empirical force field derived Swart Hessian.

RestartSCF (**Default: true**) During the optimization try to restart the SCF from the previous geometry step.

RestartFit (**Default: false**) During the optimization try to restart the SCF from fit coefficients of the previous geometry step. Can not be used simultaneously with RestartSCF.

GDIIS (**Default: false**) Use the GDIIS convergence accelerator.

StaticCosmoSurface (**Default: false**) Keep the Cosmo surface (if any) constant.

TSMode Hessian normal mode index to follow during transition state search. The specified mode is selected on the first iteration and is followed regardless of its index on subsequent iterations. Modes with zero frequencies are not counted.

Note that a (failed) geometry optimization can be continued with the Restart key by specifying as option Geometry-Optimization.

6.4 Convergence

A geometry optimization is considered *converged* when all the following criteria are met (this information is also printed to the logfile):

1. the difference between the bond energy at the current geometry and at the previous geometry step is smaller than TolE

2. the maximum Cartesian nuclear gradient is smaller than TolG

3. the root mean square (RMS) of the Cartesian nuclear gradients is smaller than $\frac{2}{3}$TolG

4. the maximum Cartesian step is smaller than TolR

5. the root mean square (RMS) of the Cartesian steps is smaller than $\frac{2}{3}$TolR

Note: if the maximum and RMS gradients are 10 times smaller than the convergence criterion, then criteria 4 and 5 are ignored

The convergence criteria can be changed via the `Converge` key:

```
GEOMETRY
   Converge {E=TolE} {Grad=TolG} {Rad=TolR} {Angle=TolA}
END
```

`TolE` The criterion for changes in the energy, in Hartree. **Default: 1e-3**.

`TolG` The criterion for the nuclear gradients, in Hartree/Angstrom. **Default: 1e-3**.

`TolR` Refers to changes in the Cartesian coordinates or bond lengths, depending on in what coordinates you optimize, in Angstrom. **Default: 1e-2**.

`TolA` Refers to changes in bond- and dihedral angles, in degrees. This is only meaningful if optimization takes place in Z-matrix coordinates. **Default: 0.5 degree**.

If only a numerical value is supplied as argument for `Converge`, rather than a specification by name, it is considered to apply to the gradients (only):

```
GEOMETRY
   Converge 1E-4
End
```

is equivalent to:

```
GEOMETRY
   Converge grad=1E-4
End
```

6.5 Numerical frequencies (Hessian)

By specifying the `RunType` *Frequencies* you can calculate the Hessian which in turn yields the harmonic frequencies and vibrational modes ($q \rightarrow 0$ *limit* of a *phonons calculation* (page 53)). The Hessian is an important product that can be used in a geometry optimization, and in particular to start a transition state search. The second derivative of the energy (i.e. the Hessian) is obtained by numerically differentiating the analytical gradients. When your unit cell has N atoms in the unit cell and no symmetry (other than translational symmetry) the number of displacements is 2x3xN. For large systems this can be very time consuming, and you could consider to calculate only a partial Hessian. If you have a very symmetric unit cell, but are interested in asymmetric modes, set the subkeys `useA1Displacements` and `doA1Projection` to false. **Note**: the lattice vectors are assumed to be fixed.

A frequency run is invoked with

```
RunType
  Frequencies
End
```

The rest is controlled by the `Frequencies` key.

`Frequencies (block-type)` Key to control the numerical frequency run:

```
Frequencies
  {Step size}
  {useA1Displacements [true | false]}
  {doA1Projection [true | false]}
  {StaticCosmoSurface [true | false]}
End
```

`Step` (**Default: 0.01**) The step size (in Å) for the numerical differentiation.

`useA1Displacements` (**Default: true**) Determine only the total symmetric modes. When disabled, NOSYM calculations will be performed.

`doA1Projection` (**Default: true**) Symmetrize the Hessian. For most users the only sensible setting is to make it equal to the value of useA1Displacements.

StaticCosmoSurface (**Default: true**) Keep Cosmo surface (if any) constant. Although this is an approximation it is numerically much more stable. If you set it to *false* there is the risk that the number of Cosmo points is not constant for all finite displacements.

This option honors the `SelectedAtoms` key, in which case only the Hessian will be calculated for the selected atoms.

6.6 Transition state search

A transition state search is technically a geometry optimization. It is a search in the 3xN space for a point with vanishing gradients. The only difference is that it will try to go uphill in the direction of the lowest vibrational mode. The normal procedure is to guess a point that is as close as possible to the transition state. In that point you do a frequency run, which should hopefully produce a lowest vibrational mode in the direction of the transition state. You then start the Transition state search from that point, using the Hessian of the frequency run. At the end you can do again a frequency run to check that there is indeed only one single negative vibrational mode.

The transition state search is invoked like this

```
RunType
    TS
End
```

The rest is controlled by the `GeoOpt` key as in a normal geometry optimization.

6.7 Partial Hessian and (pre-)optimizations

If you consider the interaction of a molecule with a surface it may be initially convenient to freeze some of the lower layers in the slab, making calculations much cheaper. This can be done for the evaluation of the Hessian (see *SelectedAtoms* (page 13)) and during a geometry optimization or transition state search (see *Constraints* (page 51)).

> **Caution:** When using a partial Hessian to start a geometry optimization note that the atoms that you select during the frequency run, are the ones that should not be frozen in the geometry optimization. In a way the selection needs to be inverted.

6.8 Constrained optimization

During a geometry optimization certain constraints can be enforced with the `Constraints` key. The constraints refer to the coordinates of the unit cell as specified on input.

Constraints (block-type) Key to control constraints during a geometry optimization:

```
Constraints
  {Atom aI {x y z}
  {Coord aI coordI {coordIvalue}
  {Dist aI aJ distance}
  {Angle aI aJ aK angle}
  {Dihed aI aJ aK aL angle}
End
```

Atom Freeze the position of atom *a1* to the current coordinate, or if specified to (*x y z*). This constraint will make the calculation cheaper, because the gradients of frozen atoms need not be calculated.

Dist Constrain the distance between atoms *a1* and *a2* to *distance*.

Angle Constrain the angle between atoms *a1*, *a2*, and *a3* to *angle*.

Dihed Constrain the dihedral angle between atoms *a1*, *a2*, *a3*, and *a4* to *angle*.

6.9 Selected atoms

When doing a frequency or geometry optimization run you can restrict it to a limited number of selected nuclei. The selection may not break the symmetry

```
SelectedAtoms a1 a2 ... an
```

The list of atoms with indices 'a1 a2 ... an' are considered selected.

Note: Currently this option only has effect in a numerical frequency run and with the Gradients keyword.

6.10 Run Types (GeoOpt, TS, Frequencies and Phonons)

RunType (block-type) Key to invoke the following calculation types:

- *Geometry Optimization* (page 48)

```
RunType
  GeoOpt
End
```

- *Transition State Search* (page 51)

```
RunType
  TS
End
```

- *Frequency Calculation* (page 50)

```
RunType
  Frequencies
End
```

- *Phonon Calculation* (page 53)

```
RunType
  Phonons
End
```

PROPERTIES

7.1 Phonons and thermodynamics

Collective oscillations of atoms around theirs equilibrium positions, giving rise to lattice vibrations, are called *phonons*. BAND can calculate phonon dispersion curves within standard harmonic theory, implemented with a finite difference method in the spirit of the program PHON (http://www.homepages.ucl.ac.uk/~ucfbdxa/phon/) [*45* (page 207)]. Within the harmonic approximation we can calculate the partition function and therefore thermodynamic properties, such as the specific heat and the free energy.

Our implementation can also handle 1D and 2D periodic systems. For the study of isotopic shift of phonons see the *AtomProps key* (page 14).

To run a phonon calculation you need to do the following steps (see: *Phonons example* (page 174)):

- *Optimize the structure* (page 48), **including the lattice vectors**. The resulting geometry is the starting geometry for the phonon run.

- Specify a super cell transformation. In principle this should be as large as possible. In practice one may want to start with a 2x2x2 cell. Set the RunType to Phonons:

```
RunType
    Phonons
End
```

- Examine with the GUI the dispersion curves, and check for negative frequencies. If these exist then either:

    – the geometry was not close enough to the minimum. Try specifying a smaller gradient convergence criteria in your *geometry optimization* (page 48)

    – the super cell transformation was too small

    – the *numerical precision* (page 31) of the phonon run was not good enough.

PhononConfig (block-type) Key to control the phonon run:

```
PhononConfig
  {SuperCell (Sub block type)}
  {StepSize step}
  {nSides [1|2]}
  {useSymmetry [True|False]}
  {MinTempKelvin tmin}
  {MaxTempKelvin tmax}
  {NumTemperatures ntemp}
End
```

SuperCell This sub block key should hold the supercell transformation, which expresses the lattice vectors in terms of the vectors of the primitive cell.

StepSize (**Default: 0.076** Bohr) The step size taken to obtain all force constants.

nSides (**Default: 2**) By default a two-sided (or quadratic) numerical differentiation of the nuclear gradients is used. Using a single-sided (or linear) numerical differentiation is computationally faster but much less accurate (**Note**: in older versions of the program only the single-sided option was available).

useSymmetry (**Default: True**) Whether or not to exploit the symmetry of the system in the phonon calculation (**note:** the general *SYMMETRY* (page 95) key does not affect the symmetry of phonon calculations).

MinTempKelvin (**Default: 0.0** Kelvin) Minimum temperature for thermodynamic plots.

MaxTempKelvin (**Default: 1000.0** Kelvin) Maximum temperature for thermodynamic plots.

NumTemperatures (**Default: 1000**) Number of temperatures for thermodynamic plots.

Here is an example input:

```
PhononConfig
  StepSize 0.0913
  SuperCell
    2 0 0
    0 2 0
    0 0 2
  SubEnd
End
```

In the standard output the thermodynamic functions are printed:

```
========================
ThermoDynamic Properties
========================
Zero-point Energy (Hartree)         0.00448537
Zero-point Energy (eV)              0.12205313


==============================================================================================================
Temperature(K)  Internal energy(Hartree)              Entropy(kB)      Free␣
→Energy(Hartree)        Specific Heat(kB)

  0.000000E+00           0.448537E-02              0.000000E+00              0.
→448537E-02           0.000000E+00
  0.100000E+01           0.448540E-02              0.116921E-01              0.
→448536E-02           0.329896E-01
  0.200000E+01           0.448566E-02              0.643295E-01              0.
→448525E-02           0.138532E+00
  ...                      ...                       ...                      ...     ␣
→                ...
```

7.2 Time-Dependent Current DFT

Time-Dependent Current Density Functional Theory (**TD-CDFT**) is a theoretical framework for computing optical response properties, such as the frequency-dependent dielectric function.

In this section, the TD-CDFT implementation for extended systems (1D, 2D and 3D) in BAND is described. The input keys are described in *NewResponse* (page 56) or in *OldResponse* (page 58).

Some examples are available in the $ADFHOME/examples/band directory and are discussed in the Examples section.

Example: TD-CDFT for bulk silicon (OldResponse) (page 163)	Tutorial: Silicon (OldResponse)
Example: TD-CDFT for MoS2 Monolayer (NewResponse) (page 161)	Tutorial: MoS2 Monolayer (NewResponse)
Example: TD-CDFT for hydrogen chain (OldResponse) (page 167)	
Example: TD-CDFT for a metal (OldResponse) (page 169)	

7.2.1 Insulators, semiconductors and metals

The TD-CDFT module enables the calculation of real and imaginary parts of the material property tensor $\chi_e(\omega)$, called the **electric susceptibility**. The electric susceptibility is related to the macroscopic **dielectric function**, $\varepsilon_M(\omega)$.

For semi-conductors and insulator, for which the bands are either fully occupied or fully unoccupied, the dielectric function $\varepsilon_M(\omega)$ comprises only of the so called interband component:

$$\varepsilon_M(\omega) = 1 + 4\pi\chi_e(\omega)$$

In general $\chi_e(\omega)$ and $\varepsilon_M(\omega)$ are tensors. They, however, simplify to scalars in isotropic systems.

For metals, for which partially-occupied bands exist, there is a so called intraband component arising due to transitions within a partially-occupied band:

$$\varepsilon_M(\omega) = 1 + 4\pi\chi_e(\omega) - 4\pi i\sigma_e(\omega)/\omega$$

7.2.2 Frequency dependent kernel

It is known that the exact Vignale-Kohn (VK) kernel greatly improves the static polarizabilities of infinite polymers and nanotubes (see reference (http://dx.doi.org/10.1063/1.2102899)), but gives bad results for the optical spectra of semiconductors and metals. For the low frequency part one needs a frequency dependent kernel, since Drude-like tails are completely absent in the adiabatic local density approximation (ALDA). With a modified VK kernel, which neglects μ_{xc} so that it reduces to the ALDA form in the static limit (see reference (http://dx.doi.org/10.1103/PhysRevB.74.245117)), much better results can be obtained. BAND currently only supports the modified VK kernel in either the QV or CNT parametrization, and it should **only be used for metals**.

7.2.3 EELS

From the macroscopic dielectric function it is possible to calculate the electron energy loss function (EELS). In transmission EELS one studies the inelastic scattering of a beam of high energy electrons by a target. The scattering rates obtained in these experiments are related to the dynamical structure factor $S(q, \omega)$ [A1]. In the special case with wavevector $q = 0$, $S(q, \omega)$ is related to the longitudinal macroscopic dielectric function. This is the long-wave limit of EELS. For isotropic system the dielectric function is simply a scalar $(1/3\text{Tr}(\varepsilon_M(\omega)))$. In this case the long-wave limit of the electron energy loss function assumes the trivial form

$$\lim_{q \to 0} 2\pi \frac{S(q, \omega)}{q^2 V} = \frac{\varepsilon_2}{\varepsilon_1^2 + \varepsilon_2^2}$$

with ε_1 and ε_2 as real and imaginary part of the dielectric function.

References

The three related Ph.D. theses, due to F. Kootstra (on TD-DFT for insulators), P. Romaniello (on TD-CDFT for metals), and A. Berger (on the Vignale-Kohn functional in extended systems) contain much background information, and can be downloaded from the SCM website (http://www.scm.com).

The most relevant publications on this topic due to the former "Groningen" group of P.L. de Boeij are [22 (page 206)], [23 (page 206)], [24 (page 206)], [25 (page 206)].

[A1] S. E. Schnatterly, in Solid State Physics Vol.34, edited by H. Ehrenreich, F. Seitz, and D. Turnbull (Academic Press, Inc., New York, 1979).

7.2.4 Input Options

In the 2017 release of BAND there are two implementations of the TD-CDFT formalism. The original implementation, relying on obsolete algorithms of BAND, is accessible via the *OldResponse* (page 58) key block. The new code section, relying on more modern algorithms of BAND, is accessible via the *NewResponse* (page 56), *NewResponseSCF* (page 57) and *NewResponseKSpace* (page 57) key blocks. The differences between the two flavours are summarized in the following table:

	OldResponse	NewResponse
3D-systems	yes	yes
2D-systems	no	yes
1D-systems	(yes)	yes
Semiconductors	yes	yes
Metals	yes	no
ALDA	yes	yes
Vignale-Kohn	yes	no
Berger2015 (3D)	yes	yes
Scalar ZORA	yes	yes
Spin Orbit ZORA	yes	no

Besides these differences, one should not expect both flavours to give the exact same result, if the reciprocal space limit is not reached! This can be explained by different approaches to evaluate the integration weights of single-particle transitions in recirpocal space.

> **Attention:** Response properties **converge slowly** with respect to k-space sampling (number of k-points). **Always check the convergence of** ε_M **with respect to** *KSPACE* (page 39) **options!!!**

NewResponse

The dielectric function is computed when the key block *NewResponse* (page 56) is present in the input. Several important settings can be defined in this key block.

Additional details can be specified via the *NewResponseKSpace* (page 57) and *NewResponseSCF* (page 57) key blocks.

NewResponse Perform a TD-CDFT calculation to obtain real and imaginary parts of the frequency-dependent dielectric function. (New version)

```
NewResponse
    {nFreq        ival1}
    {FreqLow      rval1}
    {FreqHigh     rval2}
    {EShift       rval3}
    {ActiveESpace rval4}
    {DensityCutOff rval5}
    {ActiveXYZ    string}
End
```

Omitting the specific options in the NewResponse block will cause default settings to be used during the calculation, as given below.

nFreq (**Default: 5**) the number of frequencies for which a linear response TD-CDFT calculation is performed.

FreqLow (**Default: 1.0**, unit: eV) ω_{Low} is the lower limit of the frequency range for which response properties are calculated.

FreqHigh (**Default: 3.0**, unit eV) ω_{High} is the upper limit of the frequency range for which response properties are calculated.

EShift (**Default: 0.0**, unit eV) energy shift of the virtual crystal orbitals.

ActiveESpace (**Default: 5.0**, unit eV) Modifies the energy threshold ΔE_{thresh}^{max} for which single orbital transitions $\Delta\varepsilon_{ia}$ are taken into account. There is a lower threshold ΔE_{thresh}^{min}, too.

$$\Delta E_{thres}^{max} = \omega_{High} + E_{active}$$
$$\Delta E_{thres}^{min} = max\left(\omega_{Low} - E_{active}, 0.0\right)$$
$$\Delta\varepsilon_{ia} = \varepsilon_a(virtual) - \varepsilon_i(occupied)$$

DensityCutOff (**Default: 1E-3**) For 1D and 2D systems the unitcell volume is undefined. Here, the volume is calculated as the volume bordered by the isosurface for the value **rval5** of the total density.

ActiveXYZ (**Default: T T T**) expects a string consisting of three letters of either 'T' (for true) or 'F' (for false) where the first is for the X-, the second for the Y- and the third for the Z-component of the response properties. If true, then the response properties for this component will be evaluated.

NewResponseSCF Modify the details for the linear-response self-consistent optimization cycle. Only influencing the NewResponse code!

```
NewResponseSCF
   {nCycle      ival1}
   {Mixing      rval1}
   {Criterion   rval2}
   {XC          ival2}
   {Bootstrap   ival3}
   {DIIS        lval1}
End
```

nCycle (**Default: 20**) number of SCF cycles for each frequency to be evaluated.

Mixing (**Default: 0.2**) mixing value for the SCF optimization.

Criterion (**Default: 0.001**) for the SCF convergence the RMS of the induced density change is tested. If this value is below the `Criterion` the SCF is finished. Furthermore, one can find the calculated electric susceptibility for each SCF step in the output and can therefore decide if the default value is too loose or too strict.

XC (**Default: 1**) influences if the bare induced Coulomb response (*XC 0*) is used for the effective, induced potential or the induced potential derived from the ALDA kernel as well (*XC 1*).

Bootstrap (**Default: 0**) defines if the Berger2015 kernel (*Bootstrap 1*) is used or not (*Bootstrap 0*). If you chose the Berger2015 kernel, you have to set `NewResponseSCF%XC` to '0'. Since it shall be used in combination with the bare Coulomb response only. **Note:** The evaluation of response properties using the Berger2015 is recommend for 3D systems only!

DIIS (**Default: true**) in case the DIIS method is not working, one can switch to plain mixing by setting DIIS to *false*.

NewResponseKSPACE Modify the details for the integration weights evaluation in reciprocal space for each single-particle transition. Only influencing the NewResponse code!

```
NewResponseKSPACE
   {Eta      rval1}
   {SubSimp  ival1}
End
```

Eta (**Default: 1E-5**) determines the small, finite imaginary number $i \cdot \eta$, which is necessary in the context of integration weights for single-particle transitions in reciprocal space.

SubSimp (**Default: 3**) determines into how many subintegrals each integration around a k point is split. This is only true for so-called quadratic integration grids. The larger the number the better the convergence behavior for the sampling in reciprocal space.

> **Attention:** computing time for the weights is influced linear for 1D, quadratic for 2D and cubic for 3D!

OldResponse

OldResponse Perform a TD-CDFT calculation to obtain real and imaginary parts of frequency-dependent dielectric function. (Old/Original version)

```
OldResponse
   {nfreq   ival1}
   {strtfr  rval1}
   {endfr   rval2}
   {shift   rval3}
   {cnvi    rval4}
   {cnvj    rval5}
   {ebndtl  rval6}
   {isz     ival2}
   {iyxc    ival3}
   {static}
   {newvk}
   {cnt}
   {qv}
   {Berger2015}
End
```

Omitting the specific options in the OldResponse block will cause default settings to be used during the calculation, as given below.

nfreq (**Default: 5**) the number of frequencies for which a linear response TD-CDFT calculation is performed.

strtfr (**Default: 1.0**, unit eV) is the lower bound frequency of the frequency range over which the dielectric function is calculated.

endfr (**Default: 3.0**, unit eV) is the upper bound frequency of the frequency range over which the dielectric function is calculated.

shift (**Default: 0.0**, unit eV) energy shift for the virtual crystal orbitals.

cnvi (**Default: 1d-3**) the first convergence criterion for the change in the fitcoefficients for the fitfunctions, when fitting the density.

cnvj (**Default: 1d-3**) the second convergence criterion for the change in the fitcoefficients for the fitfunctions, when fitting the density.

ebndtl (**Default: 1d-3**, unit au) the energy band tolerance, for determination which routines to use for calculating the numerical integration weights, when the energy band posses no or to less dispersion.

isz (**Default: 0**) integer indicating whether or not scalar zeroth order relativistic effects are included in the TDCDFT calculation. 0 = relativistic effects are not included, 1 = relativistic effects are included.

> **Attention:** The current implementation does **NOT** work with the option `XC%SpinOrbitMagnetization` equal **NonCollinear**!

iyxc (**Default: 0**) integer for printing yxc-tensor (see reference (http://dx.doi.org/10.1063/1.1385370)). 0 = not printed, 1 = printed.

static An alternative method that allows an analytic evaluation of the static response (normally the static response is approximated by a finite small frequency value). This option should only be used for non-relativistic calculations on insulators, and it has no effect on metals. **Note:** experience shows that KSPACE convergence can be slower.

newvk Use the slightly modified version of the VK kernel.[*58* (page 208)] When using this option one uses effectively the static option, even for metals, so one should check carefully the convergence with the KSPACE parameter.

qv/cnt Use the **QV** or **CNT** parametrization for the longitudinal and transverse kernels of the XC kernel of the homogeneous electron gas. Use this in conjunction with the newvk option.[*60* (page 208), *61* (page 208)] An example response block for a vignale kohn calculation looks as follows:

```
OldResponse
   {...}
   newvk
   iyxc    1
   cnt
END
```

Berger2015 Use the parameter-free polarization functional by A. Berger.[*59* (page 208)] This is possible for 3D insulators and metals. **Note:** The evaluation of response properties using the Berger2015 is recommend for 3D systems only!

```
OldResponse
   {...}
   Berger2015
END
```

7.3 ESR/EPR

BAND is capable to calculate electron paramagnetic resonance (EPR) parameters for paramagnetic defects in solids: hyperfine A-tensor and the Zeeman g-tensor.

The implementation of EPR parameters in BAND is described in the publications by Kadantsev and co-workers [*20* (page 206)] and [*21* (page 206)].

Hyperfine A-tensor

The A-tensor is implemented within the non-relativistic and scalar relativistic, spin-polarized Kohn-Sham scheme. The A-tensor calculation is invoked by the blockkey:

```
ATENSOR
END
```

Note: the `Unrestricted` keyword should be present.

Two methods are used for A-tensor calculation:

- Method 1: involves the gradient of the spin-polarized density and integration by parts. The isotropic component of the A-tensor obtained through integration, in a "non-local fashion".

- Method 2: the A-tensor is computed from spin-polarized density and does not relies on the integration by parts. The isotropic component is obtained in a "local fashion" from the value of the spin-polarized density on the grid points near the nuclei.

The user should be aware that numerical integration in A- and g-tensor routines is carried out over the Wigner-Seitz (WS) cell, and, therefore, to obtain a meaningful result, the defect in question should lie at or very close to the WS cell origin. This might require, on the user's part, some modification of the input geometry.

It also might happen that the size of the WS cell is not large enough for the adequate description of the paramagnetic defect in question. In this case, Method 1 will fails, since it relies on the integration by parts and assumes that the spin-polarized density is localized inside the WS cell. For the same reason, we recommend that the user removes diffuse basis set functions that describe the defect subsystem.

Finally, we note that the final result for A-tensor as presented by BAND is not scaled by the nuclear spin (as it is done in ADF) and the user is responsible for making necessary adjustments.

g-tensor

The calculation of the Zeeman g-tensor is invoked with the blockkey:

```
ESR
END
```

The Zeeman g-tensor is implemented using two-component approach of Van Lenthe and co-workers in which the g-tensor is computed from a pair of spinors related to each other by time-reversal symmetry.

Note: The keyword `Relativistic zora spin` should be present to invoke calculations with spin-orbital coupling. The user also has to specify

```
Kspace 1
```

(Γ-only calculation). The g-tensor is then computed from the HOMO spinor at the Γ point. In the output, the user can find two-contributions to the g-tensor: one that stems from the K_σ operator and a second one, that stems from the orbital angular momentum. By default, GIAO and spin-Zeeman corrections are **not** included. From our experience, these corrections are quite small.

7.4 Nuclear Quadrupole Interaction (EFG)

```
EFG (block type)
```

The electronic charge density causes an electric field, and the gradient of this field couples with the nuclear quadrupole moment, that some (non-spherical) nuclei have and can be measured by several spectroscopic techniques. The EFG tensor is the second derivative of the Coulomb potential at the nuclei. For each atom it is a 3x3 symmetric and traceless matrix. Diagonalization of this matrix gives three eigenvalues, which are usually ordered by their decreasing absolute size and denoted as $|V_{xx}|$, $|V_{yy}|$, $|V_{zz}|$. The result is summarized by the largest eigenvalue and the asymmetry parameter.

This option honors the `SelectedAtoms` key, in which case only the EFG will be calculated for the selected atoms.

7.5 NMR

With the NMR option the *shielding tensor* is calculated. There are two methods implemented: the super cell method and the single-dipole method.

1. The super cell method is according to the implementation by Skachkov *et al.* [*37* (page 207)] The symmetry will automatically be set to NOSYM. The unit cell should not be chosen too small.

2. The other method is the single-dipole method. In principle one can now use the primitive cell.[*48* (page 207)] In practice also this method needs to be converged with super cell size. However, depending on the system the required super cell may be much smaller. At a given super cell size this method is more expensive than the super cell method.

```
NMR (block type)
   {SuperCell [true | false]}
End
```

SuperCell (**Default: true**) This is the switch between the two methods, either the super cell (true), or the single-dipole method (false).

This option honors the SelectedAtoms key, in which case only the NMR properties will be calculated for the selected atoms only.

7.6 Effective Mass

EffectiveMass (block-type) In a semi-conductor the mobility of electrons and holes is related to the curvature of the bands at the top of the valence band and the bottom of the conduction band. With the effective mass option, this curvature is obtained by numerical differentiation. The estimation is done with the specified step size, and twice the specified step size, and both results are printed to give a hint on the accuracy. By far the most convenient way to use this key is without specifying any subkeys.

```
EffectiveMass
   StepSize step
   NumAbove na
   NumBelow nb
   UniqueKPoints k1 k2 ...
End
```

StepSize (**Default: 1e-3**) Size of the step taken in reciprocal space to perform the numerical differentiation.

NumAbove (**Default: 1**) Number of bands to take into account above the Fermi level.

NumBelow (**Default: 1**) Number of bands to take into account below the Fermi level.

UniqueKPoints List of unique k-points where to print the effective mass. If this option is omitted, it will use the top of valence band and the bottom of the conduction band.

7.7 Properties at Nuclei

PropertiesAtNuclei (block-type) A number of properties can be obtained near the nucleus. An average is taken over a tiny sphere around the nucleus. The following properties are available.

```
PropertiesAtNuclei
  vxc[rho(fit)]
  rho(fit)
  rho(scf)
  v(coulomb/scf)
  rho(deformation/fit)
  rho(deformation/scf)
End
```

The electron density, `rho(scf)`, is physically the most relevant one.

7.8 X-Ray Form Factors

FormFactors X-ray structure factors (Fourier analysis of the charge density) are computed after termination of the SCF procedure.

```
{FormFactors ival}
```

The key is followed by an integer specifying the number of stars of K-vectors for which the structure factors are computed. (**Default: 2**)

ELECTRONIC TRANSPORT (NEGF)

See also:

BAND-NEGF GUI tutorial

Some examples are available in the `$ADFHOME/examples/band` directory and are discussed in the Examples section.

> *Example: Main NEGF flavors* (page 136)
> *Example: NEGF with bias* (page 143)

Note: In the BAND-GUI it is possible to choose between three NEGF methods (*flavors*):

Self consistent This is the internal BAND-NEGF implementation, which is described in this page.

Self consistent + align This is the internal BAND-NEGF implementation with an extra alignment-run (workflow step 3a)

Non self consistent Computationally cheap method, equivalent to the DFTB-NEGF approach with H and S matrix elements computed by BAND (instead of DFTB).

8.1 Transport with NEGF in a nutshell

The **Non-Equilibrium Green's Functions** formalism (**NEGF**) is a theoretical framework for modeling electron transport through nano-scale devices. Electron transport is treated as a one-dimensional coherent scattering process in the "scattering region" for electrons coming in from the electrodes:

Scattering region

Left lead (L) — Central region (C) — Right lead (R)

Electrode — Electrode

2 lead repetitions

Direction of electron transport (x-axis)
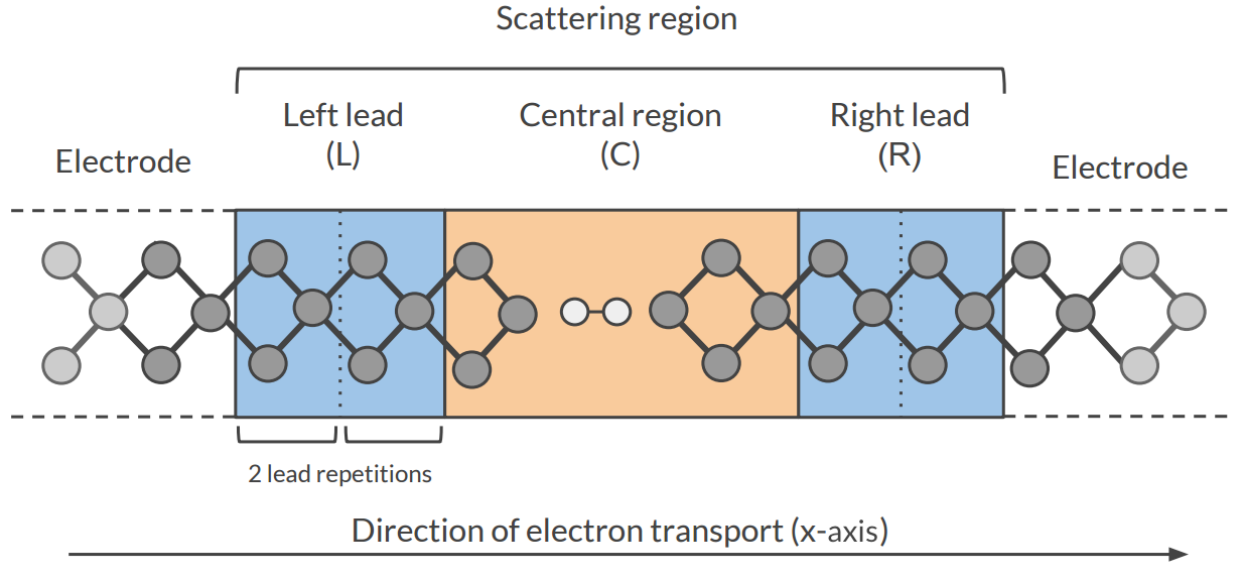
Our goal is to compute the **transmission function** $T(E)$, which describes the rate at which electrons of energy E are transferred from the left electrode to the right electrode by propagating through the scattering region. From the transmission function we can calculate the electric current for given **Bias Voltage** V applied between the electrodes:

$$I(V) = \frac{2e}{h} \int_{-\infty}^{\infty} T(E, V) \left(f(E - \mu_L) - f(E - \mu_R) \right) dE$$

where $f(E)$ is the Fermi-Dirac distribution function for a given temperature, and μ_L (μ_R) is $\epsilon_F + eV/2$ ($\epsilon_F - eV/2$), ϵ_F being the Fermi energy of the electrodes.

The transmission function $T(E)$ can be computed from the **Green's function** of our system.

The Green's function $G(E)$ of the scattering region is obtained solving the following equation:

$$(ES - H)G(E) = I$$

where S is the overlap matrix, H is the Hamiltonian and I is the identity matrix. The Hamiltonian is composed as follows (**L**, **C** and **R** denote the **left lead**, the **central region** and the **right lead** respectively):

$$H = \begin{pmatrix} H_L + \Sigma_L & H_{LC} & 0 \\ H_{LC} & H_C & H_{RC} \\ 0 & H_{RC} & H_R + \Sigma_R \end{pmatrix}$$

The two *self-energies* Σ_L and Σ_R model the two semi-infinite electrodes.

The transmission function $T(E)$ can be calculated from the Green's function $G(E)$ and the so-called *broadening matrices* $\Gamma_L(E)$ and $\Gamma_R(E)$:

$$T(E) = Tr[G(E)\Gamma_R(E)G(E)\Gamma_L(E)]$$

The broadening matrix being

$$\Gamma_L(E) = -2\Im\Sigma_L(E)$$

8.2 Self consistency

The density matrix is determined self consistently [68 (page 209)]:

$$P_{\text{in}} \rightarrow H_{KS} \xrightarrow{\text{shifts}} H_{\text{aligned}} + \Sigma_L(E) + \Sigma_R(E) \rightarrow G(E) \xrightarrow{\int de} P_{\text{out}}$$

From a guess of the density matrix the corresponding KS Hamiltonian is calculated. This Hamiltonian is aligned, and then the NEGF Hamiltonian in the complex plane is constructed by adding the self energies, representing the influence of the electrodes. From the resulting Green's function a new density matrix follows.

From the difference between input and output density a next input is guessed. This is repeated until the input and output densities converge.

For the alignment of the Hamiltonian there are two shifts. The first shift aligns the potential in the leads to the electrodes.

$$\text{shift } 1 = \frac{1}{n} \sum_{i \text{ in lead}}^{n} \frac{H_{ii}^{TB} - H_{ii}^{KS}}{S_{ii}}$$

The second and usually smaller shift results from the alignment run. A shift Δ is applied globally

$$H_{ij}^{\text{aligned}} = H_{ij} + \Delta S_{ij}$$

8.3 Contour integral

Without bias the density matrix follows from

$$P(\mu) = -\frac{1}{\pi} \int_{-\infty}^{\infty} de \, f(e, \mu) \, \Im G(e)$$

As the Green's function is singular on the real axis we add a small imaginary value (**eta**) to the energy. Still, the integrand will be very wild function, and it is numerically better to do a contour integral instead.
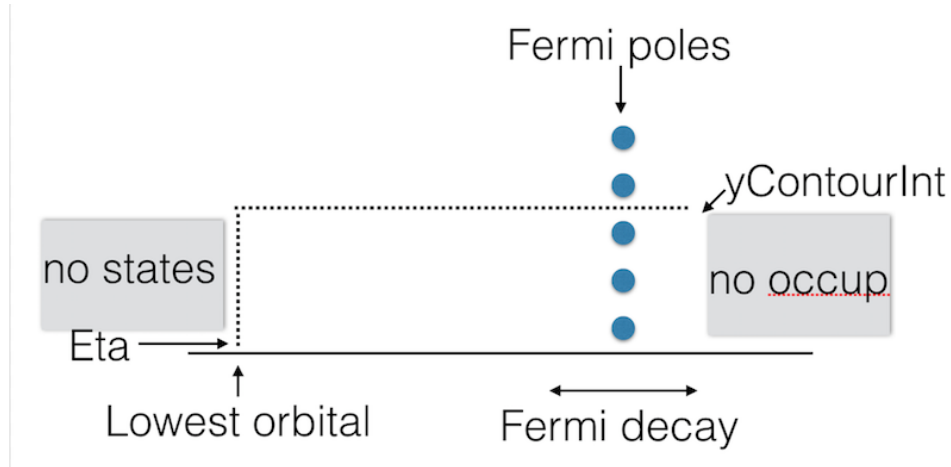
Fig. 8.1: Figure: BAND uses a rectangular contour in the complex energy plane to integrate the (integrand of the) density matrix. The integrand also needs to be evaluated in the enclosed FD poles (three in this picture).

8.4 Gate potential

There is no direct key for the gate potential. You can model this with the *FuzzyPotential* (page 29) key. Setting up the gate potential for NEGF is most conveniently done with the GUI.

8.5 Bias potential
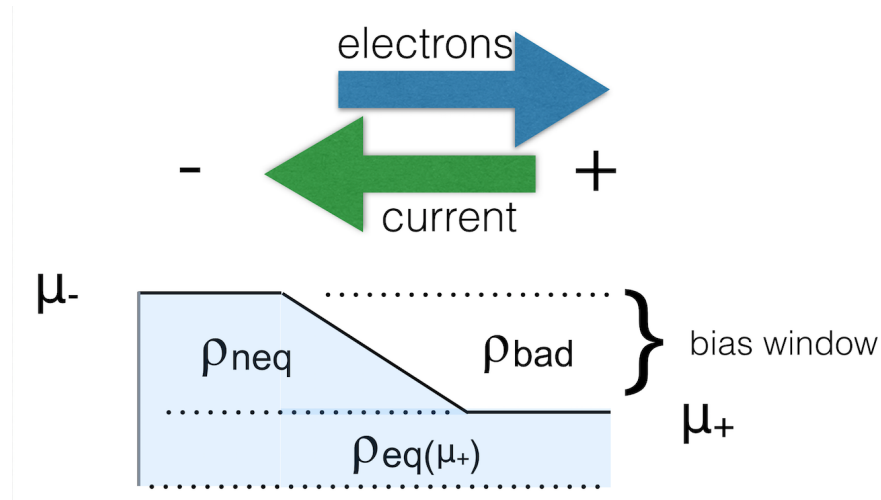
When there is a bias specified there are two important things to keep in mind.

First of all you need to define a ramp potential. In the negative lead this should have the value +V/2 and in the negative lead -V/2. The ramp should smoothly go from one to the other value. For metals one could start the ramp at the surface atoms of the lead material. For semi-conductors it is less clear. The ramp potential can be specified with the *FuzzyPotential* (page 29) key. The GUI can be helpful here.

Secondly, the expression for the density is different from the zero-bias case:

$$\rho = \rho_{eq}(\mu_+) + \rho_{neq}$$

The first (equilibrium) term is calculated with a contour integral as before, the second (non-equilibrium) part cannot be calculated with a contour integral. Instead, an integral in the complex plane (close to the real axis) is performed, the range covering the bias energy window.

See also:

PhD Thesis (https://www.scm.com/wp-content/uploads/Verzijl2012.pdf) of C. Verzijl (BAND-NEGF developer)

8.6 Workflow

The computation of the transmission function $T(E)$ within the BAND-NEGF [*68* (page 209)] formalisms requires three or four individual simulations.

Tip: Use ADFInput (GUI) to set up your BAND-NEGF calculation (see the BAND-NEGF GUI tutorial)

---

1): Lead calculation A 1D-periodic BAND calculation of the lead (including *StoreHamiltonian2* (page 71)):

Left lead (L) Right lead (R)

Lattice vector
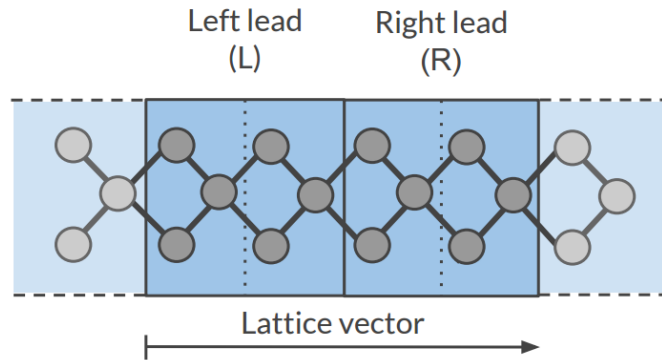
A tight binding (TB) representation is calculated for the overlap ($S(R = 0)$ and $S(R = a)$) and Fock matrix ($H(R = 0)$ and $H(R = a)$). This is not an approximation provided that the functions do not extend beyond the neighboring cells. You should choose a sufficiently large super cell for this to be true. For this reason we recommend setting the *SoftConfinement* (page 36) Quality to Basic, thus reducing the range of the functions.

2): SGF calculation A small program that determines the fermi energy ϵ_F corresponding to the TB representation, and the specified temperature. This fermi energy is typically a bit higher than the one from the lead calculation. This also tests the contour integration.

3a): Alignment run (optional) The idea is to fill the central region with bulk material. Then one expects to have zero charge in the central region. In practice this is not exactly true. In the alignment run the shift is determined that makes the central region neutral. This global shift is to be used in the next run.

3b): Transport calculation Computes the NEGF transmission function $T(E)$. The density matrix is determined fully self-consistently. Without alignment (3a) one should set `NEGF%ApplyShift2` to `False`.

To get the current as a function of bias potential you need to repeat calculation 3b for a various bias potentials.

8.7 SGF Input options

SGF is a small separate program. An input looks like:

```
$ADFBIN/sgf    << eor
TITLE Test for NEGF inputs
SAVE SIGMA
SURFACEGF
SCMCode
   KT 0.001
   ContourQuality normal
END
eor
```

It looks for a file `RUNKF` and the output is a file named `SigmaSCM`. The only important parameter is KT which is the Boltzmann constant times the temperature in Hartree. The other parameter of interest is the ContourQuality, which can be set to Basic,Normal,Good,VeryGood, or Excellent.

8.8 NEGF Input options (no bias)

The NEGF functionality is controlled by the NEGF block key:

```
NEGF
  LeadFile TB.runkf
  SGFFile Sigma.runkf
  ContourQuality [Basic|Normal|Good|VeryGood]
  EMin -0.1837465475
  EMax 0.1837465475
  NE 200
  ! From here one the options are very technical
  CheckOverlapTol rval
  Eta rval
  ApplyShift1 True
  ApplyShift2 False
  yContourInt rval
  deContourInt rval
End
```

LeadFile Contains the tight binding representation of the lead.

SGFFile The result from the SGF program. Contains the fermi energy of the lead.

ContourQuality Default: normal. The density matrix is calculated numerically via a contour integral. Setting this to Basic,Normal,Good,VeryGood influences the number of points. This influences a lot the performance.

NE The nr. of energies for the transmission energy grid.

EMin The minimum energy for the transmission grid (with respect to the fermi level of the lead).

EMax The maximum energy for the transmission grid.

The remaining options are very technical in nature.

CheckOverlapTol Default: 0.01 BAND checks how well the TB overlap matrix :math:S(R=0) represents the overlap matrix in the lead region. Elements corresponding to the outer layer are neglected, because when using a frozen core they have bigger errors.

ApplyShift1 Default: True Apply the main shift, obtained from comparing matrix elements in the leads with those from the tight-binding run. Strongly recomended.

ApplyShift2 Default: True Apply the smaller alignment shift. This requires an extra alignment run. Usually this shift is smaller.

Eta Default: 1e-5 Small value used for the contour integral: stay at least this much above the real axis. This value is also used for the evaluation of the Transmission and dos.

yContourInt Defaults: 0.3 The density is calculated via a contour integral. This value specifies how far above the real axis the (horizontal part of the) contour runs. The value is rounded in such a way that it goes exactly halfway between two fermi poles. There is a trade off: making it bigger makes the integrand more smooth, but the number of enclosed poles increases. For low temperatures it makes sense to lower this value, and use a smaller deContourInt.

deContourInt The energy interval for the contour grid. Defaults depends on the contour quality.

8.9 NEGF Input options (with bias)

With a bias potential there are some extra keys.

```
NEGF
  BiasPotential rval
  NonEqDensityMethod [1|2|3]
```

```
  BoundOccupationMethod [1|2|3]
  yRealAxisInt rval
  deRealAxisInt rval
End
```

BiasPotential Apply a bias potential (atomic units). Can be negative. One has to specify the ramp potential with the *FuzzyPotential* (page 29) key. This is mostly conveniently done with the GUI.

NonEqDensityMethod Default: 1 Let us introduce some terms [*66* (page 208)]. First of all the total density in the bias window (ignoring occupation)

$$D = \frac{1}{2\pi} \int A \left(f_- - f_+ \right)$$

And then there are the side resolved densities

$$D_{+/-} = \frac{1}{2\pi} \int A_{+/-} \left(f_- - f_+ \right)$$

The issue here is that the side resolved densities do not sum to the total one

$$D = D_+ + D_- + D_{\text{bound states}}$$

The NonEqDensityMethod is about how these integrals are calculated . With option 1, or 2 a contour integral is used for D: they are essentially the same. However, when choosing option 2, you can choose a BoundOccupationMethod, leading to other physics. If set to 3, the total density in the bias window (D) will be calculated near the real axis: this way one avoids the possibility of a negative nr. of bound states (deviating from [*66* (page 208)]).

BoundOccupationMethod Default: 1 Only relevant with NonEqDensityMethod equal 2 or 3. If set to one, the density of bound states (ignoring occupation) is simply multiplied by a half. If set to two, atoms closer to the negative lead will get a higher occupation [*66* (page 208)]. Atoms coupled to the right lead will have a low occupation. For this we recommend setting NonEqDensityMethod to 3, to avoid a possible negative number of bound states.
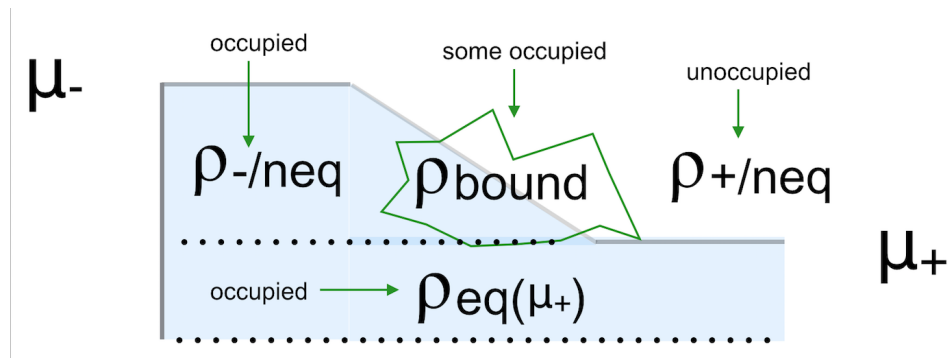
Fig. 8.2: Figure: The non-equilibrium density consists of three parts: the left and right parts (ρ-/neq and ρ+/neq) and the bound states (ρbound). We want to know the occupied part.

Setting the method BoundOccupationMethod to 1, leads to

$$\rho = \rho(\mu_+) + \rho_{\text{-/neq}} + \frac{1}{2}\rho_{\text{bound}}$$

By setting the method to 2, each atom gets its own weight in the density matrix

$$\rho_{ij} = \rho_{ij}(\mu_+) + \rho_{\text{-/neq}} + \sqrt{w_i w_j} \rho_{ij}^{\text{bound}}$$

with [*66* (page 208)]

$$w_i = \frac{\text{Tr}[D_-]_i}{\text{Tr}[D_-]_i + \text{Tr}[D_+]_i}$$

These weights are the same for all functions on an atom. The intended effect is: bound states that are coupled more strongly to the negative electrode get a higher occupation than the ones that are coupled more strongly to the positive electrode.

To summarize here are three reasonable settings

NonEqDensityMethod	BoundOccupationMethod	intention
1	1	Multiply the bound states with a half
2	2	Occupy bound states with atom-resolved w_i
3	2	... and prevent a negative nr. of bound states

yRealAxisInt **Defaults: 1e-5** The non-Equilibrium density is calculated near the real axis.

deRealAxisInt The energy interval for the real axis grid. Defaults depends on the contour quality.

To get the current from a calculation you can use adfprep:

```
$ADFBIN/adfreport RUNKF 'NEGF%current'
```

8.10 NEGF Input options (alignment)

For the (optional) alignment run there are some extra keys.

```
NEGF
  DoAlignment True
  Alpha rval
  AlignChargeTol rval
  CDIIS [True|False]
End
```

DoAlignment **Default: False**. Set this to True if you want to do an align run. Between the leads there should be lead material. The GUI can be of help here.

Alpha **Default: 1e-5**. A charge error needs to be translated in a potential shift.

$$\Delta V = \alpha \Delta Q$$

AlignChargeTol **Default: 0.1** In an alignment run you want to get the nr. of electrons in the center right. This number specifies the criterion for that.

CDIIS **Default: False**. Make the normal DIIS procedure aware of the align charge error.

8.11 Troubleshooting

The self consistent approach, unique to BAND, may be difficult to converge. If this is true for the alignment run it can be decided to skip this run. For the final transport run, here are some tips / considerations.

- Use a SZ basis for the metal atoms

- Restart (the density matrix) from the result of a smaller (such as the SZ) basis. (See "*Save* (page 93) Density-Matrix" and the *Restart* (page 87) key)

- Restart (the density matrix) from the result obtained with a smaller bias (only relevant for calculations with bias potential).

- Setting NEGF%BoundOccupationMethod to 2 (and NEGF%NonEqDensityMethod to 3) might help. Note that this affects the physics: you are differently occupying the bound states.

- Use a better NEGF%ContourQuality (there comes a computational price tag with this).

If everything fails it is possible to use BAND in a **non-self consistent way**, similar to the way DFTB-NEGF works. This option is available via the GUI.

8.12 Miscellaneous remarks on BAND-NEGF

- You should make sure that your results are converged with respect to the number of lead repetitions; the results should not change significantly if you increase the number of lead repetitions.

- It's good practice to include at least one lead repetition in the central region.

8.13 Store tight-binding Hamiltonian

Let us consider a Fourier transformation of a 1D Bloch matrix

$$S(R = na) = \int_k e^{-ikR} S(k)$$

In (the tight-binding) case that the functions do not extend beyond the neighboring cells only S(R=0) and S(R=a) are nonzero. (And S(R=-a) is equivalent to S(R=a))

```
StoreHamiltonian2
```

Adding StoreHamiltonian2 to the input cause band to determine the tight-binding representation of the overlap an fock matrix. Currently this only works for 1D periodic systems. For the overlap matrix you will get two parts. The first $S(R = 0)$ is the (symmetric) overlap matrix of atoms in the unit cell. The second $S(R = a)$ is a non symmetric matrix describing the coupling of functions in the central cell with functions in its right neighboring cell. On the RUNKF file you will find the TB representations of the overlap and Hamiltonian stored in the 'Matrices' section as "S(R)" and "H(R)", being dimensioned (nBas,nBas,2).

ANALYSIS

9.1 Density of States (DOS)

DOS (block-type) General Density-Of-States (DOS) information.

```
DOS
   {Enabled [True|False]}
   {DeltaE value}
   {Energies n}
   {Min value}
   {Max value}
   {IntegrateDeltaE [True|False]}
   {File filename}
End
```

Enabled (**Default: False**) Whether or not to calculate the density of states.

DeltaE (**Default: 0.005** a.u.) Energy step for DOS grid. Using a smaller value (e.g. 0.001) will result in a finer sampling of the DOS.

Energies Number of equidistant energy-values. This keyword supersedes the `DeltaE` keyword.

Min (**Default:** Energy of the lowest band) lower bound energy (w.r.t. Fermi level) in a.u.

Max (**Default:** Fermi energy + 0.75 a.u.) upper bound energy (w.r.t. Fermi level) in a.u.

IntegrateDeltaE (**default=true**) This subkey handles which algorithm is used to calculate the data-points in the plotted DOS.

- `IntegrateDeltaE true`: the data-points represent an integral over the states in an energy interval. Here, the energy interval depends on the number of Energies and the user-defined upper and lower energy for the calculation of the DOS. The result has as unit [number of states / (energy interval * unit cell)].

- `IntegrateDeltaE false`: the data-points do represent the number of states for a specific energy and the resulting plot is equal to the DOS per unit cell (unit: [1/energy]). Since the resulting plot can be a wild function and one might miss features of the DOS due to the step length between the energies, the default is set to the integration algorithm.

File (**Optional**) handles the name of the output file with the DOS results.

An example input:

```
DOS
   Enabled      True
   Energies     500
   Min         -0.35
```

```
    Max          1.05
    File         plotfile
End
```

According to this example, DOS values will be generated in an equidistant mesh of 500 energy values, ranging from 0.35 a.u. below the Fermi level to 1.05 a.u. above it. All information will be written to a file plotfile. The information on the plot file is a long list of pairs of values (energy and DOS), with some informative text-headers and general information. DOS values are generated for the total DOS and optionally also for some partial DOS (see the keys *GrossPopulations* (page 74) and *OverlapPopulations* (page 74)).

In the **DOS** and **Band Structure GUI modules**, it is possible to visualize partial density of states (**p-DOS**). The partial contributions are obtained from the total DOS by following the **Mulliken population analysis** partitioning prescription (see wikipedia (https://en.wikipedia.org/wiki/Mulliken_population_analysis)).

9.1.1 Gross populations

GrossPopulations (block-type) *Partial* DOS (**pDOS**) are generated for the gross populations listed under this key.

```
GrossPopulations
   {iat lq}
   {FragFun jat ifun}
   {Frag kat}
   {Sum
    ...
    EndSum}
End
```

> **iat** pDOS is generated for atom *lq*.

> **FragFun** pDOS is generated for atom *jat* with all real spherical harmonics belonging to *l*-value *ifun*.

> **Frag** pDOS of the functions belonging to atom *kat* will be calculated.

> **Sum** sum all pDOS, specified in this block.

Example:

```
GrossPopulations
   FragFun 1 2:: Second function of first atom
   Frag 2 :: Sum of all functions from second atom
   SUM:: sum following PDOSes
      Frag 1::Atom nr.1
      FragFun 2 1::First function of second atom
      5 1:: All pfunctions of fifth atom
   EndSum
End
```

9.1.2 Overlap populations

OverlapPopulations (block-type) *Overlap population weighted* DOS (**OPWDOS**), also known as the crystal orbital overlap population (**COOP**), are generated for the overlap populations listed

```
OVERLAPPOPULATIONS
   Left
      { iat lq }
```

```
        { FragFun jat ifun }
        { Frag kat }
    Right
        ...
End
```

You can use this to get the OPWDOS of two functions, or, if you like, one bunch of functions with another bunch of functions. The key-block should consist of left-right pairs. After a line with left you enter lines that specify one or more functions (according to *GrossPopulations* (page 74)), followed by a similar structure beginning with right, which will produce the OPWDOS of the left functions with the right functions. Example:

```
OVERLAPPOPULATIONS
    LEFT::First OPWDOS
        Frag 1
    RIGHT
        Frag 2
    LEFT:: Next OPWDOS
        FragFun 1 1
    RIGHT
        2 1
        FragFun 3 5
End
```

9.2 Band structure

BAND can calculate the band structure for the standard k-path in the Brillouin zone [*65* (page 208)] and saves the corresponding data to the binary file RUNKF.

The band structure is best examined with the GUI module **BandStructure** (see BAND-GUI tutorial Getting Started with BAND). Options for the band structure can be specified in the `BZStruct` block:

```
BZStruct
    {Enabled      [True|False]}
    {Automatic    [True|False]}
    {FatBands     [True|False]}
    {UseSymmetry [True|False]}
    {DeltaK       value}
End
```

Enabled (**Default: False**) Whether or not to calculate the band structure.

Automatic (**Default: True**) If `True` BAND will automatically generate the standard path through the Brillouin zone. If `False` BAND will use the user-defined path in *BZPath* (page 76).

FatBands (**Default: True**) If `True` BAND will compute the fat bands (note: only if `BZStruct%Enabled=True`). The Fat Bands are the periodic equivalent of the Mulliken population analysis (https://en.wikipedia.org/wiki/Mulliken_population_analysis).

UseSymmetry (**Default: True**) If `True` only the irreducible wedge of the Wigner-Seitz cell is sampled. If `False`, the whole (inversion-unique) Wigner-Seitz cell is sampled. **Note:** The *Symmetry* (page 95) key does not influence the symmetry of the band structure sampling.

DeltaK (**Default: 0.1**) Step (in reciprocal space, unit: 1/Bohr) for band structure interpolation. Using a smaller number (*e.g.* 0.03) will result in *smoother* band curves at the cost of an increased computation time.

Information on the k-path used for band structure plotting (including the fractional coordinates of high-symmetry k-points) can be found in the section `KPath` of the output file.

9.2.1 User-defined path in the Brillouin zone

If `BZStruct%Automatic` is `False`, BAND will compute the band structure for the user-defined path in the `BZPath` block. You should define the vertices of your path in fractional coordinates (wrt the reciprocal lattice vectors) in the `Path` sub-block. If you want to make a *jump* in your path, you need to specify a new `Path` sub-block.

In the following example we define the path `Gamma-X-W-K|U-X` for a FCC lattice:

```
BZPath
   Path
      0.000    0.000    0.000
      0.500    0.000    0.500
      0.500    0.250    0.750
      0.375    0.375    0.750
   SubEnd
   Path
      0.625    0.250    0.625
      0.500    0.000    0.500
   SubEnd
End
```

9.2.2 Definition of the Fat Bands

The *fat bands* (page 75) $F_{i,n,\sigma,\vec{k}}$ are the periodic equivalent of the Mulliken population. They are defined as:

$$F_{i,n,\sigma,\vec{k}} = \sum_j C_{i,n,\sigma,\vec{k}} C_{j,n,\sigma,\vec{k}} S_{i,j,\vec{k}}$$

where $C_{i,n,\sigma,\vec{k}}$ and $S_{i,j,\vec{k}}$ are the orbital coefficients and the overlap matrix elements respectively. The indices i and j denote basis functions, n is the band index, σ is the spin index and $\vec{k}$ is a reciprocal vector in the Brillouin zone.

9.2.3 Band Gap

The band gap (if any) is printed in the output. Here is an example for the NaCl crystal:

```
----------------------------------------
Band gap information
----------------------------------------
Number of valence electrons                  16
Valence Band index                            8
Top of valence Band (a.u.)               -0.192
Bottom of conduction Band (a.u.)         -0.039
Band gap (a.u.)                           0.153
Band gap (eV)                             4.173
Band gap (kcal)                          96.235
```

9.3 Charges

BAND always prints a number of charge analyses. Namely the Voronoi charges, the Mulliken analysis, and the Hirshfeld charges. The calculation comes at virtually no cost, so there are no keys associated with it. This is different for the Bader analysis.

9.3.1 Bader Analysis (AIM)

GridBasedAIM (block-type) Invoke the ultra fast grid based Bader analysis.[*33* (page 207), *34* (page 207)]

```
GridBasedAIM
   {SmallDensity rhosmall }
   {Iterations n }
   {UseStartDensity [false | true] }
End
```

> **SmallDensity** (**Default: 1e-6**) Where *rhosmall* is the value below which the density is ignored. This should not be chosen too small because it may lead to unassignable grid points.

> **Iterations** (**Default: 40**) The number *n* determines the maximum number of steps that may be taken to find the nuclear attractor for a grid point.

> **UseStartDensity** (**Default: false**) handles whether the analysis is performed on the startup density (*true*) rather than on the final density (*false*).

AIMCriticalPoints (block-type) Also the critical points of the density can be determined. The algorithm starts from a regular mesh of points, and from each of these it walks towards its corresponding critical point.

```
AIMCriticalPoints
   { GridPadding pad }
   { GridSpacing space }
   { eqvPointsTol tol }
End
```

> **GridPadding** (**Default: 0.7** Bohr) *pad* determines how much extra space is added to the starting guess domain in the search for the critical points.

> **GridSpacing** (**Default: 0.5** Bohr) The variable *space* determines the distance between the initial trial points.

> **eqvPointsTol** (**Default: 0.27** Bohr) *tol* is used as a criterion to whether or not two critical points are the same.

Note: The Bader (AIM) analysis is performed on the fitted density (see *Density fitting (Zlm Fit)* (page 40)). We advise to use a Good (or better) ZlmFit quality.

9.4 Fragments

A fragment feature is available albeit rather primitive. It allows for the analysis of the DOS in a fragment basis and for the calculation of the deformation density with respect to fragment densities. A typical application is the periodical adsorption of one or more molecules on a surface. For instance, consider periodic adsorption of hydrogen molecules over a surface. First you calculate the free molecule in the same orientation as when adsorbed to the substrate. Since you would like to use a molecular fragment, it makes sense to put the molecules far apart (large lattice spacing) and force dispersion to be neglected (KSPACE 1). To use the fragment in the next run you need to rename the result file ("RUNKF"), to something like "frag.runkf", see the example script discussed below or the *example* (page 175) covering this topic.

9.4.1 Fragment key

Fragment (block-type) Defines a fragment. This key takes as argument the fragment file name (absolute path or path relative to the executing directory). You can define several fragments for a calculation.

```
Fragment filename
  indexFragAt indexCurrentAt
  {...}
  {Labels
    labelName1
    {...}
  Subend}
End
```

indexFragAt indexCurrentAt One has to associate the atoms of the fragment to the atoms of the current calculation. So, for each atom of the fragment the *indexFragAt* has to be associated uniquely to the *indexCurrentAt* for the current calculation.

Labels This subkey gives the possibility to introduce labels for the fragment orbitals.

Example:

```
Fragment
  1 3 !  atom 1 of this fragment is assigned to third atom
  2 4 !  atom 2 of this fragment is assigned to fourth atom
  Labels
    Sigma
    Sigma*
    Pi_x
    Pi_y
    Pi_x*
    Pi_y*
  Subend
End
```

In this example the first six fragment orbitals will be labeled as stated in the body of this key. The remaining orbitals are labeled by the default labeling system (e.g. 1/FO/5, etc.). The labels are used in combination with options like `Print Eigens` and `Print OrbPop`. (See also `Print OrbLabels`). This key can be given once for each fragment.

Tip: Specifying:

```
Print Eigens
```

for a calculation produces output concerning the eigen states, thereby providing a means to identify the eigen states (e.g. to be sigma, pi, et cetera). So, one can label the orbitals of a fragment according to this information.

9.5 Energy Decomposition Analysis

In BAND there are two fragment-based energy decomposition methods available: the periodic energy decomposition analysis (PEDA)[56 (page 208)] and the periodic energy decomposition analysis combined with the natural orbitals of chemical valency method (PEDA-NOCV)[56 (page 208)].

9.5.1 Periodic Energy Decomposition Analysis (PEDA)

```
PEDA
```

If present in combination with the `fragment` keyblocks the decomposition of the interaction energy between fragments is invoked and the resulting energy terms (ΔE_{int}, ΔE_{disp}, ΔE_{Pauli}, ΔE_{elstat}, ΔE_{orb}) presented in the output file. (See the *example* (page 181) or the tutorial)

> **Attention:** In case of the error message "Fragments cannot be assigned by a simple translation!", BAND does only allow for fragments which can be transformed to the structure in the PEDA calculation by a simple translation. So, a rotation is not allowed.

9.5.2 Periodic Energy Decomposition Analysis and natural orbitals of chemical valency (PEDA-NOCV)

`PEDANOCV (block-type)`

If present in combination with the `fragment` keyblocks and the `PEDA` key the decomposition of the orbital relaxation term is performed. The binary result file will contain the information to *plot NOCV Orbitals and NOCV deformation densities* (page 89). (See the *example* (page 184) or the tutorial)

```
PEDANOCV
    {EigValThresh rval}
End
```

EigValThresh (**Default: 0.001**) The threshold controls that for all NOCV deformation densities with NOCV eigenvalues larger than *rval* the energy contribution will be calculated and the respective pEDA-NOCV results will be printed in the output

9.6 3D field visualization with BAND

With ADFView you can visualize three-dimensional fields from the results of a BAND Calculation (**runkf** file).

Following is a list of relevant fields, with a short explanation and illustrative pictures (from a simple non-periodic H_2 calculation). All fields are in atomic units (a.u.) (https://en.wikipedia.org/wiki/Atomic_units).

Total_Density (rho) The electronic density $\rho(r)$. The integral of the electronic density over the whole space (or, for periodic systems, over the unit cell) equals the total number of electrons (valence + core).
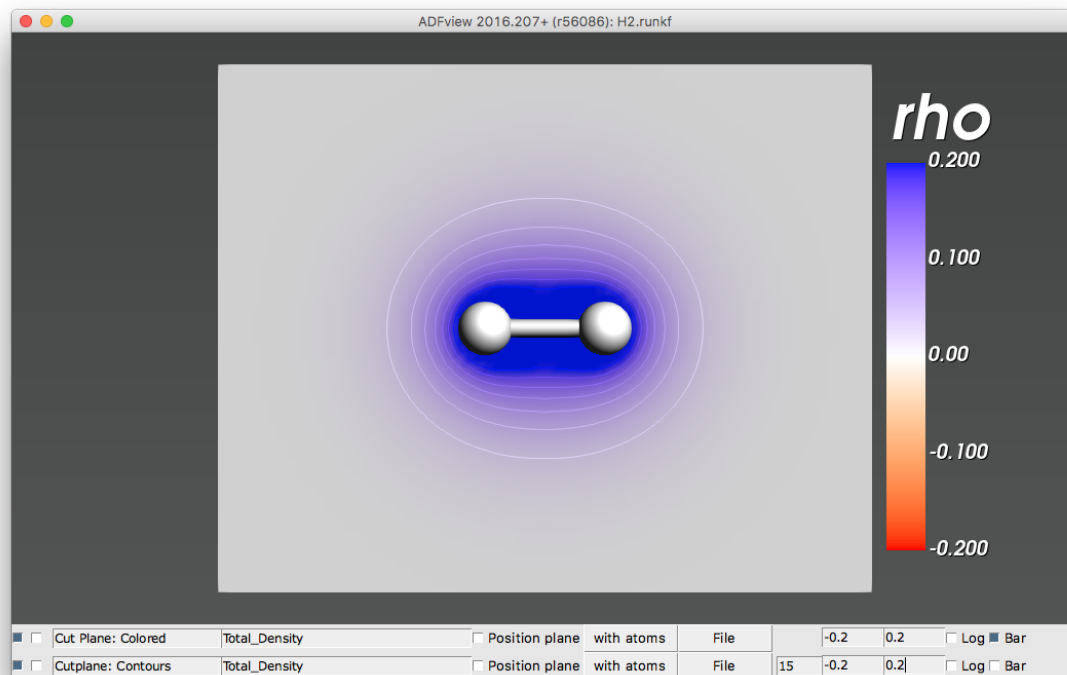
Deformation density (rho(deformation)) The deformation density is the difference between total density $\rho(r)$ and reference density $\rho_{\text{reference}}(r)$

$$\rho_{\text{deformation}}(r) = \rho(r) - \rho_{\text{reference}}(r)$$

The reference density $\rho_{\text{reference}}(r)$ is defined as the sum of densities of spherical spin-unrestricted isolated atoms.

The deformation density is electrically neutral, i.e. its integral over the whole space (or, for periodic systems, over the unit cell) is zero. Positive values of deformation density indicate density accumulation wrt isolated atoms; negative values represent density depletion. In our H_2 example, the deformation density shows how there is electron accumulation in the bonding region between the two hydrogen atoms.
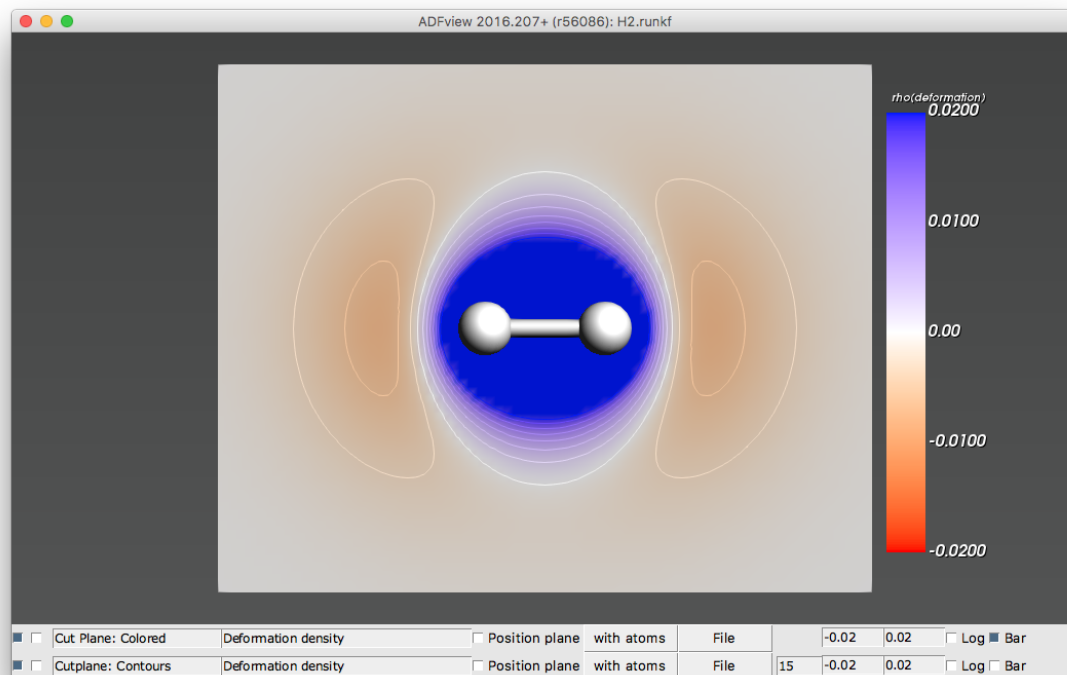
SCF_CoulombPot (v(coulomb)) The total Coulomb potential (nuclear + electronic potentials). BANDs convention for the Coulomb potential: the potential of positive charges (like nuclei) is **negative**, while the potential of negative charges (like electrons) is **positive**. In our example, the nuclear potential (negative) is larger than the electronic potential (positive) in the region of space near the H_2 molecule.
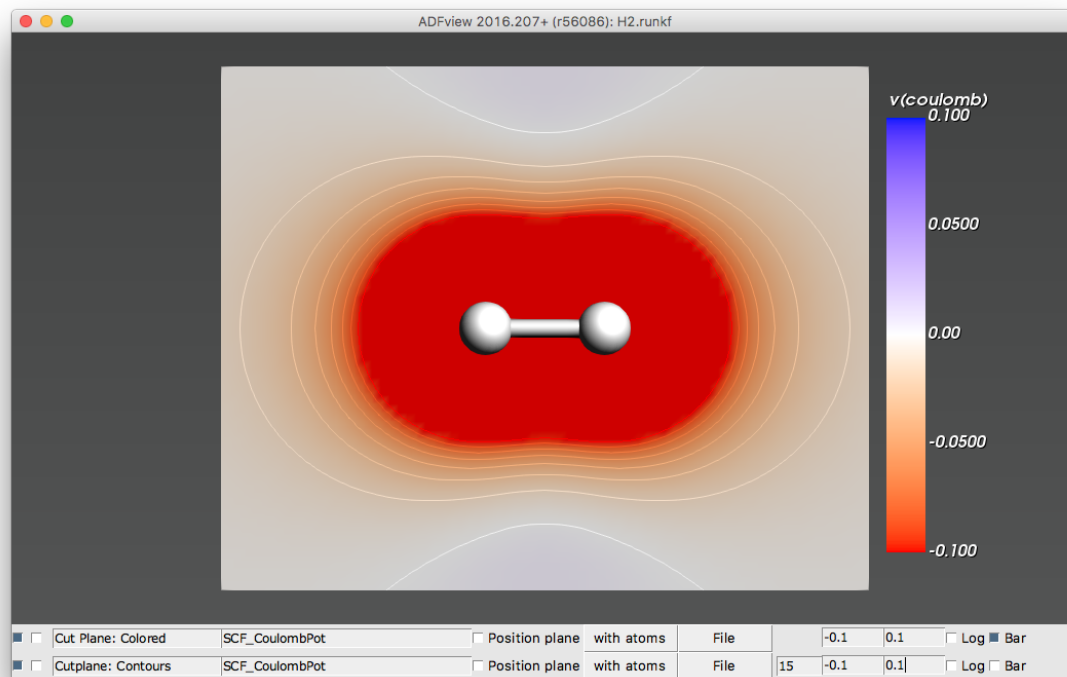
Note: The sign convention for potentials in **BAND** is the opposite to the **ADF** sign convention.

SCF_CoulombDeformation (v(coulomb/deformation)) The Coulomb potential originating from the (overall neutral) deformation density.
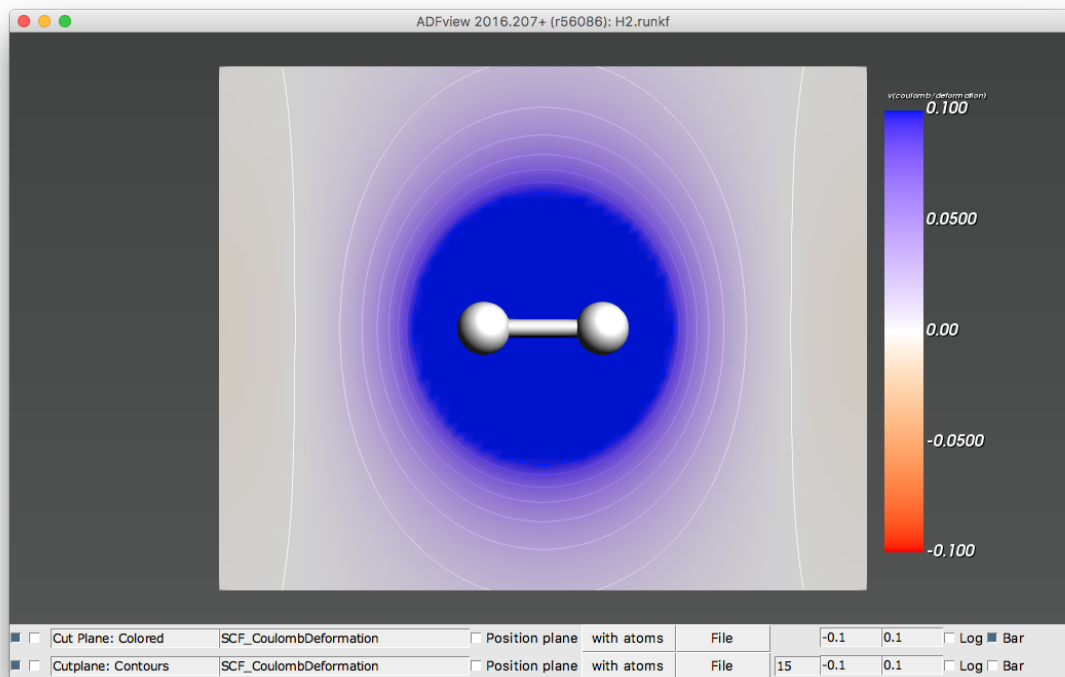
xc pot (vxc(rho)) The Exchange Correlation (XC) potential. Electrons are *attracted* by negative XC potentials (just like they are *attracted* by the negative nuclear Coulomb potential)
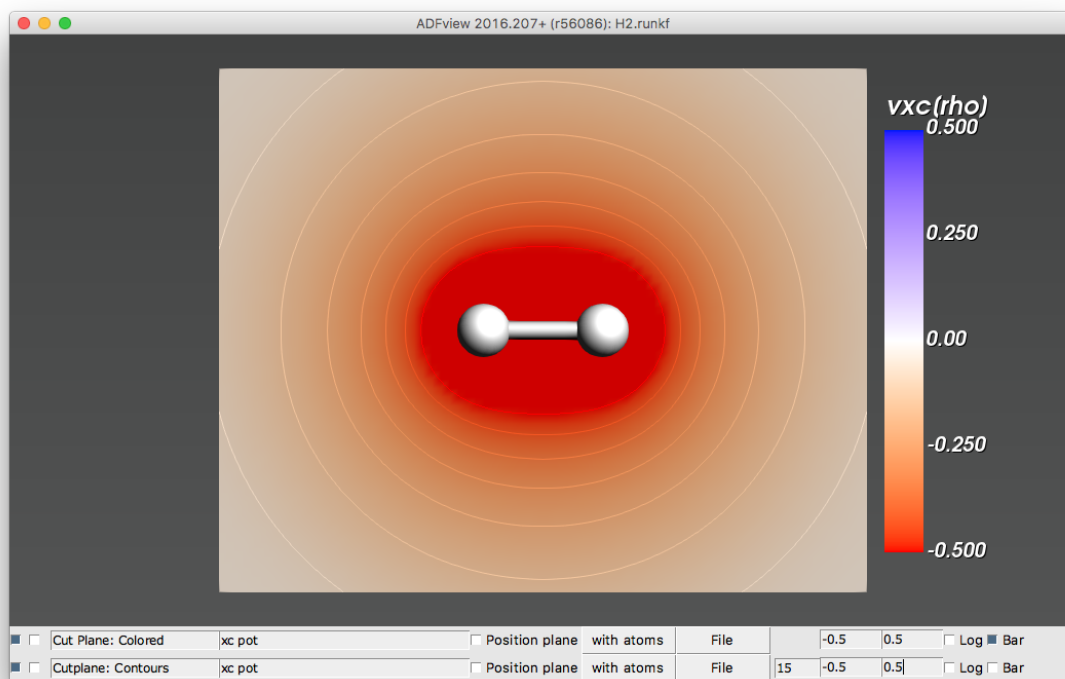
Orbitals (occupied/virtual) The Kohn-Sham orbitals.

Note: Be aware that there is an over-all arbitrariness in the sign of the orbitals

---

Here we show the occupied and first virtual orbital of H_2.
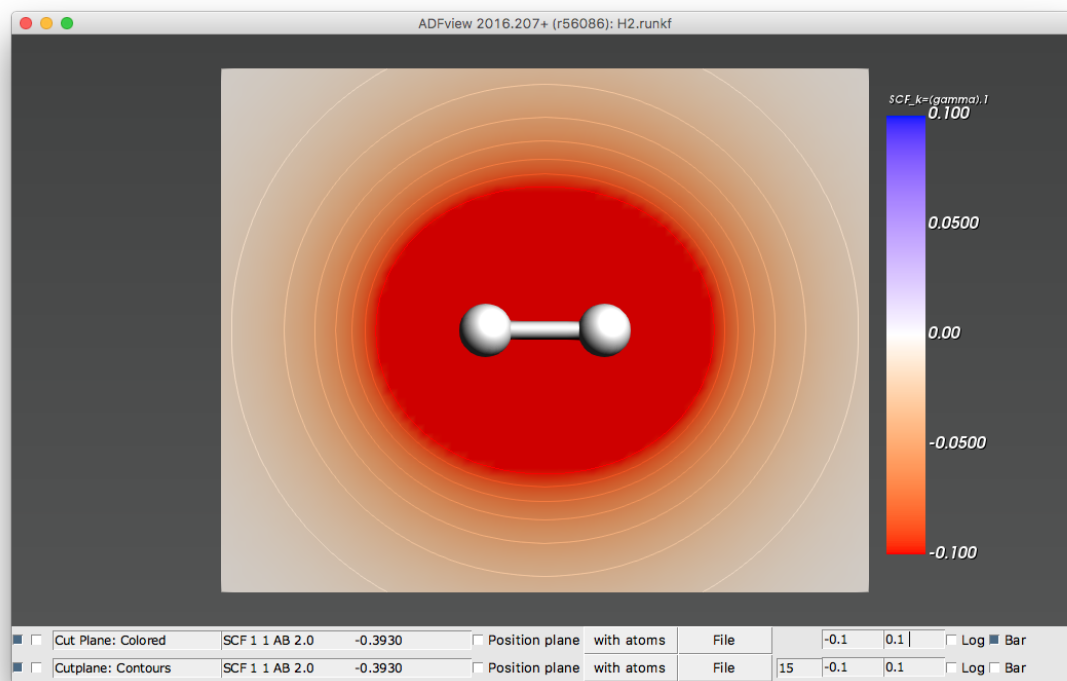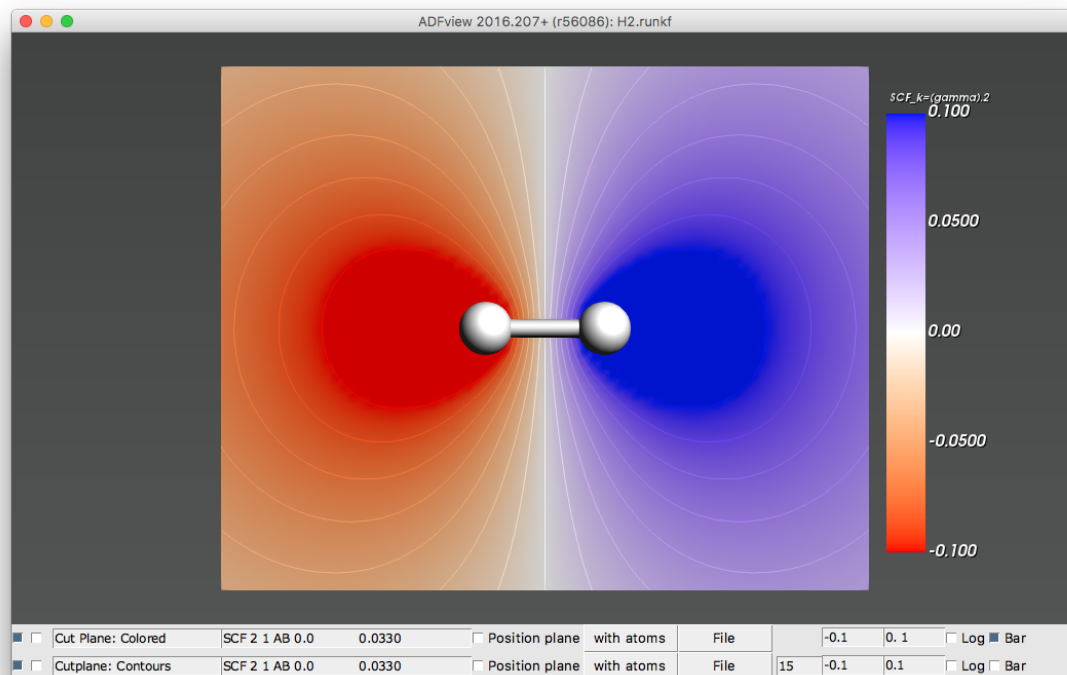
RESTARTS

The main results of a BAND calculation are stored in the RUNKF file. If you save this file you can use it to restart your calculation. The input for the restart calculation is essentially the same, except for some extra keys, like `Restart`, `Grid`, and `DensityPlot`.

Plots of the density (and many other symmetric properties) can can be obtained with the key `DensityPlot`. Density and orbital plot restarts require the specification of the `Grid` key. With the subkey `SCF` you can start the SCF procedure with the last solution from the restart file. This can be useful if the SCF did not converge or if you want to compute some post-SCF properties (e.g. the *DOS* (page 73) or the *band structure* (page 75)). Similarly, a geometry optimization can be restarted with the subkey `GeometryOptimization` You can use the geometry of a previous calculation.

Some examples are available in the `$ADFHOME/examples/band` directory and are discussed in the Examples section.

10.1 Restart key

Restart Tells the program that it should restart with the restart file.

```
Restart
   File filename
   {option}
   UseDensityMatrix True/False
End
```

File *filename* is the name of the restart file

UseDensityMatrix If True, SCF the density matrix will be used. Requires you to set the "*Save* (page 93) DensityMatrix" in the previous run.

options is the program part to do a restart for:

- SCF (from orbitals or density matrix), see See example *RestartSCF* (page 128).

- GeometryOptimization

- Geometry

- *OrbitalPlot* (page 89)

- *DensityPlot* (page 88)

- *NOCVdRhoPlot* (page 90)

- *NOCVOrbitalPlot* (page 89)

Usually the input for a restart is the same as for the original calculation, where you add some extra options.

10.2 Grid

Grid (block-type) Used for restart options `OrbitalPlot`, `DensityPlot`, `NOCVOrbitalPlot` and `NOCVdRhoPlot`. There are two ways to define your grid. The most easy way is to use the Type key which can have the values **Coarse**, **Medium**, and **Fine**, for example

```
Grid
   Type Coarse
End
```

This generates automatically a grid around the atoms in the unit cell. The alternative is to specify everything by hand. The following input would create a cube from (-1,-1,-1) to (1,1,1)

```
Grid
   -1 -1 -1 ! Starting point
   1 0 0 0.1   ! vec1 and dvec1
   0 1 0 0.1   ! vec2 and dvec2
   0 0 1 0.1   ! vec3 and dvec3
   20 20 20    ! nr. of steps along three directions
End
```

10.3 Plots of the density, potential, and many more properties

DensityPlot (block-type) Goes together with the `Restart%DensityPlot` and `Grid` keys. Example input

```
...
Restart vi
   File my.runkf
   DensityPlot
End

Grid
   Type Coarse
End

DensityPlot
   rho(fit)
   vxc[rho]
End
...
```

After such a run you get a TAPE41 file that you should rename to my.t41, and view with ADFview.

The most common properties to plot are:

- `rho(fit)` The fitted density.
- `v(coulomb)` The Coulomb potential.
- `vxc[rho(fit)]` the XC potential (using the fitted density)

- `vxc[rho]` XC potential of the exact density

- `rho` The density

- `|gradRho|` The norm of the gradient of the density

- `tau` The symmetric kinetic energy density

- `LDOS` The local density of states. (See *LDOS key* (page 90))

- `elf[rho]` The electron localization function

Some more specialized options are:

- `rho(deformation/fit)` the fitted deformation density

- `rho(atoms)` The density of the startup atoms

- `v(coulomb/atoms)` The Coulomb potential of the start density

- `s[rho]` Reduced density gradient. Common ingredient for XC functionals

- `s[rho(fit)]` Same as above, now for the fit density

- `alpha[rho]` Ingredient for some meta-GGAs

In the BAND example directory there is the *Frags_COCu* (page 175) example which shows how this can be used in combination with the `Fragment` key.

10.4 Orbital plots

OrbitalPlot (block-type) Goes together with the `Restart%OrbitalPlot` and `Grid` keys. In the BAND example directory there is the *Cu_slab* (page 189) example that shows how this can be used. Example input

```
...
Restart
   File my.runkf
   OrbitalPlot
End

Grid
   Type Coarse
End

OrbitalPlot
   1 Band 5 8  ! for k-point 1 plot bands 5 to 8
   5 Band 6    ! for k-point 5 plot band 6
   6 -0.2 +0.3 ! for k-point 6 plot bands between -0.2 and +0.3 a.u. w.r.t Fermi
↪level
End
...
```

After such a run you get a TAPE41 file that you should rename to my.t41, and view with ADFview.

10.5 NOCV Orbital Plots

NOCVOrbitalPlot (block-type) Goes together with the `Restart%NOCVOrbitalPlot` and `Grid` keys. See example *PEDANOCV_MgO+CO* (page 184). Example input

```
...
Restart
    File my.runkf
    NOCVOrbitalPlot
End

Grid
    Type Coarse
End

NOCVOrbitalPlot
    1 Band 5 8 ! for k-point 1 plot NOCV Orbitals 5 to 8
End
...
```

After such a run you get a TAPE41 file that you should rename to my.t41, and view with ADFview.

10.6 NOCV Deformation Density Plots

NOCVdRhoPlot (block-type) Goes together with the `Restart%NOCVdRhoPlot` and `Grid` keys. See example *PEDANOCV_MgO+CO* (page 184). Example input

```
...
Restart
    File my.runkf
    NOCVdRhoPlot
End

Grid
    Type Coarse
End

NOCVdRhoPlot
    1 Band 5 8 ! for k-point 1 plot NOCV deformation densities 5 to 8
End
...
```

After such a run you get a TAPE41 file that you should rename to my.t41, and view with ADFview.

10.7 LDOS (STM)

LDOS (block-type) Local Density-Of-States information. This can be used to generate STM images in the Tersoff-Hamann approximation.[*35* (page 207)]

```
LDOS
    { Shift eshift }
    { DeltaNeg deneg }
    { DeltaPos depos }
End
```

Shift (**Default: 0.0** a.u.) The energy bias with respect to the fermi level in a.u..

DeltaNeg (**Default: 1e-4** a.u.) defines the lower bound energy as *eshift-deneg*.

DeltaPos (**Default: 1e-4** a.u.) defines the upper bound energy as *eshift+depos*.

The local density of states is integrated over the resulting interval. Example of an LDOS restart

```
Restart
    File my.runkf
    DensityPlot
End

Grid
    Type Coarse
End

DensityPlot
    LDOS
    Shift       0.1
    DeltaNeg .001
    DeltaPos 0
End
```

According to this example, we restart from the result file of a previous calculation. The calculation will generate a file TAPE41 which can be viewed with ADFview. (Rename the file to my.t41)

See also *Restart* (page 87), and *DensityPlot* (page 88).

10.8 Complete example scripts for visualization

Below follows a complete example how to use BAND for visualizing. We start with the normal calculation for e.g. a Li slab

```sh
#!/bin/sh
# Contents of the file LiSlab.job

"$ADFBIN/band" << eor
TITLE Li Slab

UNITS
    length Angstrom
    angle Degree
END

ATOMS Li
   Li  0.0    0.0    0.0
   Li -1.745 -1.745 -1.745
END

Lattice
   3.49 0.000000 0
   0.000000 3.49 0
End

BasisDefaults
   BasisType DZ
   Core Large
End

XC
```

```
   LDA SCF VWN
END


end input
eor


# store the result file
mv RUNKF LiSlab.runkf
```

Note that we store the result file in "LiSlab.runkf". In the next run we use this file to generate the plot file. The job is essentially a copy of the first job with some extra lines. In this case we instruct the program to generate plot information for the density and the LDOS

```
#!/bin/sh
# Contents of the file LiSlab_restart.job

"$ADFBIN/band" << eor
TITLE Li Slab

UNITS
    length Angstrom
    angle Degree
END

Restart
   File LiSlab.runkf
   DensityPlot
End

Grid
   Type Coarse
End

DensityPlot
   rho(fit)
   LDOS
End

LDOS
   shift 0.0
End

ATOMS
   Li   0.0    0.0     0.0
   Li  -1.745 -1.745 -1.745
END

Lattice
   3.49 0.000000 0
   0.000000 3.49 0
End

BasisDefaults
   BasisType DZ
   Core Large
End

XC
```

```
   LDA SCF VWN
END

end input
eor

# Store the result file, and more importantly, the plot file (.t41)
mv RUNKF LiSlab_restart.runkf
mv TAPE41 LiSlab.t41
```

And then you can visualize the result with ADFview.

```
$ADFBIN/ADFview  LiSlab.runkf
```

It is most convenient to start ADFview from the .runkf file rather than from the .t41.

10.9 Save

With "Save -something-" directives you can tell BAND to save extra information.

`TAPE10` One option is "Save TAPE10", which will cause the file with the grid to be saved.

`DensityMatrix` The "Save DensityMatrix" command causes the density matrix to be written to the RUNKF file, needed with `Restart%UseDensityMatrix true`.

EXPERT OPTIONS

11.1 Symmetry

The symmetry of the system is automatically detected. Normally the symmetry of the initial system is maintained. One can lower the symmetry with the `Symmetry` key. In such cases the keyword `POTENTIALNOISE` can force the solution away from the initial symmetry.

SYMMETRY The most common option is **NOSYM** forcing the program to run without any symmetry. It is also possible to run in a lower symmetry other than NOSYM but this is more involved. The argument should be a list of numbers, representing the operators to maintain. The way to proceed is as follows. First run the calculation with the following added to your input

```
print symmetry
stopafter gemtry
```

and then you look in the output for (here the first four operators are listed)

```
   64    SYMMETRY OPERATORS:

NO              MATRIX              TRANSL       AXIS     DET   ROTATION
-----------------------------------------------------------------------

1)      1.000   0.000   0.000      0.000      0.000    1.0      1
        0.000   1.000   0.000      0.000      0.000
        0.000   0.000   1.000      0.000      1.000

2)      1.000   0.000   0.000      0.000      0.000    1.0      1
        0.000   1.000   0.000      5.400      0.000
        0.000   0.000   1.000      0.000      1.000

3)      1.000   0.000   0.000      5.400      0.000    1.0      1
        0.000   1.000   0.000      0.000      0.000
        0.000   0.000   1.000      0.000      1.000

4)      1.000   0.000   0.000      5.400      0.000    1.0      1
        0.000   1.000   0.000      5.400      0.000
        0.000   0.000   1.000      0.000      1.000
```

from this list you should select the desired operators and use that in your final calculation, for example

```
Symmetry 1 7 21 31
```

POTENTIALNOISE The initial potential for the SCF procedure is constructed from a sum-of-atoms density. Added to this is some small noise in the numerical values of the potential in the points of the integration grid. The

purpose of the noise is to help the program break the initial symmetry, if that would lower the energy, by effectively inducing small differences between (initially) degenerate orbitals.

```
PotentialNoise rval
```

The noise in the potential is randomly generated between zero and an upper limit *rval* (**Default: 1e-4** a.u.). This can be used therefore to suppress the noise by choosing zero, or to increase it by specifying some large number.

11.2 Excited States

By default the levels are occupied according to the aufbau principle. In some cases it is possible to create holes below the Fermi level or uneven occupation for up and down spin with the `Occupations` (Γ-only) and alternatively the `EnforcedSpinPolarization` (for an arbitrary number of k-points) key.

OCCUPATIONS (block-type) Allows one to input specific occupations numbers. Applies only for calculations that use only one k-point (i.e. pseudo-molecule calculations).

```
OCCUPATIONS
    1 occupations_alpha {// occupations_beta}
End
```

- `occupations_beta` and the separating double slash (//) must not be used in a spin-restricted calculation.
- `occupations_alpha/beta` is a sequence of values assigned to the states ('bands') in energy ordering.

If you want to enforce a specific spin-polarization (instead of occupying according to the aufbau principle) you can use the `EnforcedSpinPolarization` key:

```
EnforcedSpinPolarization rval
```

EnforcedSpinPolarization Allows to specify the spin polarization, i.e. the difference between the number of alpha and beta electron. e.g. `EnforcedSpinPolarization 1` means that there is one more alpha electron than beta electrons.

ElectronHole (block-type) Allows one to specify an occupied band which shall be depopulated, where the electrons are then moved to the Fermi level. For a spin-restricted calculation 2 electrons are shifted and for a spin-unrestricted calculation only one electron is shifted.

```
ElectronHole
  BandIndex ival1
  SpinIndex ival2
End
```

BandIndex *ival1* defines which occupied band shall be depopulated.

SpinIndex *ival2* defines the spin of the shifted electron. (1 or 2)

General Remark: To accomplish the excitation of a specific low-lying AO of one atom it helps to define for all other atoms a basis set with frozen core and for the chosen one an all-electron basis set. Then the lowest bands are equal to the AOs of the chosen atom. (See the example *Si_ElectronHole* (page 197))

11.3 The programmer key

The programmer has many options, most of which are set automatically. However, with the programmer key you can override many default behaviors. An overview is printed early in the output, and looks like

```
====================
R U N   C O N F I G
====================


Calculate gradient of basis  .   .   .   .   .   .   .   .   .   .   .   T
Calculate sec. der. of basis .   .   .   .   .   .   .   .   .   .   .   F
Calculate operator T working on basis .   .   .   .   .   .   .   .   T
Calculate gradient of fit .   .   .   .   .   .   .   .   .   .   .   .   T
Calculate sec. der. of fit.   .   .   .   .   .   .   .   .   .   .   .   F
Calculate gradient of fit pot.   .   .   .   .   .   .   .   .   .   .   T
Kin. energy via grad.   .   .   .   .   .   .   .   .   .   .   .   .   .   F
Calculate gradient of energy .   .   .   .   .   .   .   .   .   .   .   T
Calculate kin. energy density.   .   .   .   .   .   .   .   .   .   .   F
Calculate nuc. grad of kin. energy density  .   .   .   .   .   .   .   F
Calculate der. of density .   .   .   .   .   .   .   .   .   .   .   .   T
Calculate second der. of density.   .   .   .   .   .   .   .   .   .   F
Calculate nuc. grad. of density .   .   .   .   .   .   .   .   .   .   T
Calculate nuc. grad. of grad. density .   .   .   .   .   .   .   .   F
Exact rho during SCF.   .   .   .   .   .   .   .   .   .   .   .   .   .   T
Exact grad. rho during SCF.   .   .   .   .   .   .   .   .   .   .   .   F
Exact grad. rho post SCF  .   .   .   .   .   .   .   .   .   .   .   .   T
Store history of dens. matrix.   .   .   .   .   .   .   .   .   .   .   F
Store original Bloch functions  .   .   .   .   .   .   .   .   .   .   T
Direct  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   F
Direct basis  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   F
Calculate DOS .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   T
Calculate Band structure  .   .   .   .   .   .   .   .   .   .   .   .   T
Calculate EFG .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   F
```

Some of these option are controllable via the Programmer key.

Programmer (block-type)

> **UseZlmFit [true | false]** Determines whether or not to use ZlmFit. When using this the DirectBas key will be enabled.

> **FlioSinglePrecision [true | false]** The basis and fit functions are written and read to so-called FLIO files. With this option the double precision numbers can be converted to single precision, saving half of the IO. The results tend to be still fairly accurate. SCF convergence is usually possible to 1e-5. For deeper convergence disable this option.

> **DirectBas [true | false]** Normally basis functions (and possibly derivatives) are calculated once and stored on disk. Enabling this option causes BAND not to store the basis functions but to recalculate them when needed. This reduces the amount of IO, at the cost of increased CPU. Also disables FlioSinglePrecision.

> **ExactGradRho [true | false]** The gradient (and second derivative) of the density are usually obtained via the fitted density. With this option you force the program to use the exact density gradient. This requires the calculation of basis set derivatives.

> **KinViaGrad [true | false]** Matrix elements of the kinetic energy operator are normally calculated as

$$T_{ij} = \frac{1}{2} \int \Psi_i \nabla \cdot \nabla \Psi_j$$

Using partial integration this can be rewritten as

$$T_{ij} = -\frac{1}{2} \int (\nabla \Psi_i) \cdot \nabla \Psi_j$$

and requires the calculation of the gradient of the basis functions.

UseSharedMemory [true | false] Whether or not to use shared memory in a BAND calculation. Doing so might reduce the memory requirement. A reason to disable it might be a bug (like a hangup or race condition) due to this feature.

DidervCompat [true | false] Originally all radial derivatives of NAOs were done numerically. However, more accurate derivatives are available from the DIRAC subroutine. To get the old (and slightly worse) derivatives set this option to *true*.

RECOMMENDATIONS AND TROUBLESHOOTING

12.1 Recommendations

12.1.1 Model Hamiltonian

The most important ingredients are the kinetic energy (relativistic effects) and the XC functional.

Relativistic model

By default we do not use relativistic effects. The best approximation is to use spin-orbit coupling, however that is very expensive. The scalar relativistic option comes for free, and for light elements will give very similar results as non-relativistic theory, and for heavy ones better results w. r. t. experiment. We recommend to always use this (scalar ZORA). To go beyond to the spin-orbit level can be important when there are heavy elements with p valence electrons. Also the band gap appears quite sensitive for the spin-orbit effect.

XC functional

The default functional is the LDA, that gives quite good geometries but terrible bonding energies. GGA functionals are usually better at bonding energies, and among all possibilities the PBE is a common choice. Using a GGA is not a lot more expensive than using plain LDA. For the special problem of band gaps there are a number Model Hamiltonians available (eg. TB-mBJ and GLLC-SC). The `Unrestricted` option will be needed when the system is not closed shell. For systems interacting through dispersion interactions it is advised to use the Grimme corrections. Unfortunately there is no clear-cut answer to this problem, and one has to try in practice what works best.

12.1.2 Technical Precision

In principle it is simple to control the technical precision with the NumericalQuality key. Here is an example how you could tweak a bit more

```
NumericalQuality Normal ! this is the easy knob, sets the default quality for the␣
↪keys below

BeckeGrid
  Quality Basic ! tweak the grid
End
KSpace
  Quality Good  ! tweak the k-space grid
End
ZlmFit
```

```
  Quality Basic   ! tweak the density fit
End
SoftConfiment
  Quality VeryGood   ! tweak the radial confinement of basis functions
End
```

Here are per issue hints for when to go for a better quality (but it is by no means complete)

- **BeckeGrid**: Increase quality if there are geometry convergence problems. Also negative frequencies can be caused by the grid.

- **KSpace**: Increase quality for metals

- **ZlmFit**: Increase quality if the SCF does not converge.

- **SoftConfinment**: Increase quality for weakly bonded systems, such as layered materials

The other important issue is the basis set. For geometry optimization the TZP is advised. For properties TZ2P. The best available basis set is QZ4P. All these are quite big, and therefore the DZ basis set is an attractive basis set for quick scans. For geometry optimization the DZP basis works quite well, but it is available only for light elements, and defaults to the TZP basis for other elements.

12.1.3 Performance

The performance is influenced by the model Hamiltonian and basis set, discussed above. Here follow more technical tips.

Reduced precision

One of the simplest things to try is to run your job with NumericalQuality Basic. For many systems this will work well, and it can be used for instance to pre-optimize a geometry. However, it can also cause problems such as problematic SCF convergence, geometry optimization, or simply bad results. See above how to tweak more finely the *Technical Precision* (page 99).

Memory usage

Another issue that is the choice CPVector (say the vector length of you machine) and the number of k-points processed together during the calculation of the parameters. In the output you see the used value

```
                    ==========================
                    = Numerical Integration =
                    ==========================


TOTAL NR. OF POINTS                                  4738
BLOCK LENGTH                                          256
NR. OF BLOCKS                                          20
MAX. NR. OF SYMMETRY UNIQUE POINTS PER BLOCK           35
NR. OF K-POINTS PROCESSED TOGETHER IN BASPNT            5
NR. OF SYMMETRY OPERATORS (REAL SPACE)                 48
SYMMETRY OPERATORS IN K-SPACE                          48
```

If you want to change the default settings you can specify the `CPVector` and `KGRPX` keywords. The optimal combination depends on the calculation, on the machine. Example

```
CPVector 512
KGRPX 3
```

Note: bigger is not necessarily better.

Reduced basis set

When starting work on a large unit cell it is wise to start with a DZ basis. With such a basis, one can test for instance the quality of the k-space integration. However, for most properties, the DZ basis is probably not very accurate. You can next go for the DZP (if available) or TZP basis set, but that may be a bit of overkill.

Frozen core for 5d elements

The standard basis sets TZ2P are not optimal for third-row transition elements. The V basis offers the possibility to freeze the 4f,5s, and 5p functions. With this you can usually still get quite good results. Sometimes you need to relax the frozen core dependency criterion

```
Dependency Core=0.8 ! The frozen core overlap may not be exactly 1
```

If you want to use the TZ2P you can change it like this one for Au

```
AtomType AuDIRAC Au14 12 ! note: now 12 core functions, was 10
VALENCE
1S
2S
2P
3S
3P
3D
4S
4P
4D
4F
5S
5P
5D   ! from here on valence
6S  1
SubEnd
BasisFunctions
!  5S     4.600000
   6S     3.150000
   6S     1.210000
!  5P     6.650000
!  5P     3.100000
   5D     5.050000
   5D     1.500000
   6P     1.950000
   5F     2.500000
SubEnd
```

12.2 Troubleshooting

12.2.1 SCF does not converge

Some systems are more difficult to converge than others. A Pd slab for instance is easier to converge than an Fe slab. Generally, what you do in a problematic case is to go for more conservative settings. The two main option are to decrease `SCF%Mixing` and/or `DIIS%Dimix`.

```
SCF
   Mixing 0.05 ! more conservative mixing
End

Diis
   DiMix 0.1 ! also more conservative strategy for DIIS procedure
   Adaptable false ! disable automatic changing of dimix
End

Convergence
   Degenerate Default ! For most calculations this is quite a good idea anyway
End
```

An other option is to first run the system with a SZ basis, which may be easier to converge. Then you can *Restart* (page 87) the SCF with a larger basis set from this result.

Sometimes SCF convergence problems are caused by bad precision. An indication of this is when there are many iteration after the HALFWAY message. The simplest thing to try is to see whether increasing the `NumericalAccuracy` helps. Specifically an insufficient quality of the **density fit** may cause problems. For systems with heavy elements the quality of the **Becke grid** may also play a role. Another potential problem is using **only one k-point**. The following could be a typical example

```
NumericalQuality Basic ! this is an expensive calculation, like to keep it basic

KSpace
  Quality Normal ! add this when basic gives you only one k-point
End

ZlmFit
  Quality Normal   ! but we do a better density fit (you can also try Good)
End

BeckeGrid
  Quality Normal ! better grid especially for heavy atoms (you can also try Good)
End
```

An alternative is to try a **LIST** method. For sure the cost of a single SCF iteration will increase, but it may reduce the number of SCF cycles, see *Diis%Variant* (page 43).

```
Diis
  Variant LISTi ! invoke the LISTi method
End
```

For heavy elements the use of a small or no frozen core may complicate the SCF convergence.

12.2.2 Geometry does not converge

One thing that you should make sure is that at least the **SCF converges**. If that is so, then maybe the **gradients are not accurate enough**. Here are some settings to improve the accuracy of the gradients

```
RadialDefaults
  NR 10000   ! more radial points
End

NumericalQuality Good
```

12.2.3 Negative frequencies in phonon spectra

When doing a phonon calculation one sometimes encounter unphysical negative frequencies. There are two likely causes: either the **geometry was not in the minimum geometry** (see `GeoOpt%Converge`), or the **step size** used in the Phonon run is too large (see *PhononConfig%StepSize* (page 53)). Also **general accuracy** issues may be the cause, such as numerical integration, k-space integration and fit error.

12.2.4 Basis set dependency

A calculation aborts with the message: dependent basis. It means that for at least one k-point in the BZ the set of Bloch functions, constructed from the elementary basis functions is so close to linear dependency that the numerical accuracy of results is in danger. To check this, the program computes, for each k-point separately, the overlap matrix of the Bloch basis (normalized functions) and diagonalizes it. If the smallest eigenvalue is zero, the basis is linearly dependent. (Negative values should not occur at all!). Given the limited precision of numerical integrals and other aspects in the calculation, you are bound for trouble already if the smallest eigenvalue is very small, even if not exactly zero. The program compares it against a criterion that can be set in input (key `Dependency` option *Bas*).

If you encounter such an error abort, you are strongly advised not to adjust the criterion so as to pass the internal test: there were good reasons to implement the test and to set the default criterion at its current value. Rather, you should adjust your basis set. There are two ways out: using confinement or removing basis functions.

Using confinement

Usually the dependency problem is due to the diffuse basis functions. This is especially so for highly coordinated atoms. One way to reduce the range of the functions is to use the Confinement key. In a slab you could consider to use confinement only in the inner layers, and to use the normal basis to the surface layers. The idea is that basis functions of the surface atoms can describe the decay into the vacuum properly, and that inside the slab the diffuseness of the functions is not needed. If all the atoms of the slab are of the same type, you should make a special type for the inner layers: simply put them in a separate Atoms block. The confinement can be specified per type.

Removing basis functions

You can remove one or more basis functions and maybe modify some of the (other) STO basis functions. The program prints information that helps you determine which basis functions should be modified/removed. Another way to modify your basis set, is to use the confinement keyword. This has the effect of making the diffuse basis functions more localized, thus reducing problematically large overlap with similar functions on neighboring atoms.

In the standard output file, after the error message, you will find a list of eigenvalues of the overlap matrix. If only the first is smaller than the threshold, you should remove one basis function. If more eigenvalues are very small, it is likely that you have to remove more than one function, although you can of course try how far you can get by eliminating just one.

Next the program prints the so-called Dependency Coefficients: a list of numbers, one for each basis function. Those with a large value are the suspicious ones. If you find two coefficients that are significantly larger than the others, you should replace the two corresponding functions by one. Easiest is to remove one of them (take the one with the bigger coefficient). If one of them is a numerical orbital from Dirac and the other an STO, remove the STO. If both are STOs, remove one and replace the other by some kind of average (regarding the radial characteristic: exponential factor and power of radial coordinate).

To identify how the functions in your input correspond to the list the underlies the series of Dependency Coefficients, you have to set up the list of basis functions as follows:

- Consider an outer loop over all atom TYPES. These correspond, in order as well as in number, to the sequence of AtomType keys in your input file.

- For each type, consider a loop over all atoms of that type, i.e. the atoms in the ATOM block corresponding to the AtomType key at hand.

- For each atom (each AtomType key), first write down all DIRAC basis functions, then all STOs. When writing down the functions, be aware that each entry in your input file specifies a function *set*, by the quantum number L and hence corresponds to 2L+1 actual basis functions.

- Regarding the DIRAC basis functions: they belong to the list of basis functions only if the key Valence occurs in the pertaining DIRAC input block. If not, no DIRAC functions of that type are included in the basis. *If the Dirac functions are included, you must omit the Core functions and include only the Valence functions from that DIRAC block.* The first record in your DIRAC block with two numbers defines (by the first number) the total number of function sets in the DIRAC block (which you can verify by simple counting) and (by the second number) the number of Core function sets among them. The Core function sets, if any, are always the first so many in the list in the DIRAC block.

The program stops as soon as it encounters a dependency problem. This may happen for the first k-point. After you have adjusted the basis set following the above guidelines, you will have solved it. However, it may easily happen that the problem shows up again, but now for another (later) k-point, where other entries in the basis set may cause trouble. Do not think you have repaired the first problem incorrectly. Just repeat the procedure until you pass all k-points in the basis set construction without errors. Typically (as a last remark), although not necessarily, the first k-point may have a dependency problem from too many *s*-type functions, while other k-points may be more sensitive to the series of *p*-functions in your basis.

12.2.5 Frozen core too large

BAND calculates the overlap matrix of the core functions, and this should approximate the unit matrix. When the deviation is larger then the frozen-core overlap criterion the program stops. The default criterion (0.98) is fairly strict. The safest solution is to choose a smaller frozen core. For performance reasons, however, this may not be the preferred option. In practice you might still get reliable results by setting the criterion to 0.8, see the *Dependency* (page 45) keyword. For the *5d* transition metals, for instance, you can often freeze the 4f orbital, thus reducing the basis set considerably. We strongly advise you to compare these results to a calculation with a smaller core. Such tests can be performed with a smaller unit cell or with a lower quality for the KSPACE block key.

12.3 Various issues

12.3.1 Understanding the logfile

In practice you will look often at the logfile to see whether the calculation is going fine. Here is a logfile for a single point calculation.

```
10:04:09   INIT
10:04:09   BAND development version   RunTime: Sep30-2014 10:04:09   Nodes: 1   Procs: 8
10:04:09   Specify title in input
10:04:09   using 2014 defaults convention
10:04:09   All basis functions smoothly confined at radius:   7.0
10:04:10   RADIAL
10:04:10   POINTS
10:04:11   CELLS
10:04:11   NUMGRD
10:04:11   ELSTAT
10:04:11   ATMFNC
10:04:11   CalcAtomicProperties
10:04:11   PREPAREBAS
10:04:11   ------ K ..    6
10:04:12   PREPAREHAM
10:04:12   ------ K ..    6
10:04:12   PREPAREFIT
10:04:13   start of SCF loop
10:04:13   initial density from psi
10:04:14   cyc=  0 err=0.00E+00 cpu=   1s ela=   2s
10:04:16   cyc=  1 err=7.83E-01 meth=m nvec= 1 mix=0.0750 cpu=   1s ela=   2s fit=5.
→02E-02
10:04:17   cyc=  2 err=7.14E-01 meth=d nvec= 2 mix=0.2000 cpu=   1s ela=   1s fit=2.
→12E-02
10:04:19   cyc=  3 err=9.39E-02 meth=d nvec= 3 mix=0.2000 cpu=   1s ela=   2s fit=2.
→72E-02
10:04:20   cyc=  4 err=2.82E-02 meth=d nvec= 3 mix=0.2200 cpu=   1s ela=   1s fit=2.
→92E-02
10:04:22   cyc=  5 err=4.48E-03 meth=d nvec= 3 mix=0.2420 cpu=   1s ela=   2s fit=2.
→92E-02
10:04:23   HALFWAY
10:04:23   cyc=  6 err=3.07E-03 meth=d nvec= 4 mix=0.2420 cpu=   1s ela=   1s fit=2.
→93E-02
10:04:25   cyc=  7 err=6.90E-04 meth=d nvec= 4 mix=0.2662 cpu=   1s ela=   1s fit=2.
→93E-02
10:04:26   cyc=  8 err=2.68E-04 meth=d nvec= 4 mix=0.2928 cpu=   1s ela=   2s fit=2.
→93E-02
10:04:28   cyc=  9 err=1.28E-04 meth=d nvec= 4 mix=0.3221 cpu=   1s ela=   1s fit=2.
→93E-02
10:04:29   cyc= 10 err=6.90E-05 meth=d nvec= 4 mix=0.3543 cpu=   1s ela=   1s fit=2.
→93E-02
10:04:30   SCF CONVERGENCE
10:04:31   cyc= 11 err=3.43E-06 meth=d nvec= 5 mix=0.3543 cpu=   1s ela=   1s fit=2.
→93E-02
10:04:32   cyc= 12 err=3.43E-06 meth=d nvec= 1 mix=1.0000 cpu=   1s ela=   1s fit=2.
→93E-02
10:04:32   ------ K ..    6
10:04:34
10:04:34   Max. cycle time CP:      1.379
10:04:34                 IO:         0.309
10:04:34
10:04:34   Mean cycle time CP:      1.459
10:04:34                 IO:         0.151
10:04:34                 EL:         1.636
10:04:34
10:04:34   final mix.par.           0.083
10:04:34   Approx. conv.rate:       0.000
10:04:34
```

```
10:04:34  FERMI ENERGY:          -0.2198 A.U.
10:04:34                         -5.9802 E.V
10:04:34  Band gap:               0.1394 A.U.
10:04:34                          3.7943 E.V
10:04:34   ENERGY
10:04:34  ENERGY OF FORMATION:   -1.3496 A.U.
10:04:34                        -36.7236 E.V.
10:04:34                       -846.8654 KCAL/MOL
10:04:34   CHARGE
10:04:34   HIRSH
10:04:34   CM5CHARGES
10:04:34   GRADIENTS
10:04:38   DOS
10:04:38  cannot use good scaling dos routines
10:04:38  storing all partial DOS
10:04:38  integrate over delta E
10:04:39   BZSTRUCT
10:04:39   PREPAREBAS
10:04:39  ------ K ..    6
10:04:39   PREPAREHAM
10:04:39  ------ K ..    6
10:04:41  copy T(V/VOC)
10:04:41  copy eigensystem
10:04:41  NORMAL TERMINATION
10:04:41  END
```

There are three different phases. The first phase is the preparation phase. The second phase is the SCF procedure. The third part is the properties phase. Particularly important are the SCF CONVERGENCE and NORMAL TERMINATION messages.

Let us take a closer look at a line during the SCF.

```
10:04:19  cyc=  3 err=9.39E-02 meth=d nvec= 3 mix=0.2000 cpu=   1s ela=   2s fit=2.
↪72E-02
```

The meaning of cyc is the iteration number, so it is the third iteration. The self consistent error (err) is 9.4e-2. The method (meth) to guess the density for the next cycle is d, meaning DIIS, being a linear combination (nvec) of three vectors. The density is biased (mix) by 0.2 towards output densities. The SCF cycle took 1 second of cpu time (per core), and needed 2 seconds of real time. Finally the error of the density fitting was 2.7e-2.

12.3.2 Breaking the symmetry

In some cases you want to break the symmetry. An example of this is when you want to get the antiferromagnetic state of Fe. Another common example is when you want to apply geometry constraints on atoms.

The easiest way to do this is of course to disable all symmetry, see *SYMMETRY%NOSYM* (page 95), but this might make your calculation more expensive than is needed. A bit more elegant way is to define separate types for the equivalent atoms. Here follows an example input for antiferromagnetic iron

```
! The two iron atoms have different "types" to break the symmetry
ATOMS
  Fe.a   0.0    0.0    0.0
  Fe.b  -1.435 -1.435  1.435
end

Lattice
   -1.435  1.435  1.435
```

```
    1.435 -1.435  1.435
    2.87   2.87  -2.87
End

CONVERGENCE
   CRITERION 1.0e-4
   Degenerate default
   SpinFlip 2  ! Flip (startup) spin density at second atom
END
```

Another solution is to use the expert SYMMETRY keyword.

12.3.3 Labels for the basis functions

You see the labels for the basis functions in for instance the DOS section of the output. The labels are also used in combination with options like `Print Eigens` and `Print OrbPop`. (See also *Print* (page 9)).

What do the labels look like? A normal atomic basis function, i.e. a numerical orbital or a Slater type orbital, gets a label like <atom number>/<element>/<orbital type>/<quantum numbers description>/<exp in sto>

Example with a Li and a H atom:

```
1/LI/NO/1s
1/LI/NO/2s
1/LI/STO/2s/1.4
1/LI/STO/2p_y/1.3
1/LI/STO/2p_z/1.3
1/LI/STO/2p_x/1.3
2/H/NO/1s
2/H/STO/1s/1.9
...
```

Core states will just get simple numbers as labels:

```
CORE STATE 1
CORE STATE 2
```

With the `Fragment` key you can give meaningful names to the fragment option, see `Fragment%Labels` and `DosBas`.

12.3.4 Reference and Startup Atoms

The formation energy of the crystal is calculated with respect to the reference atoms. BAND gives you the formation energy with respect to the spherically symmetric spin-*restricted* LDA atoms. If you want the program to do the spin-unrestricted calculation for the atoms you can give key Unrestricted the extra option Reference. We do not recommend this as it would give you the false (except in special cases) feeling that you've applied the right atomic correction energy so as to obtain the 'true' bonding energy with respect to isolated atoms. The true atomic correction energy is the difference in energy between the used artificial object, i.e. the spherically symmetric, spin-restricted atom with possibly fractional occupation numbers, and the appropriate multiplet state. The spin-*un*restricted reference atom would still be spherically symmetric, with possibly fractional occupations: it would only have the probably correct (Hund's rule) net spin polarization.

The startup density is normally the sum of the restricted atoms. In case you do an unrestricted calculation you may want to get the sum of the unrestricted atoms as startup density by giving key Unrestricted the extra option StartUp. This does not always provide a better startup density since all atoms will have their net-spins pointing up. If a frozen

core is used this option can sometimes lead to a negative valence density, because the frozen core is derived from the restricted atom. The program will stop in such a case.

No matter what reference or startup atoms you use, core orbitals and NOs originate always from the restricted free-atom calculation, because we don't want a spatial dependence of the *basis functions* on spin.

12.3.5 Numerical Atoms, Basis functions, and Fit functions

The program starts with a calculation of the free atoms, assuming spherical symmetry. The formation energy is calculated w.r.t such atoms. You have to specify the configuration (i.e. which orbitals are occupied) in the Dirac subkey of the block key AtomType, and you can for instance use the experimental configuration. Keep in mind, however, that this is not necessarily the optimal configuration for your density functional. For instance, Ni has experimentally two electrons in the 4s shell, but with LDA you will find that it is energetically more profitable to move one electron from the 4s to the 3d. The configuration of the reference atoms does not (i.e. should not) affect the final (SCF) density.

Besides the available basis sets in $ADFHOME/atomicdata/band, you could in principle use the basis functions from the database of the molecular ADF program (see the documentation of ADF for how this database is organized). The functions you will find there are STOs, which is not optimal since BAND offers you the option to use NOs from the numerical atom. The most efficient approach is to use the NOs and remove from the ADF basis set those STOs that are already well described by the NOs.

As an example we will construct a basis for the Ni atom with orbitals frozen up to the *2p* shell, derived from a triple-zeta ADF basis. In the Dirac subkey of the block key AtomType you specify that the NOs up to *2p* should be kept frozen and that the 3d and *4s* NOs be included in the valence basis. Copy from the ADF database all *3d*, *4s* and the polarization functions into the BasisFunctions subkey of the block key AtomType and remove the middle STOs of the *3d* and the *4s*.

Usually it is already quite adequate for a good-quality basis to augment each NO with one STO. You could then take a double zeta ADF basis and remove one of the 3d and one of the 4s STOs. We often find that such a basis, with one STO added per NO, has a quality that is comparable to *triple* zeta STO sets. We strongly recommend that you use combined NO/STO bases. Of course, you may want to verify the quality of the basis set by calculations on a few simple systems.

You can copy the fit functions from the ADF database into the FitFunctions subkey of the block key AtomType. As a matter of experience (and justified by a somewhat different handling of fit functions between the two programs), BAND is in most cases (we have not yet seen an exception) less sensitive to the quality of the fit set than ADF is.

EXAMPLES

13.1 Introduction

The ADF package contains a series of sample runs for the BAND program. Provided are UNIX scripts to run the calculations and the resulting output files.

The examples serve:

- To check that the program has been installed correctly: run the sample inputs and compare the results with the provided outputs. *Read the remarks below about such comparisons.*

- To demonstrate how to do calculations: an illustration to the User manuals. The number of options available in BAND is substantial and the sample runs do not cover all of them. They should be sufficient, however, to get a feeling for how to explore the possibilities.

- To work out special applications that do not fit well in the User's Guide.

Where references are made to the operating system (OS) and to the file system on your computer the terminology of UNIX type OSs is used.

All sample files are stored in subdirectories under $ADFHOME/examples/, where $ADFHOME is the main directory of the ADF package. The main subdirectory for the BAND examples is $ADFHOME/examples/band. Each sample run has its own directory. For instance, $ADFHOME/examples/band/NaCl/ contains a BAND calculation on the NaCl bulk crystal. Each sample subdirectory contains:

- A file TestName.run: the UNIX script to execute the calculation or sequence of calculations of the example

- A file TestName_orig.out: the resulting output(s) against which you can compare the outcome of your own calculation.

Notes:

- Running the examples on Windows: You can run an example calculation by double-clicking on the appropriate .run file. After the calculation has finished, you can compare the TestName.out file with the reference TestName_orig.out file. See remarks about comparing output files below.

- The UNIX scripts make use of the *rm* (remove) command. Some UNIX users may have aliased the *rm* command. They should accordingly adapt these commands in the sample scripts so as to make sure that the scripts will remove the files. New users may get stuck initially because of files that are lingering around after an earlier attempt to run one of the examples. In a subsequent run, when the program tries to open a similar (temporary or result) file again, an error may occur if such a file already exists. Always make sure that no files are left in the run-directory except those that are required specifically.

- It is a good idea to run each example in a separate directory that contains no other important files.

- The run-scripts use the environment variables ADFBIN and ADFRESOURCES. They stand respectively for the directory that contains the program executables and the main directory of the database. To use the scripts as they are you must have defined the variables ADFBIN and ADFRESOURCES in your environment. If a parallel

(PVM or MPI) version has been installed, it is preferable to have also the environment variable NSCM. This defines the default number of parallel processes that the program will try to use. Consult the Installation Manual for details.

- As you will note the sample run scripts refer to the programs by names like 'adf', 'band', and so on. When you inspect your $ADFBIN directory, however, you may find that the program executables have names 'adf.exe', 'band.exe'. There are also files in $ADFBIN with names 'adf', 'band', but these are in fact scripts to execute the binaries. We strongly recommend that you use these scripts in your calculations, in particular when running parallel jobs: the scripts take care of some aspects that you have to do otherwise yourself in each calculation.

- You need a license file to run any calculations successfully. If you have troubles with your license file, consult the Installation manual. If that doesn't help contact us at support@scm.com

Many of the provided samples have been devised to be short and simple, at the expense of physical or chemical relevance and precision or general quality of results. They serve primarily to illustrate the use of input, necessary files, and type of results. The descriptions have been kept brief. Extensive information about using keywords in input and their implications is given in the User's Guide and the Utilities and Property Programs documents (NMR, DIRAC, and other utility programs).

When you compare your own results with the sample outputs, you should check in particular (as far as applicable):

- Occupation numbers and energies of the one-electron orbitals;

- The optimized geometry;

- Vibrational frequencies;

- The bonding energy and the various terms in which it has been decomposed;

- The dipole moment;

- The logfile. At the end of a calculation the logfile is automatically appended (by the program itself) to the standard output.

General remarks about comparisons:

- For technical reasons, discussion of which is beyond the scope of this document, differences between results obtained on different machines, or with different numbers of parallel processes, may be much larger than you would expect. They may significantly exceed the machine precision. What you should check is that they fall well (by at least an order of magnitude) within the *numerical integration* precision used in the calculation.

- For similar reasons the orientation of the molecule used by the program may be different on different machines, even when the same input is supplied. In such cases the different orientations should be related and only differ in some trivial way, such as by a simple rotation of all coordinates by 90 degrees around the z-axis. When in doubt, contact an ADF representative.

- A BAND run may generate, apart from result files that you may want to save, a few scratch files. The UNIX scripts that run the samples take care of removing these files after the calculations have finished, to avoid that the program aborts in the next run by attempting to open a 'new' file that is found to exist already.

- A sample calculation may use one or more data files, in particular *fragment* files. The samples are self-contained: they first run the necessary pre-calculations to produce the fragment files. In 'normal' research work you may have libraries of fragments available, first for the 'basic atoms', and later, as projects are developing, also for larger fragments so that you can start immediately on the actual system by attaching the appropriate fragment files.

Default settings of print options result in a considerable amount of output. This is also the case in some of the sample runs, although in many of them quite a bit of 'standard' output is suppressed by inserting applicable print control keys in the input file. Consult the User's Guide about how to regulate input with keys in the input file.

13.2 Model Hamiltonians

13.2.1 Example: MetaGGA functionals

Download LiMetaGGA.run

Example for a post SCF metaGGA calculation.

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title Li bulk

KSpace 3

accuracy 5

DIIS
   dimix 0.3
   ncycledamp 0
end

scf
   mixing 0.6
end

xc
   metaGGA postscf
end

define
   ha=6.60/2
end

Lattice
   -ha   ha   ha
    ha  -ha   ha
    ha   ha  -ha
end

Atoms
   Li   0.0  0.0  0.0
end

BasisDefaults
   BasisType DZ
   Core Large
end

end input
eor
```

13.2.2 Example: Relativistic effects: Platinum slab

Download Pt_slab.run

This example can of course be compared directly to the Cu slab. This example is important, as SCF convergence is frequently difficult in slab calculations. The specifications in the `CONVERGENCE`, `SCF`, and `DIIS` blocks are typical. Such settings are recommended in slab calculations with convergence problems.

The `DEGENERATE` subkey specifies that bands with the same energy should have the same occupation numbers. This helps SCF convergence. The same is true for the values for the `MIXING` subkey in the SCF block and the `DIMIX` subkey in the `DIIS` block. Please note that the recommended value for `Mixing` is approximately half of the value for `Dimix`.

Another important feature in BAND is that it enables relativistic treatments for heavy nuclei. Both the `ZORA scalar relativistic` option and spin-orbit effects have been implemented. The line

```
Relativistic ZORA SPIN
```

specifies that in this case both the scalar relativistic effects (ZORA) and spin-orbit effects (SPIN) will be taken into account. Whereas the `ZORA` keyword does not make the calculation much more time-consuming, the same cannot be said for the spin-orbit option. Usually the ZORA keyword will give the most pronounced relativistic effects and the spin-orbit effects will be a fairly minor correction to that. We therefore recommend scalar ZORA as a good default method for treating heavy nuclei.

The `DEPENDENCY` keyword means that the calculation should continue even if the basis is nearly linearly dependent (as measured by the eigenvalues of the overlap matrix).

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title Platinum slab

Comment
 Technical
   Low quadratic K space integration
   Low real space integration accuracy
 Features
   Lattice   : 2D
   Unit cell : 3 atoms, 1x1
   Basis     : NO+STO w/ core
   Options   : Spinorbit ZORA
End

Convergence
   Degenerate 1.0E-03
End

SCF
   Iterations 60
   Mixing 0.06
End

DIIS
   NCycleDamp 15
   DiMix 0.15
End

KSpace 3
Accuracy 3

Relativistic ZORA SPIN

Dependency Basis=1E-8
```

```
Define
   latt=7.41
   lvec=latt/SQRT(2.0)
   ysh=lvec/SQRT(3.0)
   dlay=latt/SQRT(3.0)
End

Lattice
   SQRT(3.0)*lvec/2.0  0.5*lvec
   SQRT(3.0)*lvec/2.0 -0.5*lvec
End

Atoms
   Pt   0   0        0    :: layer 1
   Pt  -ysh 0.0    -dlay :: layer 2
   Pt   ysh 0.0 -2.0*dlay :: layer 3
End

END INPUT
eor
```

13.2.3 Example: Spin polarization: antiferromagnetic iron

Download BetaIron.run

With the UNRESTRICTED keyword we do a spin polarized calculation. Normally this would converge to the ferromagnetic solution.

With the SpinFlip keyword we make sure that we start with an antiferromagnetic density.

For antiferromagnetic iron we need a larger unit cell of two atoms. Since these atoms appear to the program as symmetry equivalent we have to specify them as separate types.

```
$ADFBIN/band << eor

TITLE Beta iron

UNITS
    length Angstrom
END

! The two iron atoms have different "types" to break the symmetry
ATOMS
  Fe.a   0.0    0.0    0.0
  Fe.b  -1.435 -1.435  1.435
end

Lattice
   -1.435  1.435  1.435
    1.435 -1.435  1.435
    2.87   2.87  -2.87
End

CONVERGENCE
   CRITERION 1.0e-4
   Degenerate default
```

```
   SpinFlip 2  ! Flip (startup) spin density at second atom
END

BasisDefaults
   BasisType DZ
   Core Large
End

UNRESTRICTED

end input
eor
```

13.2.4 Example: Graphene sheet with dispersion correction

`Download Graphene_Dispersion.run`

A normal GGA would give only negligible interaction between two graphene sheets.

Use the dispersion option in the `XC` key block.

In the first run we use **BP86-D**, in the second **BLYP-D3** and in the third run **BLYP-D3(BJ)**.

```
# First run: dispersion default

"$ADFBIN/band" <<eor
TITLE Dispersion energy with two parallel graphene sheets

XC
   gga scf bp86
   dispersion default
End

NumericalQuality Basic

GeometryFile $ADFHOME/examples/band/Graphene_Dispersion/Graphene_double_layer.xyz

BasisDefaults
   BasisType TZP
   Core Large
End

end input
eor

rm RUNKF

# Second run: dispersion Grimme3

"$ADFBIN/band" <<eor
TITLE Grimme3 dispersion energy with two parallel graphene sheets

XC
   gga scf blyp
   dispersion Grimme3
end
```

```
Gradients
End

NumericalQuality Basic

GeometryFile $ADFHOME/examples/band/Graphene_Dispersion/Graphene_double_layer.xyz

BasisDefaults
   BasisType TZP
   Core Large
End

end input
eor

rm RUNKF

# Third run: dispersion Grimme3 bjdamp

"$ADFBIN/band" <<eor

XC
   gga scf blyp
   dispersion Grimme3 bjdamp
end

Gradients
End

NumericalQuality Basic

GeometryFile $ADFHOME/examples/band/Graphene_Dispersion/Graphene_double_layer.xyz

BasisDefaults
   BasisType TZP
   Core Large
End

end input
eor
```

13.2.5 Example: H on perovskite with the COSMO solvation model

Download `HonPerovskite_Solvation.run`

We want to model H adsorption on a Perovskite surface in a solution, modeled by a COSMO surface.

We create only the COSMO surface above the slab with the `RemovePointsWithNegativeZ` option.

```
"$ADFBIN/band" <<eor

DefaultsConvention 2014

TITLE Hydrogen on Perovksite wit solvation

Solvation
   Surf Delley
```

```
    charge method=inver
End

PeriodicSolvation
   nstar 3
   SymmetrizeSurfacePoints true
   RemovePointsWithNegativeZ true
End

Screening
   rmadel 30 ! to speed up the calculation
End

Units
    length Angstrom
    angle Degree
End

Atoms
   H    0.0   0.000000000   0.900000000
   Ca   0.0   0.000000000   0.000000000
   Ca   0.0   3.535533906  -3.535533906
   Ti  -2.5  -3.535533906   0.000000000
   Ti  -2.5   0.000000000  -3.535533906
   O    0.0  -3.535533906   0.000000000
   O    2.5   1.767766953  -1.767766953
   O    2.5  -1.767766953  -1.767766953
End

Lattice
   5.0 0.000000000 0.0
   0.0 7.071067812 0.0
End

Gradients
End

Convergence
   Criterion 1.0e-4
End

BasisDefaults
   BasisType SZ
   Core Large
End

end input
eor
```

13.2.6 Example: Applying a homogeneous electric field

`Download EField.run`

With the `EFIELD` keyword you can specify a static electric field in the z-direction.

```
$ADFBIN/band << eor
DefaultsConvention 2014
```

```
Title Electric Field (field in a.u.)

EField
   Unit a.u.
   eZ=0.03
end

KSpace 1

Gradients
End

lattice
   15.0  0.00  0.00
   0.00  15.0  0.00
End

Atoms
   H    0.0 0.0 0.0
   Li   0.0 1.0 3.0
End

BasisDefaults
   BasisType TZP
   Core Large
End

end input
eor
```

13.2.7 Example: Finite nucleus

```
Download FiniteNucleus.run
```

Normally the nucleus is approximated as a point charge. However we can change this to a finite size. Properties that might be affected are *EFG* (page 60), and the *A-tensor* (page 60). For such calculations one needs to crank up the precision and also use a relativistic Hamiltonian.

```
#! /bin/sh

# First run: NuclarModel PointCharge

"$ADFBIN/band" <<eor

NuclearModel PointCharge

Efg
End

Atensor
End

Unrestricted
relativistic zora
```

```
PropertiesAtNuclei
  rho
  rho(deformation/scf)
  vxc[rho(fit)]
  rho(fit)
  v(coulomb)
End

RadialDefaults
   nr 10000
End

UNITS
    length Angstrom
    angle Degree
END

lattice
   30.0 0.0 0.0
End

ATOMS
   Au  0.000000      0.000000      0.000000
End

NumericalQuality Good

BasisDefaults
   BasisType TZ2P
   Core None
End

XC
   gga PBE
END

end input
eor

rm RUNKF
rm Points

# Second run: NuclearModel Gaussian

"$ADFBIN/band" <<eor

NuclearModel Gaussian

Efg
End

Atensor
End

Unrestricted
relativistic zora

PropertiesAtNuclei
```

```
   rho
   rho(deformation/scf)
   vxc[rho(fit)]
   rho(fit)
   v(coulomb)
End

RadialDefaults
    nr 10000
End

UNITS
    length Angstrom
    angle Degree
END

lattice
    30.0 0.0 0.0
End

ATOMS
   Au   0.000000      0.000000      0.000000
End

NumericalQuality Good

BasisDefaults
   BasisType TZ2P
   Core None
End

XC
   gga PBE
END

end input
eor
```

13.2.8 Example: Fixing the Band gap of NiO with GGA+U

```
Download NiO_Hubbard.run
```

With the UNRESTRICTED keyword we do a spin polarized calculation.

With the HubbardU key block we set up the GGA+U calculation. You need to specify per atom type (only two here, Ni, and O) the U and the *l*-value to which it should be applied.

```
$ADFBIN/band << eor
TITLE NiO

UNITS
    length Angstrom
    angle Degree
END

HubbardU
   printOccupations true
```

```
    Enabled true
    uvalue 0.3 0.0  ! Type 1 (Ni) will get U=0.3
    lvalue 2 -1     ! Will be used for the d-orbitals of Ni
End

Atoms
   Ni 0.0 0.0 0.0
   O 2.085 2.085 2.085
END

Lattice
   0.000 2.085 2.085
   2.085 0.000 2.085
   2.085 2.085 0.000
End

CONVERGENCE
   CRITERION 1.0e-4
   Degenerate default
END

DIIS
   CLARGE 10
   CHUGE 10
   DIMIX 0.2
   NVCTRX 20
   NCYCLEDAMP 0
END

SCF
   iterations 50
   mixing 0.1
END

BasisDefaults
   BasisType TZP
   Core Large
End

XC
   GGA Becke Perdew
END

UNRESTRICTED

Dependency basis=1e-6

end input
eor
```

13.2.9 Example: Fixing the band gap of ZnS with the TB-mBJ model potential

Download ZnS_ModelPotential.run

With the `XC` subkey model we invoke the so-called `TB-mBJ` model potential, which increases band gaps for solids.

```
$ADFBIN/band << eor
DefaultsConvention 2014

TITLE ZnS pot=TB-mBJ

XC
   model TB-mBJ
END

UNITS
   length Angstrom
   angle Degree
END

ATOMS
   Zn  0.0000  0.0000  0.0000
   S   1.3525  1.3525  1.3525
END

Lattice
   0.000000 2.705 2.705
   2.705 0.000000 2.705
   2.705 2.705 0.000000
End

BasisDefaults
   BasisType DZ
   Core Large
End

end input
eor
```

13.3 Precision and performance

13.3.1 Example: Convenient way to specify a basis set

`Download BasisDefaults.run`

This example shows some of the flexibility of the `BasisDefaults` key. The defaults are set to a DZ basis set with a Large frozen core. As the example shows, it is possible to override the defaults per atom type:

```
$ADFBIN/band << eor

Title CO + H2: fine tuning the basis defaults

NumericalQuality Basic

Atoms
   C               0.0   0.0   0.0
   O.large_basis  0.0   2.13  0.0
   H.large_basis  4.0   0.0   0.0
   H               4.0   1.43  0.0
End
```

```
BasisDefaults
   ! Cheap defaults
   BasisType DZ
   Core Large

   C             Core=None        ! This C has no frozen core
   O.large_basis  BasisType=TZ2P   ! This O with a larger basis
   H.large_basis  BasisType=V      ! This one also with a larger basis
End

END INPUT
eor
```

13.3.2 Example: Tuning precision and performance

```
Download Peptide_NumericalQuality.run
```

This example shows how to tune the numerical quality of the calculation. This will influence both efficiency and accuracy of the calculation.

```
$ADFBIN/band << eor

NumericalQuality Normal

ZlmFit
   Quality Normal
End

BeckeGrid
   Quality Basic
End

KSpace
   Quality Basic
End

SoftConfinement
   Quality VeryGood
End

UNITS
   length Angstrom
END

ATOMS
   C  -2.543276676    0.646016253   -0.226282061
   C  -1.380007216   -0.349821933   -0.099968062
   C   1.066549862   -0.581911934   -0.064823014
   C   2.223931363    0.423839954   -0.118070453
   N  -0.149937993    0.193000383   -0.179010633
   N   3.452833267   -0.128914507   -0.101813389
   O  -1.589886979   -1.564606357    0.062390357
   O   2.010772661    1.647347397   -0.186192833
   H  -2.480330907    1.422845016    0.554868474
   H   3.629655835   -1.142731500   -0.018098016
   H  -2.511564496    1.180719545   -1.193540463
   H   0.024515371    1.206808884   -0.244500253
```

```
   H   1.160598100   -1.320381370   -0.884522980
   H   1.071343640   -1.136930542    0.888913220
END

Lattice
      7.211585775    0.000000000    0.000000000
End

BasisDefaults
   BasisType DZ
   Core Large
End

XC
   GGA PBE
END


end input
eor
```

13.3.3 Example: Multiresolution

Download H2O_Multiresolution.run

This example demonstrates how to use different levels of numerical precision for different regions, with the aim of increasing computational efficiency.

Let us assume that we are interested in having an accurate description only for a subregion of a large chemical system (in this simple example, the central water molecule embedded in a small water cluster). The system can be divided into sub-regions and different levels of numerical accuracy can be used for each of these sub-regions.[*53* (page 208)]

In this example we will tweak the basis set (`BasisType`), the numerical integration quality (`BeckeGrid`) and the density fitting quality (`ZlmFit`):

- Region_1 (central water molecule): All-electron TZP basis set, good integration and fitting quality;

- Region_2 (nearby water molecules): DZP basis set, normal integration and fitting quality;

- Region_3 (far away water molecules): DZ basis set, basic integration and fitting quality.

Note: For the atoms that have not been explicitly defined in the `AtomDepQuality` sub-blocks, the quality defined in `NumericalQuality` will be used (Basic in this example).

```
$ADFBIN/band << eor

UNITS
   length Angstrom
END


Atoms
  O.Region_1      0.00000000      0.00000000      0.00000000   !   1
  H.Region_1      0.95598067     -0.00675125      0.04695605   !   2
  H.Region_1     -0.27235084      0.54445503      0.73860412   !   3
  O.Region_2     -0.60494216      1.46490169      2.08012471   !   4
  H.Region_2     -1.36174124      1.46490169      2.66613184   !   5
  H.Region_2      0.15185693      1.46490169      2.66613184   !   6
  O.Region_2      2.20300071     -1.46837786     -1.32565612   !   7
```

```
   H.Region_2       1.41269969      -1.75817168      -1.78128023     !    8
   H.Region_2       2.91509001      -1.94050071      -1.75713167     !    9
   O.Region_2      -1.50296488       0.38335767      -2.26270407     !   10
   H.Region_2      -1.22374034       1.23595156      -2.59627085     !   11
   H.Region_2      -0.94183314       0.22778137      -1.50305317     !   12
   O.Region_2      -1.26586015      -2.22743706       0.52905656     !   13
   H.Region_2      -1.86350326      -2.02544308       1.24889726     !   14
   H.Region_2      -0.71759486      -1.44720433       0.44668931     !   15
   O.Region_2       2.56741860      -1.62760305       1.40348274     !   16
   H.Region_2       2.62568379      -1.68307735       0.44971272     !   17
   H.Region_2       3.11146028      -0.87440516       1.63338252     !   18
   O.Region_3       1.85561863       3.40985337       0.07797373     !   19
   H.Region_3       2.20215679       2.60864811      -0.31462138     !   20
   H.Region_3       1.19434847       3.10556187       0.69948059     !   21
   O.Region_3       0.08469270      -1.99944529      -2.83628507     !   22
   H.Region_3      -0.58300996      -2.66470790      -2.66970682     !   23
   H.Region_3      -0.37211705      -1.16665310      -2.71828366     !   24
   O.Region_3       2.96038454       1.32696848      -1.40635988     !   25
   H.Region_3       2.86805676       0.38177416      -1.52566106     !   26
   H.Region_3       3.78683519       1.54337235      -1.83797428     !   27
   O.Region_3       0.28813964      -4.51118280       0.55801684     !   28
   H.Region_3       0.19940883      -4.70530100      -0.37503968     !   29
   H.Region_3      -0.11511472      -3.64948276       0.66290372     !   30
   O.Region_3      -3.63296983      -2.55354516      -0.88850230     !   31
   H.Region_3      -3.58826479      -2.50371291      -1.84331521     !   32
   H.Region_3      -2.72024484      -2.51147134      -0.60336163     !   33
   O.Region_3       1.76012648       1.19830856       3.25007043     !   34
   H.Region_3       1.65653616       0.36151385       3.70305293     !   35
   H.Region_3       2.52729927       1.07488325       2.69118366     !   36
   O.Region_3      -3.73810993      -0.13907200       0.75150467     !   37
   H.Region_3      -3.66396331      -0.94861779       0.24624494     !   38
   H.Region_3      -3.89254913       0.53994702       0.09482818     !   39
   O.Region_3      -3.98747199       1.47439249      -1.48255293     !   40
   H.Region_3      -3.68541503       2.11381006      -0.83752870     !   41
   H.Region_3      -3.18462420       1.07965163      -1.82278430     !   42
   O.Region_3      -3.36484791       1.08559064       3.12556138     !   43
   H.Region_3      -3.71752974       0.74651716       2.30288636     !   44
   H.Region_3      -3.68790098       0.47982864       3.79251822     !   45
   O.Region_3       2.15143451      -2.15387984       4.03356174     !   46
   H.Region_3       2.18159460      -2.07271050       3.08032966     !   47
   H.Region_3       2.77736140      -1.50229172       4.34945851     !   48
End

BasisDefaults
   BasisType DZ
   O.Region_1 BasisType=TZP Core=None
   H.Region_1 BasisType=TZP Core=None
   O.Region_2 BasisType=DZP
   H.Region_2 BasisType=DZP
End




NumericalQuality Basic

SoftConfinement
   Quality Normal
End
```

```
BeckeGrid
   AtomDepQuality
        1 Good
        2 Good
        3 Good
        4 Normal
        5 Normal
        6 Normal
        7 Normal
        8 Normal
        9 Normal
       10 Normal
       11 Normal
       12 Normal
       13 Normal
       14 Normal
       15 Normal
       16 Normal
       17 Normal
       18 Normal
   SubEnd
End

ZlmFit
   AtomDepQuality
        1 Good
        2 Good
        3 Good
        4 Normal
        5 Normal
        6 Normal
        7 Normal
        8 Normal
        9 Normal
       10 Normal
       11 Normal
       12 Normal
       13 Normal
       14 Normal
       15 Normal
       16 Normal
       17 Normal
       18 Normal
   SubEnd
End

END INPUT
eor
```

13.3.4 Example: BSSE correction

```
Download BSSE.run
```

This example shows how to calculate the basis set superposition error for the interaction of CO with H_2. In this shell script we loop over progressively better basis sets.

```
#!/bin/bash


for bas in DZ TZ2P QZ4P
do

core=Large


$ADFBIN/band <<EOF

Atoms
C 0 0 0
O 2.13 0 0
H 0 0 6
H 1.5 0 6
End


BasisDefaults
  BasisType $bas
  Core $core
End


end input

EOF

ECOH2=`$ADFBIN/adfreport  RUNKF -r 'Bond energies%final bond energy'`

rm RUNKF

$ADFBIN/band <<EOF

Atoms
H 0 0 6
H 1.5 0 6
End


BasisDefaults
  BasisType $bas
  Core $core
End


end input

EOF

EH2=`$ADFBIN/adfreport  RUNKF -r 'Bond energies%final bond energy'`


rm RUNKF

$ADFBIN/band <<EOF
```

```
Atoms
Gh.C 0 0 0
Gh.O 2.13 0 0
H 0 0 6
H 1.5 0 6
End


BasisDefaults
  BasisType $bas
  Core $core
End


end input

EOF

EH2_GHOST_CO=`$ADFBIN/adfreport  RUNKF -r 'Bond energies%final bond energy'`



rm RUNKF

$ADFBIN/band <<EOF

Atoms
C 0 0 0
O 2.13 0 0
End


BasisDefaults
  BasisType $bas
  Core $core
End


end input

EOF

ECO=`$ADFBIN/adfreport  RUNKF -r 'Bond energies%final bond energy'`

rm RUNKF

$ADFBIN/band <<EOF

Atoms
C 0 0 0
O 2.13 0 0
Gh.H 0 0 6
Gh.H 1.5 0 6
End


BasisDefaults
  BasisType $bas
```

```
  Core $core
End


end input

EOF

ECO_GHOST_H2=`$ADFBIN/adfreport  RUNKF -r 'Bond energies%final bond energy'`

rm RUNKF

EV=27.212
echo "Start report"
echo "basis set: $bas"
echo "H2 + CO : $ECOH2"
echo "H2 : $EH2"
echo "H2 (with ghost CO) : $EH2_GHOST_CO"
echo "CO : $ECO"
echo "CO (with ghost H2) : $ECO_GHOST_H2"
BSSEEV=`$ADFBIN/startpython -c "print ( $EH2 - $EH2_GHOST_CO + $ECO - $ECO_GHOST_H2 )␣
→*$EV"`
echo "BSSE correction: $BSSEEV (eV)"
BOND1EV=`$ADFBIN/startpython -c "print ( $ECOH2 - $EH2 - $ECO ) *$EV"`
BOND2EV=`$ADFBIN/startpython -c "print $BOND1EV + $BSSEEV"`
echo "Bond energy: $BOND1EV (eV)"
echo "Bond energy + BSSE: $BOND2EV (eV)"
echo "End report"

done
```

13.4 Restarts

13.4.1 Example: Restart the SCF

`Download RestartSCF.run`

This example shows how you can continue with an unfinished calculation. It consists of two runs. After the first run
the RUNKF file is saved, and the renamed file is used in the second run. The second run is almost a copy for the first,
except for the *Restart* (page 87) key. It is also possible to restart from a smaller basis set (provided that the functions
are contained in the bigger basis set). Finally you can also restart from a density matrix, but this should be explicitly
saved (unlike the orbitals).

```
# ---------------------------- first run --------------------------

$ADFBIN/band << eor
Title B chain

skip dos

XC
   GGA Becke Perdew
END

UNRESTRICTED
```

```
DIIS
   NCycleDamp 0
   DiMix 0.5
   Adaptable false ! Otherwise it converges to a spin-restricted solution
End

Define
   ddd=4.0
   ccc=1.5
End

Lattice
   ddd
End

Atoms
   B 0.0 0.0 0.0
End

BasisDefaults
   BasisType TZ2P
   Core Large
End

END INPUT
eor

mv RUNKF BChain.runkf

# --------------------------- second run -------------------------

$ADFBIN/band << eor
Title B chain restart

XC
   GGA Becke Perdew
END

UNRESTRICTED

Restart
   File BChain.runkf
   scf
end

Define
   ddd=4.0
   zzz=3
End

Lattice
   ddd
End

Atoms
   B 0.0 0.0 0.0
End
```

```
BasisDefaults
   BasisType TZ2P
   Core Large
End

END INPUT
eor

rm RUNKF

# ---------------------------- third run -------------------------

$ADFBIN/band << eor
Title B chain bas=SZ

Save DensityMatrix

skip dos

XC
   GGA Becke Perdew
END

UNRESTRICTED

DIIS
   NCycleDamp 0
   DiMix 0.3
   Adaptable false ! Otherwise it converges to a spin-restricted solution
End

Define
   ddd=4.0
   ccc=1.5
End

Lattice
   ddd
End

Atoms
   B 0.0 0.0 0.0
End

BasisDefaults
   BasisType SZ
   Core Large
End

END INPUT
eor

mv RUNKF BChain.bas=SZ.runkf

# ---------------------------- fourth run ------------------------

$ADFBIN/band << eor
```

```
Title B chain restart bas=SZ from density matrix


XC
   GGA Becke Perdew
END

UNRESTRICTED

Restart
   File BChain.bas=SZ.runkf
   scf
   useDensityMatrix true
end

Define
   ddd=4.0
   zzz=3
End

Lattice
   ddd
End

Atoms
   B 0.0 0.0 0.0
End

BasisDefaults
   BasisType SZ
   Core Large
End

END INPUT
eor

rm RUNKF



# --------------------------- fifth run --------------------------

$ADFBIN/band << eor
Title B chain restart bas=TZ2P from orbitals


XC
   GGA Becke Perdew
END

UNRESTRICTED

Restart
   File BChain.bas=SZ.runkf
   scf
   useDensityMatrix false
end

Define
```

```
   ddd=4.0
   zzz=3
End

Lattice
   ddd
End

Atoms
   B 0.0 0.0 0.0
End

BasisDefaults
   BasisType TZ2P
   Core Large
End

END INPUT
eor

rm RUNKF

# ---------------------------- sixth run --------------------------

$ADFBIN/band << eor
Title B chain restart bas=TZ2P from density matrix (bas=SZ)

XC
   GGA Becke Perdew
END

UNRESTRICTED

Restart
   File BChain.bas=SZ.runkf
   scf
   useDensityMatrix true
end

Define
   ddd=4.0
   zzz=3
End

Lattice
   ddd
End

Atoms
   B 0.0 0.0 0.0
End

BasisDefaults
   BasisType TZ2P
   Core Large
End

END INPUT
```

```
eor

rm RUNKF
```

13.4.2 Example: Restart SCF for properties calculation

Download RestartProperties.run

This example shows how to restart the SCF and compute various properties, like a density of states, and a bandstructure plot, or the effective mass.

```
# ======================
# polyethylene .xyz file:
# ======================

cat << eor > polyethylene.xyz
6

C      -0.623348981    -0.055000000     0.425969423
C       0.633348981     0.015000000    -0.422636089
H      -0.633348981     0.964974570     1.055290696
H      -0.623348981    -0.914974570     1.055290696
H       0.633348981     0.904974570    -1.051957363
H       0.613348981    -0.914974570    -1.061957363
VEC1   2.553395923     0.000000000     0.000000000
eor


# =================================================
# Simple single point calculation (no properties)
# =================================================

$ADFBIN/band <<EOF
  GeometryFile polyethylene.xyz

  BasisDefaults
    BasisType DZ
  End

  Unrestricted

  End input
EOF

mv RUNKF restart.runkf


# ==================================
# Restart and compute some properties
# ==================================

$ADFBIN/band <<EOF
  GeometryFile polyethylene.xyz

  BasisDefaults
    BasisType DZ
  End
```

```
   Unrestricted

   Restart
      SCF
      File restart.runkf
   End

   DOS
      Enabled True
   End

   BZStruct
      Enabled True
      DeltaK 0.3
   End

   EffectiveMass
   End

   End input
EOF
```

13.4.3 Example: Properties on a grid

Download `BeO_tape41.run`

Saving the RUNKF file of a calculation gives rise to the opportunity to restart from it to calculate properties on a grid, like densities, potentials, or crystal orbitals. Find more details here.

Regarding the following example, in the first run we perform a single-point calculation for a bulk BeO system. After the calculation finished the *RUNKF* file shall be renamed to *BeO.runkf*. In the second run we restart from this file. We specify to use a regular grid and ask the program to calculate a bunch of properties on that grid.

First job:

```
$ADFBIN/band << eor
Title BeO

Comment
   in the "ideal" most symmetric configuration: u=3/8, and c=sqrt(8/3)*a
End

DefaultsConvention 2014

NumericalQuality Basic

xc
  gga scf bp86
end

Define
  uuu=3/8
  aaa=5.10
  ccc=sqrt(8/3)*aaa
End

Coordinates Fractional
```

```
ATOMS
  Be  0 0 0
  Be  1/3 1/3 1/2
  O    0 0 uuu
  O    1/3 1/3 uuu+1/2
END

Lattice
  aaa 0 0
  0.5*aaa 0.5*sqrt(3)*aaa 0
  0 0 ccc
End

BasisDefaults
  BasisType DZ
  Core large
end

end input
eor

mv RUNKF BeO.runkf
```

Second job:

```
$ADFBIN/band << eor
DefaultsConvention 2014

Title BeO

Comment
 in the "ideal" most symmetric configuration: u=3/8, and c=sqrt(8/3)*a
End

Restart
  File BeO.runkf
  DensityPlot
End

Grid
   Type Coarse
End

DensityPlot
   rho(deformation/fit)
   rho(fit)
   rho(atoms)
   v(coulomb/atoms)
   v(coulomb)
   vxc[rho(fit)]
End

NumericalQuality Basic

xc
  gga scf bp86
end
```

```
Define
  uuu=3/8
  aaa=5.10
  ccc=sqrt(8/3)*aaa
End

Coordinates Fractional

ATOMS
  Be  0 0 0
  Be  1/3 1/3 1/2
  O    0 0 uuu
  O    1/3 1/3 uuu+1/2
END

Lattice
  aaa 0 0
  0.5*aaa 0.5*sqrt(3)*aaa 0
  0 0 ccc
End

BasisDefaults
  BasisType DZ
  Core large
end

end input
eor
```

13.5 NEGF

13.5.1 Example: Main NEGF flavors

`Download NEGF_Cr_wire.run`

This example shows how to use the *NEGF* (page 67) functionality.

It starts of with **Method 1**: the non-self consistent approach. Here, BAND merely serves to provide matrix elements, being unaware of the electrodes.

Then follows **Method 2**: here the NEGF density is really used to calculate the matrix elements.

Method 3 is a variation on Method 2, and includes an extra alignment run.

Setting up a NEGF calculation is quite hard without the GUI.

```
#! /bin/sh

#␣
↪=================================================================================================
#         Method #1: non-self consistent NEGF (uses the conductance program, like␣
↪DFTB)
#␣
↪=================================================================================================
```

```
# ======================================  Method #1. Run #1  ␣
↪======================================

"$ADFBIN/band" <<eor
Title method_1_run_1

KSpace
    Quality Good
End

NumericalQuality Basic

Units
    length Angstrom
END

ATOMS
   Cr.1 0.0 0.0 0.0
   Cr.2 2.5 0.0 0.0
END

Lattice
   5.0 0.0 0.0
End

BasisDefaults
   BasisType DZ
   Core Large
End

Unrestricted

StoreHamiltonian2

end input
eor

mv RUNKF lead.runkf

# ======================================  Method #1. Run #2  ␣
↪======================================

"$ADFBIN/band" <<eor
Title method_1_run_2

NumericalQuality Basic

Units
    length Angstrom
END

ATOMS
    Cr.1L  -10.0 0.0 0.0
    Cr.2L   -7.5 0.0 0.0
    Cr.C    -5.0 0.0 0.0
    Cr.C    -2.5 0.0 0.0
    Cr.C     0.0 0.0 0.0
    Cr.C     2.5 0.0 0.0
```

```
    Cr.C    5.0 0.0 0.0
    Cr.1R   7.5 0.0 0.0
    Cr.2R  10.0 0.0 0.0
END

Lattice
   22.5 0.0 0.0
End

BasisDefaults
   BasisType DZ
   Core Large
End

Unrestricted

StoreHamiltonian2
StoreHamAsMol

end input
eor

mv RUNKF scattering.runkf

# ====================================  Method #1. Run #3 ␣
↪====================================

$ADFBIN/conductance <<EOF
EnergyGrid min=-5 max=5 num=200
Files
   Leads      lead.runkf
   Scattering scattering.runkf
End

end input
EOF

mv scattering.runkf RUNKF
$ADFBIN/cpkf "ConductanceResults.kf" "RUNKF" "results --rename NEGF"


echo 'StartTestingMe'
    echo "Method 1. Transmission"
    $ADFBIN/adfreport RUNKF -r "NEGF%transmission#12.5f##1"
echo 'StopTestingMe'

rm RUNKF

#␣
↪===================================================================================================
#                      Method #2: self consistent NEGF without alignment
#␣
↪===================================================================================================


# ===================================  Method #2. Run #1 ␣
↪===================================
```

```
"$ADFBIN/band" <<eor
Title method_2_run_1

KSpace
    Quality Good
End

NumericalQuality Basic

UNITS
    length Angstrom
END

ATOMS
    Cr.1 0.0 0.0 0.0
    Cr.2 2.5 0.0 0.0
END

Lattice
 5.0 0.0 0.0
End

BasisDefaults
  BasisType DZ
  Core Large
End

Unrestricted
StoreHamiltonian2

end input
eor

# ====================================== Method #2. Run #2 ␣
↪======================================

$ADFBIN/sgf   << eor
TITLE Test for NEGF inputs
SAVE SIGMA
SURFACEGF
SCMCode
   KT 0.001
   ContourQuality normal
END
eor

mv RUNKF TB.runkf
mv SigmaSCM Sigma.runkf

# ====================================== Method #2. Run #3 ␣
↪======================================

"$ADFBIN/band" <<eor
Title method_2_run_3

NumericalQuality Basic

UNITS
```

```
        length Angstrom
END

ATOMS
    Cr.1L -10.0 0.0 0.0
    Cr.2L -7.5 0.0 0.0
    Cr.C -5.0 0.0 0.0
    Cr.C -2.5 0.0 0.0
    Cr.C 0.0 0.0 0.0
    Cr.C 2.5 0.0 0.0
    Cr.C 5.0 0.0 0.0
    Cr.1R 7.5 0.0 0.0
    Cr.2R 10.0 0.0 0.0
END

BasisDefaults
  BasisType DZ
  Core Large
End

Unrestricted
NEGF
  LeadFile        TB.runkf
  SGFFile         Sigma.runkf
  ApplyShift2     False
  ContourQuality  normal
  EMin           -0.1837465475
  EMax            0.1837465475
  NE 200
End

end input
eor

echo 'StartTestingMe'
    echo "Method 2. Transmission"
    $ADFBIN/adfreport RUNKF -r "NEGF%transmission#12.5f##1"
echo 'StopTestingMe'

rm RUNKF

#␣
↪===========================================================================================================
#                        Method #3: self consistent NEGF with alignment run
#␣
↪===========================================================================================================

# =====================================  Method #3. Run #1  ␣
↪=====================================

"$ADFBIN/band" <<eor
Title method_3_run_1

KSpace
    Quality Good
End

NumericalQuality Basic
```

```
UNITS
    length Angstrom
END

ATOMS
    Cr.1 0.0 0.0 0.0
    Cr.2 2.5 0.0 0.0
END

Lattice
 5.0 0.0 0.0
End

BasisDefaults
  BasisType DZ
  Core Large
End

Unrestricted
StoreHamiltonian2

end input
eor

# ======================================  Method #3. Run #2  ␣
↪======================================

$ADFBIN/sgf   << eor
TITLE Test for NEGF inputs
SAVE SIGMA
SURFACEGF
SCMCode
   KT 0.001
   ContourQuality normal
END
eor

mv RUNKF TB.runkf
mv SigmaSCM Sigma.runkf

# ======================================  Method #3. Run #3  ␣
↪======================================

"$ADFBIN/band" <<eor
Title method_3_run_3

NumericalQuality Basic

UNITS
    length Angstrom
END

ATOMS
    Cr.1L 0.0 0.0 0.0
    Cr.2L 2.5 0.0 0.0
    Cr.C 5.0 0.0 0.0
    Cr.C 7.5 0.0 0.0
```

```
      Cr.C 10.0 0.0 0.0
      Cr.C 12.5 0.0 0.0
      Cr.1R 15.0 0.0 0.0
      Cr.2R 17.5 0.0 0.0
END

BasisDefaults
  BasisType DZ
  Core Large
End

Unrestricted
NEGF
  DoAlignment True
  LeadFile TB.runkf
  SGFFile Sigma.runkf
  ContourQuality normal
  EMin -0.1837465475
  EMax 0.1837465475
  NE 200
  AlignChargeTol 0.0001
End

end input
eor

mv RUNKF align.runkf

# ======================================  Method #3. Run #4 ␣
↪======================================

"$ADFBIN/band" <<eor
Title method_3_run_4

NumericalQuality Basic

UNITS
    length Angstrom
END

ATOMS
    Cr.1L -10.0 0.0 0.0
    Cr.2L -7.5 0.0 0.0
    Cr.C -5.0 0.0 0.0
    Cr.C -2.5 0.0 0.0
    Cr.C 0.0 0.0 0.0
    Cr.C 2.5 0.0 0.0
    Cr.C 5.0 0.0 0.0
    Cr.1R 7.5 0.0 0.0
    Cr.2R 10.0 0.0 0.0
END

BasisDefaults
  BasisType DZ
  Core Large
End

Unrestricted
```

```
NEGF
   LeadFile TB.runkf
   SGFFile Sigma.runkf
   AlignmentFile align.runkf
   ContourQuality normal
   EMin -0.1837465475
   EMax 0.1837465475
   NE 200
End

end input
eor

echo 'StartTestingMe'
    echo "Method 3. Transmission"
    $ADFBIN/adfreport RUNKF -r "NEGF%transmission#12.5f##1"
echo 'StopTestingMe'

rm RUNKF
```

13.5.2 Example: NEGF with bias

```
Download NEGF_bias.run
```

This example shows how to use the *NEGF* (page 67) key when including a bias potential between the electrodes. It starts of with the usual *tight-binding* run, followed by an *SGF* one. The *alignment* run is omitted. Finally, there is a loop over bias potentials. Here the scale feature of the *FuzzyPotential* (page 29) is used. The current is appended to a text file, which one could plot eg. with *gnuplot*.

Setting up a NEGF calculation is quite hard without the GUI.

```
"$ADFBIN/band" <<eor
TITLE tight-binding

KSpace
    Quality VeryGood
End

SoftConfinement
    Quality Basic
End

UNITS
    length Angstrom
    angle Degree
END

ATOMS
    Li.1 0.0 0.0 0.0
    Li.2 2.876 0.0 0.0
    Li.3 5.752 0.0 0.0
END

Lattice
 8.628 0.0 0.0
End
```

```
BasisDefaults
BasisType DZ
Core Large
End

StoreHamiltonian2

end input
eor


$ADFBIN/sgf   << eor
TITLE Test for NEGF inputs
SAVE SIGMA
SURFACEGF
SCMCode
   KT 0.001
   ContourQuality normal
END
eor

mv RUNKF TB.runkf
mv SigmaSCM Sigma.runkf

REPORT=Li-CuAg.report
touch $REPORT

for bias in -0.01 0.01
do

"$ADFBIN/band" <<eor
TITLE bias=$bias

SoftConfinement
    Quality Basic
End

UNITS
    length Angstrom
END


ATOMS
    Li.1L -15.818 0.0 0.0
    Li.2L -12.942 0.0 0.0
    Li.3L -10.066 0.0 0.0
    Li.1C -7.19 0.0 0.0
    Li.2C -4.314 0.0 0.0
    Cu.C -0.7 -1.0 0
    Ag.C 0.7 1.0 0
    Li.3C 4.314 0.0 0.0
    Li.4C 7.19 0.0 0.0
    Li.1R 10.066 0.0 0.0
    Li.2R 12.942 0.0 0.0
    Li.3R 15.818 0.0 0.0
END
```

```
BasisDefaults
BasisType DZ
Core Large
End


NEGF
  LeadFile TB.runkf
  SGFFile Sigma.runkf
  EMin -0.1837465475
  EMax 0.1837465475
  NE 200
  ApplyShift2 False
  BiasPotential $bias
End

FuzzyPotential
scale $bias
1   0.5
2   0.5
3   0.5
4   0.5
5   0.5
6   0.2
7   -0.2
8   -0.5
9   -0.5
10  -0.5
11  -0.5
12  -0.5
end


end input
eor
current=`$ADFBIN/adfreport RUNKF 'NEGF%current'`
echo "NEGFREPORT: Bias=$bias, Current=$current" >> $REPORT

rm RUNKF

fname=`ls Transmission_*.plt`

echo "start of transmission (bias=$bias)"
cat $fname
echo "end of transmission"

rm Transmission_*.plt

done

echo "Start of report"
cat $REPORT
echo "End of report"
```

13.6 Structure and Reactivity

13.6.1 Example: NaCl: Bulk Crystal

Download NaCl.run

A bulk crystal computation for Sodium Chloride (common salt), with a subsequent DOS analysis, using a Restart facility to use the results from a preceding calculation.

The BAND input follows slightly different conventions from the ADF input, for historical reasons.

The COMMENT keyword allows users to provide some information about the run which may be of use later. Usually a brief summary of the run is given here.

Numerical integration precision is controlled with the key *Accuracy* (page 39). The accuracy for integrals over the Brillouin Zone is set by the *KSpace* (page 39) key. The latter should, generally, take as value an *odd* number (3, 5...) to invoke the more accurate *quadratic* tetrahedron integration procedure. For *even* values it will revert to the *linear* tetrahedron method, which is almost always inferior in accuracy.

The key Angstroms specifies that geometric data, such as lattice constants are in Angstrom units.

Since there are 3 data records in the Lattice block, the calculation will assume 3-dimensional periodicity, with lattice vectors as indicated. Note that lattice vectors are undefined up to linear combinations among themselves. Internally, the program will recombine the input vectors so as to minimize the size of the actually used vectors.

The input line Coordinates Fractional means that atomic positions are input as coefficients in terms of the lattice vectors, rather than as absolute (Cartesian) coordinate values.

For each of the atoms in the calculations, Na and Cl here, there must be data blocks to specify various items. First, their positions in the crystal unit cell (key Atoms). Second, the single isolated atom computation that will serve as start-up (Dirac). Third, any Slater-type orbital basis functions (BasisFunctions) for that atom. Fourth, the fit functions (FitFunctions) for the calculation of the Coulomb potential and. The third item (BasisFunctions) is optional and not present in this example.

The basis set, numerical atomic orbitals that are computed by the Herman-Skillman type subprogram DIRAC, and the fit sets that one has to insert into the BAND input files can be taken from the database you find in the directory *$ADFHOME/atomicdata/Band*. In contrast to the shown example, a more comfortable and automatized way of specifying the basis and fit sets is the keyblock *BasisDefaults* (page 31).

After the first calculation, a single point calculation of bulk NaCl, is finished the standard result file from BAND, the *RUNKF* file, is saved under the name 'NaCl.runkf'. This file will be used in the follow-up calculation of Density-of-States properties.

```
$ADFBIN/band << eor
Title Title NaCl (from neutral atoms)

Comment
 Technical
   Hybrid K space integration (3D)
   Low real space integration accuracy
   Fractional coordinates
   Lengths in Angstrom
   Parameters Dirac procedure
 Features
   Lattice   : 3D
   Unit cell : 2 atoms
   Basis     : NO w/ core
   Options   : Save restart file
End
```

```
Accuracy 3.5
Kspace 3

Units
 Length Angstrom
End

Lattice
  0    2.75   2.75
  2.75  0     2.75
  2.75  2.75   0
End

Coordinates Fractional
Atoms
  Na 0.0 0.0 0.0
  Cl 0.5 0.5 0.5
End

AtomType Na
Dirac   Na
  4 1
  Radial  2000
  RMin    1E-4
  RMax    60
  VALENCE
  1 0
  2 0
  2 1
  3 0 1.0
SubEnd

FitFunctions
  1 0 18.9
  2 0 30.3
  2 0 15.5
  3 0 14.9
  3 0  8.9
  4 0  7.8
  4 0  5.1
  4 0  3.3
  5 0  2.8
  5 0  1.9
  5 0  1.3
  2 1 14.3
  3 1  9.9
  4 1  6.7
  4 1  3.4
  5 1  2.4
  5 1  1.3
  3 2 10.5
  4 2  5.4
  5 2  3.0
  5 2  1.3
  4 3  5.8
  5 3  1.7
  5 4  2.0
```

```
SubEnd
End

AtomType Cl
Dirac  Cl
  5 3
  VALENCE
  1 0
  2 0
  2 1
  3 0
  3 1  5.0
SubEnd

FitFunctions
  1 0 29.1
  2 0 49.5
  2 0 26.1
  3 0 25.8
  3 0 15.8
  4 0 14.2
  4 0  9.4
  4 0  6.2
  5 0  5.4
  5 0  3.8
  5 0  2.6

  2 1 21.2
  3 1 16.5
  4 1 12.4
  4 1  6.8
  5 1  5.1
  5 1  3.1

  3 2 16.6
  4 2  9.4
  5 2  5.5
  5 2  2.6

  4 3  8.7
  5 3  3.3

  5 4  4.0
SubEnd
End

End Input
eor

mv RUNKF NaCl.runkf

rm Points
```

The next run has largely the same input and provides a restart of the previous run.

The key `DOS` in the block `Restart` tells the program to pick up the indicated file as restart file *and* to use it for DOS analysis purposes.

The `DOS` key block details the energy grid (and range) and the file to write the data to. The optional keys

GROSSPOPULATIONS and OverlapPopulations invoke the computation of, respectively, gross populations and overlap populations (i.e. for each of these the density-of-states values in the user-defined energy grid).

```
$ADFBIN/band << eor
Title Title NaCl (from neutral atoms)   DOS analysis (restart)

Comment
 Technical
   Hybrid K space integration (3D)
   Low real space integration accuracy
   Fractional coordinates
   Lengths in Angstrom
   Parameters Dirac procedure
 Features
   Lattice   : 3D
   Unit cell : 2 atoms
   Basis     : NO w/ core
   Options   : Use restart file for DOS
               Analysis: DOS, PDOS, COOP
End

Restart
  File NaCl.runkf
  DOS
End

Accuracy 3.5
Kspace 3

SCF
 Iterations 15
End

Units
 Length Angstrom
End

Lattice
  0    2.75   2.75
  2.75  0     2.75
  2.75  2.75   0
End

Coordinates Fractional
Atoms
  Na 0.0 0.0 0.0
  Cl 0.5 0.5 0.5
End

DOS
  File NaCl.dos
  Energies 1000
  Min −0.5
  Max  0.5
End

GROSSPOPULATIONS
 FRAG 1
 FRAG 2
```

```
 SUM
   1 0
   2 0
 ENDSUM
End

OVERLAPPOPULATIONS
LEFT
 FRAG 1
RIGHT
 FRAG 2
LEFT
  1 0
  1 1
RIGHT
  2 0
  2 1
End

AtomType Na
Dirac   Na
  4 1
  Radial  2000
  RMin    1E-4
  RMax    60
  VALENCE
  1 0
  2 0
  2 1
  3 0 1.0
SubEnd

FitFunctions
  1 0 18.9
  2 0 30.3
  2 0 15.5
  3 0 14.9
  3 0  8.9
  4 0  7.8
  4 0  5.1
  4 0  3.3
  5 0  2.8
  5 0  1.9
  5 0  1.3
  2 1 14.3
  3 1  9.9
  4 1  6.7
  4 1  3.4
  5 1  2.4
  5 1  1.3
  3 2 10.5
  4 2  5.4
  5 2  3.0
  5 2  1.3
  4 3  5.8
  5 3  1.7
  5 4  2.0
SubEnd
```

```
End

AtomType Cl
Dirac  Cl
  5 3
  VALENCE
  1 0
  2 0
  2 1
  3 0
  3 1  5.0
SubEnd

FitFunctions
  1 0 29.1
  2 0 49.5
  2 0 26.1
  3 0 25.8
  3 0 15.8
  4 0 14.2
  4 0  9.4
  4 0  6.2
  5 0  5.4
  5 0  3.8
  5 0  2.6

  2 1 21.2
  3 1 16.5
  4 1 12.4
  4 1  6.8
  5 1  5.1
  5 1  3.1

  3 2 16.6
  4 2  9.4
  5 2  5.5
  5 2  2.6

  4 3  8.7
  5 3  3.3

  5 4  4.0
SubEnd
End

End Input
eor
```

Finally, we copy the contents of the DOS result file to standard output

```
echo Contents of DOS file
cat NaCl.dos
```

13.6.2 Example: Atomic energies

```
Download H_ref.run
```

This example consists of several atomic energy calculations:

- Formation energy of the H-atom w.r.t. spherical atom

- Formation energy of the H-atom w.r.t. spherical atom

- Spin polarization energy of the H-atom w.r.t. spherical atom

- Spin polarization (relativistic) energy of the H-atom w.r.t. spherical atom

- Spin polarization energy of the H-atom w.r.t. spin unrestricted atom

- Spin polarization (relativistic) energy of the H-atom w.r.t. spin unrestricted atom

Only the first run is given here:

```
$ADFBIN/band << eor
Title Formation energy of the H-atom w.r.t. spherical atom

Comment
 Technical
   Quadratic K space integration
   High real space integration accuracy
 Features
   Lattice   : 2D, quasi atom
   Unit cell : 1 atom, 1x1
   Basis     : NO+STO
End

Kspace 5
Accuracy 5.0

Convergence
   Criterion 1E-6
End

Define
   far=20.0
End

Lattice
   far 0.0 0.0
   0.0 far 0.0
End

Atoms
   H   0.0 0.0 0.0
End

AtomType H
Dirac H
1  0
VALENCE
1S 1
SubEnd

BasisFunctions
1S   1.58
2P   1.0
SubEnd
```

```
FitFunctions
 1S   3.16
 1S   2.09
 1S   1.38
 2S   1.50
 2P   4.00
 2P   2.65
 2P   1.75
 3D   4.00
 3D   2.50
 4F   3.00
 5G   4.00
SubEnd


End


END INPUT
eor
```

13.6.3 Example: Hydrogen on Pt surface

```
Download H_on_Pt.run
```

This example is in many ways similar to the *Pt slab example* (page 111). We refer to the explanations in that example for details. Suffice it to say here that the convergence criteria have again been chosen such that also difficult slab calculations have a good chance to converge. Further, this calculation is once again at the spin-orbit relativistic level. The geometry is such that two layers of Pt are covered by a hydrogen layer.

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title Hydrogen on platinum

Comment
 Technical
   Low quadratic K space integration
   Low real space integration accuracy
 Features
   Lattice   : 2D
   Unit cell : 4 atoms, 1x1
   Basis     : NO+STO
   Options   : Spinorbit ZORA
End

SCF
   Mixing 0.1
   Iterations 100
End

Convergence
   Degenerate default
   Criterion 1e-6
End

DIIS
   NCycleDamp 0
```

```
   DiMix 0.15
End

KSpace 3
Accuracy 3

Relativistic ZORA SPIN

Dependency Basis=1E-8

Define
   latt=7.41
   lvec=latt/SQRT(2.0)
   ysh=lvec/SQRT(3.0)
   dlay=latt/SQRT(3.0)
   height=3.0
End

Lattice
   SQRT(3.0)*lvec/2.0  0.5*lvec
   SQRT(3.0)*lvec/2.0 -0.5*lvec
End

Atoms
   Pt   0.0  0.0   0     :: layer 1
   Pt  -ysh  0.0  -dlay :: layer 2
   H    0.0  0.0   height
End

END INPUT
eor
```

13.6.4 Example: Calculating the atomic forces

```
Download BNForce.run
```

This example shows how to calculate the gradient of the energy with respect to atomic displacements, using the `GRADIENTS` key block. For more details on consult the documentation for *Gradients* (page 47).

```
$ADFBIN/band << eor
Title BN zincblende structure (force calculation)

Gradients
End

NumericalQuality Basic  ! for speed, not very accurate

Units
   Length Angstrom
End

Define
   a = 3.60
End

Lattice
   0.0   0.5*a 0.5*a
```

```
   0.5*a 0.0   0.5*a
   0.5*a 0.5*a 0.0
End

Atoms
   B  0.000000 0.000000 0.000000
   N  0.2404*a 0.2404*a 0.2404*a
end

BasisDefaults
   BasisType TZ2P
   Core Large
End

END INPUT
eor
```

In the output you will find the result

```
FINAL GRADIENTS
   1  B  0.017517  0.017517  0.017517
   2  N -0.017517 -0.017517 -0.017517
```

13.6.5 Example: Optimizing the geometry

`Download H2BulkGeo.run`

This example shows how to optimize the geometry, using the `GEOOPT` key block, including a restart of a previous optimization. More details can be found in the documentation of the *GEOOPT* (page 48) key block.

This example consists of two runs. The first run performs 5 iterations regarding the geometry optimization. And the second run exploits the possibility to restart a geometry optimization based on the *RUNKF* of a previous, presumably non-converged run.

```
$ADFBIN/band << eor
DefaultsConvention 2014

Title H2 bulk geometry optimization

Comment
    Demonstrates the GeoOpt block
    GoeOpt iterations limited to make it faster
End

GeoOpt
  Iterations 5
  Converge Grad=1e-6
End

Units
   Angle Radian
End

Define
   aaa=1
End
```

```
ATOMS
  H    0.0 0.0 0.0
  H    aaa 0.0 0.0
End

Lattice
  5.0*aaa  0         0
  0        5.0*aaa   0
  0        0         5.0*aaa
End

BasisDefaults
   BasisType DZP
End

END INPUT
eor

rm logfile

mv RUNKF H2BulkGeo.runkf
```

Note that the *RUNKF* file is saved and renamed to *H2BulkGeo.runkf*. In the next run we use the result file to continue the geometry optimization.

```
$ADFBIN/band << eor
DefaultsConvention 2014

Title H2 bulk geometry optimization

Comment
    Restarting the geometry optimization
End

Restart
  File H2BulkGeo.runkf
  GeometryOptimization
end

GeoOpt
  Iterations 5
  Converge Grad=1e-6
End

Units
   Angle Radian
End

Define
   aaa=1
End

ATOMS
  H    0.0 0.0 0.0
  H    aaa 0.0 0.0
End

Lattice
```

```
  5.0*aaa  0          0
  0        5.0*aaa  0
  0          0        5.0*aaa
End

BasisDefaults
   BasisType DZP
End

END INPUT
eor
```

13.6.6 Example: Nuclear gradient with a MetaGGA functional

Download CO_MetaGGA_Force.run

This example shows how to calculate the gradient of the energy with respect to atomic displacements for a metaGGA functional, using the *GRADIENTS* (page 47) and *XC* (page 17) key block.

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title CO chain

KSpace 3

accuracy 5

DIIS
   dimix 0.2
   ncycledamp 0
end

SCF
   Mixing 0.3
end

XC
   metaGGA scf force tpss
end

gradients
end

Lattice
   6.00  0.00  0.00
end

Atoms
   C  0.00  0.00  0.00
   O  2.20  0.00  0.00
end

BasisDefaults
   BasisType DZ
   Core Large
end
```

```
end input
eor
```

13.6.7 Example: Frequency run and transition state search

Download H2SlabFreqTS.run

This example demonstrates in the first step how to calculate the Hessian, the vibrational modes and adjacent frequencies using the *RunType* (page 52) key block for an H_2 slab. The second run uses the pre-calculated Hessian and performs a transition state search along the frequency mode with the smallest frequency by using the *RunType* (page 52) key block again.

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title H2 slab frequency calculation near minimum
Comment
    Optimized for speed / Accuracy low
    Basis very small
    Demonstrates the frequencies runtype
End

KSpace 2

Accuracy 4

screening
  RMadel 9
end

Dependency Basis=1e-8

RunType
  Frequencies
End

GeoOpt
  RestartSCF true
End

SCF
  Mixing 0.2
End

Convergence
  Degenerate default
End

DIIS
  NCycleDamp 0
  DiMix 0.5
End

Units
  Length Angstrom
```

```
  Angle Radian
End

ATOMS
:: TS coords
:: 0.408818   -0.059284    0.374229
:: 0.305357    0.059284   -0.374229
:: Local minimum coords:
   H   -0.4  0   0.1
   H    0.4  0  -0.1
End

Lattice
   2.645886  0         0
   0         2.645886  0
End

AtomType H

DIRAC H
 1  0
 VALENCE
 1S 1
SubEnd

BasisFunctions
 1S   0.69

2P   1.25
SubEnd

FitFunctions
 1S   3.16
 1S   2.09
 1S   1.38
 2S   1.50
 2P   4.00
 2P   2.65
 2P   1.75
 3D   4.00
 3D   2.50
SubEnd

End

END INPUT
eor
mv RUNKF H2SlabFreq.runkf

rm Points
```

Here comes the second run where the Hessian (from the frequency run) is used to find the transtition state

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title H2 slab TS search
Comment
```

```
    Optimized for speed / Accuracy low
    Basis very small
    Demonstrates the GeoOpt block
End

KSpace 2

Accuracy 5.0

screening
  RMadel 9
end

Dependency Basis=1e-8

RunType
 TS
End

GeoOpt
  Iterations 50
  Converge Grad=1e-3
  RestartSCF true
  InitialHessian H2SlabFreq.runkf
End

SCF
  Mixing 0.2
End

Convergence
  Degenerate default
End

DIIS
  NCycleDamp 0
  DiMix 0.5
End

Units
  Length Angstrom
  Angle Radian
End

ATOMS
  H   -0.4  0  0.1
  H    0.4  0 -0.1
End

Lattice
  2.645886   0.000000   0.000000
  0.000000   2.645886   0.000000
End

AtomType H

DIRAC H
 1  0
```

```
 VALENCE
 1S 1
SubEnd

BasisFunctions
 1S   0.69
2P   1.25
SubEnd

FitFunctions
 1S   3.16
 1S   2.09
 1S   1.38
 2S   1.50
 2P   4.00
 2P   2.65
 2P   1.75
 3D   4.00
 3D   2.50
SubEnd

End

END INPUT
eor

mv RUNKF H2SlabTS.runkf

rm Points
```

13.7 Time dependent DFT

13.7.1 Example: TD-CDFT for MoS2 Monolayer (NewResponse)

`Download NewResp_2DMoS2Restart.run`

This example demonstrates how to calculate the frequency-dependent dielectric function with the help of the `NewResponse` implementation for a two-dimensional system. (see *NewResponse* (page 56)) Furthermore, the general setup to run the TD-CDFT section as a restart calculation is presented as well. This allows for splitting of the frequency range into several parts, which can then be calculated in separate calculation without the overhead of evaluating the groundstate properties for each of them! Hence, it is a trivial parallelization possibility.

Groundstate Optimization

```
$ADFBIN/band <<EOF
  GeometryFile MoS2_2D_1L.xyz

  Symmetry NoSym

  NumericalQuality basic
  Relativistic ZORA

  KSpace
    Grid 5 5
  End
```

```
    BasisDefaults
      BasisType DZP
      Core      Large
    End

    End input
EOF
mv RUNKF restart.runkf
```

TD-CDFT+Restart Calculation

> **Caution:** One has to make sure to use the same Symmetry/NumericalQuality/KSpace/BasisDefaults/ZORA/...
> options for the groundstate calculation and for the restart calculation! Otherwise a *normal* groundstate SCF opti-
> mization will be performed in the restart calculation.

```
$ADFBIN/band <<EOF
  GeometryFile MoS2_2D_1L.xyz

  Symmetry NoSym

  NumericalQuality basic
  Relativistic ZORA

  KSpace
    Grid 5 5
  End

  BasisDefaults
    BasisType DZP
    Core      Large
  End

  Restart
    SCF
    File restart.runkf
  End

  NewResponse
    nFreq        11
    FreqLow      2.0
    FreqHigh     3.0
    ActiveESpace 10.0
  End

  NewResponseSCF
    nCycle       30
  End

  End input
EOF
mv RUNKF prop.runkf
```

The results are accessible via the standard output or via the prop.runkf file. For the latter, one can use the **AD-
Freport** command `$ADFBIN/adfreport prop.runkf RESPDIELRE` and `$ADFBIN/adfreport prop.`

`runkf RESPDIELIM` to print the components of the dielectric function for the real (*RESPDIELRE*) and imaginary (*RESPDIELIM*) part separately. In the following tables, only the diagonal components are presented:

Table 13.1: **Real part**

Frequency (au)	$\epsilon_1(XX)$	$\epsilon_1(YY)$	$\epsilon_1(ZZ)$
0.0735	8.1622063	8.1788067	1.8845925
0.0772	8.7718566	8.7960299	1.8891231
0.0808	9.6251443	9.6631930	1.8941277
0.0845	10.9457271	11.0126367	1.8996502
0.0882	13.4618956	13.6001321	1.9057858
0.0919	26.5135344	25.9300685	1.9126665
0.0955	6.1134118	4.1756368	1.9204849
0.0992	6.2789015	4.6880515	1.9295347
0.1029	13.7665058	11.5484340	1.9403044
0.1066	-7.2575153	-5.8285172	1.9537079
0.1102	-0.7937277	1.2661253	1.9718981

Table 13.2: **Imaginary part**

Frequency (au)	$\epsilon_2(XX)$	$\epsilon_2(YY)$	$\epsilon_2(ZZ)$
0.0735	0.0015601	0.0015758	0.0000213
0.0772	0.0020566	0.0020839	0.0000200
0.0808	0.0029274	0.0029798	0.0000216
0.0845	0.0047632	0.0048794	0.0000231
0.0882	0.0104743	0.0107877	0.0000246
0.0919	0.2658531	0.1942899	0.0000264
0.0955	12.8856772	14.5286319	0.0000294
0.0992	9.7571573	10.1567455	0.0000338
0.1029	7.5936072	6.7674596	0.0000399
0.1066	13.0264038	9.5897946	0.0000487
0.1102	0.2483041	0.3222301	0.0000676

The more convenient option is to plot the spectral data directly with the help of **ADFspectra**. Just type:

```
$ADFBIN/adfspectra prop.runkf
```

13.7.2 Example: TD-CDFT for bulk silicon (OldResponse)

```
Download Silicon.run
```

The time-dependent current DFT functionality, as implemented by Kootstra, Berger, and Romaniello, enables you to calculate frequency-dependent dielectric functions for 1-dimensional and 3-dimensional periodic systems. In the present example, a standard geometry for bulk Silicon is given. The important part in this example is of course the *OLDRESPONSE* (page 58) key block. It specifies that 7 frequencies shall be computed, with an even spacing between 0.0 eV and 6.8 eV. In this example scalar ZORA relativistic effects are switched on with the `isz` line in the `OLDRESPONSE` key block.

```
$ADFBIN/band << eor
DefaultsConvention pre2014

TITLE Silicon

ACCURACY 5
```
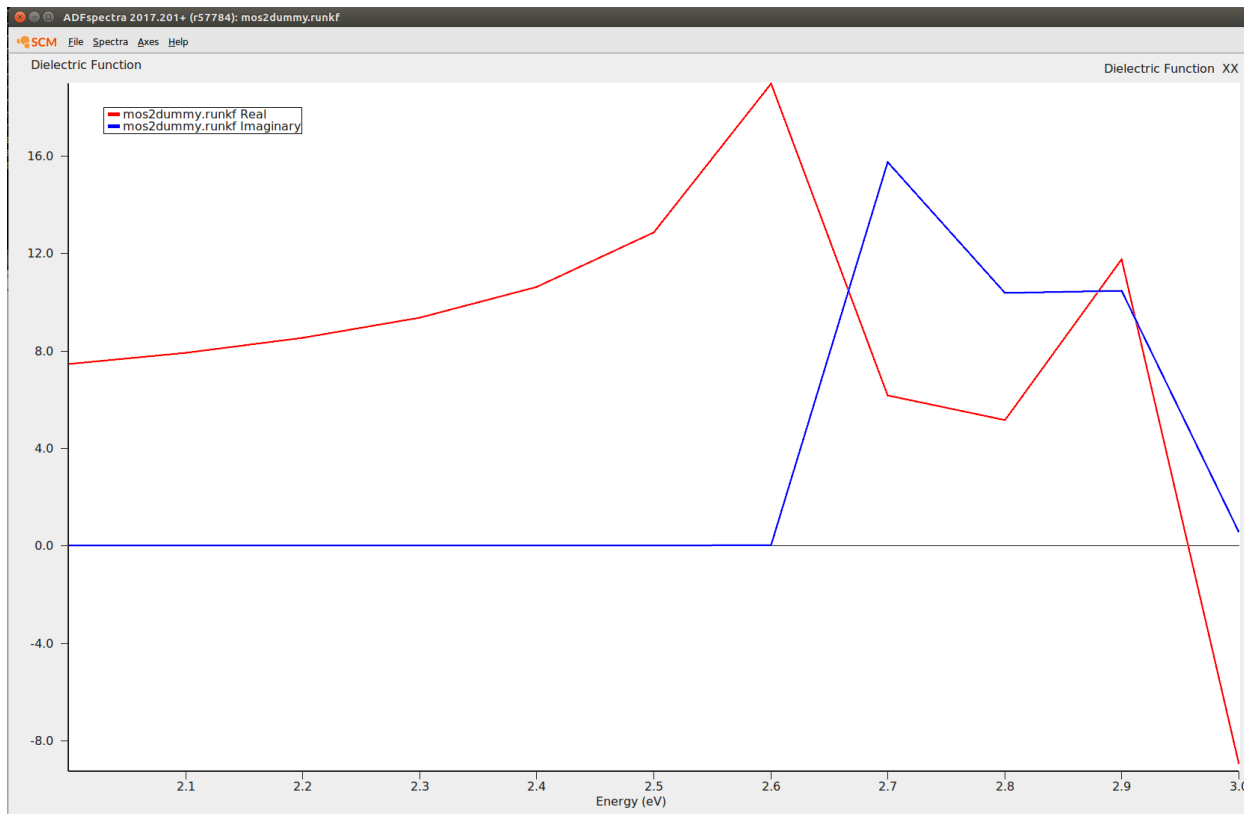
```
KSPACE 2

DEPENDENCY BASIS 1e-10

UNITS
    LENGTH ANGSTROM
END

OLDRESPONSE
    nfreq   7
    strtfr  0.0
    endfr   6.80285
    isz     1
END

DEFINE
    AAA=5.43
    HA=AAA/2
END

LATTICE
    0   HA  HA
    HA  0   HA
    HA  HA  0
END

ATOMS
    Si  0.0   0.0   0.0
```

```
   Si  HA/2  HA/2  HA/2
END

END INPUT
eor
```

For Silicon the real and imaginary parts of the dielectric function:

$$\epsilon(\omega) = 1 + 4\pi\chi(\omega)$$

are calculated.

In the output file, the results will look something like the fragment below. The output specifies for which frequency the dielectric function is determined, and then proceeds to print the values for the 3x3 tensors.

The real and imaginary parts are printed separately. At this frequency, the imaginary part is still zero. Because of the high symmetry of the system, the real part is a constant times the unit matrix except for numerical noise.

```
Frequency     0.833333E-01 au    2.26756     eV
Start the SCF procedure
* Real
Chi_jj   X     -12.8363        0.142802E-18   0.547977E-17
Chi_jj   Y     0.202883E-17   -12.8363        0.121052E-17
Chi_jj   Z     0.124042E-16   0.215311E-17   -12.8363
* Imag
Chi_jj   X     0.000000E+00   0.000000E+00   0.000000E+00
Chi_jj   Y     0.000000E+00   0.000000E+00   0.000000E+00
Chi_jj   Z     0.000000E+00   0.000000E+00   0.000000E+00
*
```

After each frequency has been treated, the results are summarized for each main diagonal component separately in a table. The frequency/energy is again printed in two different units, the Dielectric Function is printed in a.u. The values for Chi, which are trivially related to those printed here, are summarized in a separate table.

```
=================================================================
==         Frequency         ===      Dielectric Function    ==
==     a.u.    ==     e.V.    ===       Re     ==       Im     ==
============XX-dir================================================
   0.416667E-01    1.13378          16.1119        0.000000E+00
   0.833333E-01    2.26756          23.7904        0.000000E+00
   0.125000        3.40134          15.8529        35.8574
   0.166667        4.53512         -3.49949        20.2221
   0.208333        5.66890         -6.60897        12.3661
   0.250000        6.80268         -6.42943        6.87957
============YY-dir================================================
   0.416667E-01    1.13378          16.1119        0.000000E+00
   0.833333E-01    2.26756          23.7904        0.000000E+00
   0.125000        3.40134          15.8529        35.8574
   0.166667        4.53512         -3.49949        20.2221
   0.208333        5.66890         -6.60897        12.3661
   0.250000        6.80268         -6.42943        6.87957
============ZZ-dir================================================
   0.416667E-01    1.13378          16.1119        0.000000E+00
   0.833333E-01    2.26756          23.7904        0.000000E+00
   0.125000        3.40134          15.8529        35.8574
   0.166667        4.53512         -3.49949        20.2221
   0.208333        5.66890         -6.60897        12.3661
   0.250000        6.80268         -6.42943        6.87957
```

Results of the test calculation (red/blue) are plotted in next Figure together with experimental data (yellow/green). The results for the seven specified frequencies are given. It should be obvious that more frequencies are needed (resulting in longer run times) to obtain a smooth curve in which peaks cannot be missed because of too coarse interpolation.
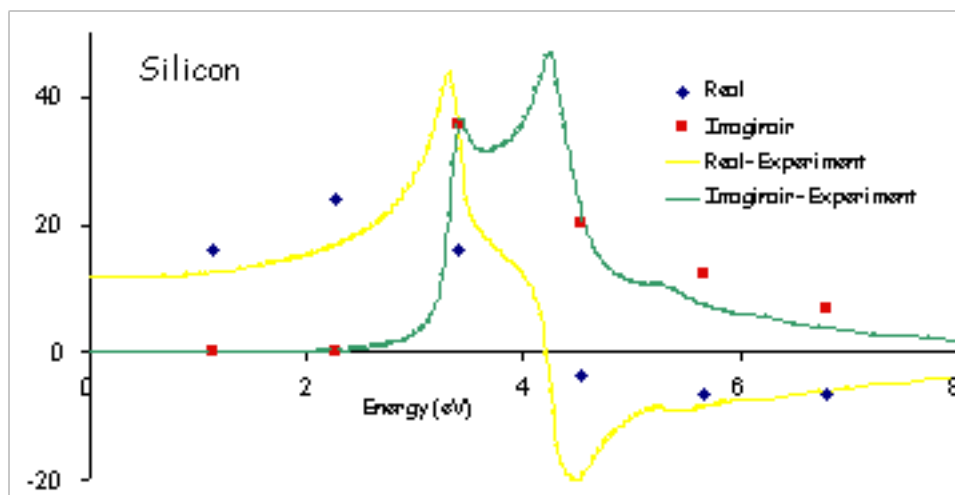
13.7.3 Example: TD-CDFT for bulk helium (OldResponse)

Download He_crystal.run

```
$ADFBIN/band << eor
DefaultsConvention pre2014

TITLE HELIUM CUBIC

ACCURACY 5
KSPACE 3

OLDRESPONSE
   nfreq  10
   strtfr 0.0      :: eV
   endfr  5.44228 :: eV
   ebndtl 1d-2     :: au
END

DEFINE
   HA = 12
END

LATTICE
   HA    0    0
    0   HA    0
    0    0   HA
END

ATOMS
   He   0.0   0.0   0.0
END

BASISDEFAULTS
BASISTYPE V
```

```
END

END INPUT
eor
```

13.7.4 Example: TD-CDFT for bulk diamond (OldResponse)

Download Diamond.run

```
$ADFBIN/band << eor
DefaultsConvention pre2014

TITLE DIAMOND

ACCURACY 5
KSPACE 2

Dependency  Basis=1.e-6

UNITS
   LENGTH ANGSTROM
END

OLDRESPONSE
   nfreq    7
   strtfr   0.0
   endfr    19.0480
END

DEFINE
   AAA=3.57
   HA=AAA/2
END

LATTICE
   0   HA  HA
   HA  0   HA
   HA  HA  0
END

ATOMS
   C  0.00  0.00  0.00
   C  HA/2  HA/2  HA/2
END

BasisDefaults
   BasisType DZ
End

END INPUT
eor
```

13.7.5 Example: TD-CDFT for hydrogen chain (OldResponse)

Download H_chain.run

The input for a one-dimensional system is no different from that for a three-dimensional system. Besides the number of frequencies, and the beginning and end frequency of the frequency range, the *OLDRESPONSE* (page 58) key block should contain stricter than default convergence criteria (`cnvi` and `cnvj`) for the fit coefficients describing the density change.

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title H2-chain

ACCURACY 5
KSPACE 3

DEPENDENCY BASIS 1e-10

OLDRESPONSE
   nfreq  10
   strtfr 0.0
   endfr  0.408171
   cnvi   1d-5
   cnvj   1d-5
END

DEFINE
   HX = 4.5
END

LATTICE
   HX
END

ATOMS
   H    1.0  0.001 0.0
   H   -1.0 -0.001 0.0
END

END INPUT
eor
```

The output for this calculation will give rise to a table like the following one:

```
========================================================================
==         Frequency          ===       Polarizability(xx)      ==
==     a.u.    ==      e.V.    ===       Re      ==      Im      ==
========================================================================
   0.166667E-02   0.453512E-01   127.119         0.00000
   0.333333E-02   0.907023E-01   127.201         0.00000
   0.500000E-02   0.136054       127.338         0.00000
   0.666667E-02   0.181405       127.530         0.00000
   0.833333E-02   0.226756       127.778         0.00000
   0.100000E-01   0.272107       128.083         0.00000
   0.116667E-01   0.317458       128.446         0.00000
   0.133333E-01   0.362809       128.868         0.00000
   0.150000E-01   0.408161       129.351         0.00000

========================================================================
==         Frequency          ===           Chi_JJ (xx)         ==
==     a.u.    ==      e.V.    ===       Re      ==      Im      ==
========================================================================
```

```
   0.00000         0.00000        -2.74118         0.00000
   0.166667E-02    0.453512E-01   -2.74153         0.00000
   0.333333E-02    0.907023E-01   -2.74259         0.00000
   0.500000E-02    0.136054       -2.74436         0.00000
   0.666667E-02    0.181405       -2.74685         0.00000
   0.833333E-02    0.226756       -2.75005         0.00000
   0.100000E-01    0.272107       -2.75399         0.00000
   0.116667E-01    0.317458       -2.75866         0.00000
   0.133333E-01    0.362809       -2.76409         0.00000
   0.150000E-01    0.408161       -2.77028         0.00000
==================================================================
```

13.7.6 Example: TD-CDFT for a metal (OldResponse)

Download Cu_resp_NR_ALDA.run

This example demonstrates how to calculate the dielectric function for a metal. The *OLDRESPONSE* (page 58) key block is as usual.

> **Attention:** The convergence w.r.t. KSPACE parameter should be examined.

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title: OLDRESPONSE of Cu within ALDA

Comment
 Technical
   Hybrid k space integration (3D)
   Reasonable real space integration accuracy
   Definitions of variables
 Features
   Lattice   : 3D
   Unit cell : 1 atom
   Basis     : NO+STO w/ core (DZ/Cu.2p)
   Options   : ALDA
End

kspace  7
Accuracy 4.0

OLDRESPONSE
   nfreq   2
   strtfr  2.72114
   endfr   6.80285
END

Units
   Length Angstrom
End

Define
   halflatt=3.61/2
End
```

```
Lattice :: FCC
   0        halflatt halflatt
   halflatt 0        halflatt
   halflatt halflatt 0
End

Atoms
   Cu  0.00  0.00  0.00
End

BasisDefaults
   BasisType DZ
   Core Large
End

END INPUT
eor
```

13.8 Spectroscopy

13.8.1 Example: Hyperfine A-tensor

Download `TiF3a.run`

Example for an ESR A-tensor calculation.

Be aware that the calculation must be spin unrestricted and the *ATENSOR* (page 60) keyword must be present, too.

```
$ADFBIN/band << eor
DefaultsConvention 2014

Title TiF3

Units
   Length Angstrom
End

Define
   a = 25
   RTIF = 1.78
   RY = RTIF*SQRT(3)/2
End

Unrestricted

EFG
End

ATensor
End

Atoms
   Ti  0.0     0.0  0.0
   F   RTIF    0.0  0.0
   F  -RTIF/2  RY   0.0
```

```
   F  -RTIF/2 -RY   0.0
end

BasisDefaults
   BasisType TZP
   Core None
End

END INPUT
eor
```

13.8.2 Example: Zeeman g-tensor

Download `TiF3g.run`

Example for an ESR g-tensor calculation. More information in the documentation of *ESR* (page 59)

Be aware that this calculation must include the spin-orbit, relativistic approximation (`Relativistic ZORA Spin`)!

```
$ADFBIN/band << eor
DefaultsConvention 2014

Title TiF3

Relativistic ZORA SPIN

esr
end

Units
   Length Angstrom
End

Define
   RTIF = 1.78
   RY  = RTIF*SQRT(3)/2
End

Atoms
   Ti  0.0  0.0    0.0
   F   0.0  RTIF   0.0
   F  -RY  -RTIF/2 0.0
   F   RY  -RTIF/2 0.0
end

BasisDefaults
   BasisType DZ
   Core None
End

END INPUT
eor
```

13.8.3 Example: NMR

```
Download PE-NMR.run
```

With the *NMR* (page 61) key block you can specify for which atom you want the shielding tensor.

Be aware that the NMR option is not implemented for the frozen core approximation, hence one must set option `core` of the `BasisDefaults` key block to *NONE*.

```
$ADFBIN/band << eor
DefaultsConvention pre2014

TITLE PE

NMR
   nmratom=1
   ms0=1.
end

Units
   Length Angstrom
   Angle  Degree
end

xc
   GGA Always Becke Perdew
end

DEPENDENCY BASIS 1e-10

KSPACE 3
Accuracy 5

define
   aCCC = 112.777
   aHCH = 109.47
   alf2 = aCCC/2
   bet  = aHCH/2.
   rcc=1.533
   rch=1.09
   z3 = rch*cos(bet)
   y3 = rch*sin(bet)
   T = 2*rcc*sin(alf2)
   C2x =  rcc*sin(alf2)
   C2z = -rcc*cos(alf2)
end

Lattice
   T         0.        0.
end

Atoms
   C   0.          0.          0.
   C   C2x         0.          C2z
   H   0.           y3         z3
   H   0.          -y3         z3
   H   C2x          y3         C2z-z3
   H   C2x         -y3         C2z-z3
end
```

```
BasisDefaults
   BasisType TZ2P
   Core NONE
End


END INPUT
eor
```

13.8.4 Example: EFG

Download `SnO_EFG.run`

The calculation of the electric field gradient is invoked by the *EFG* (page 60) key block.

Since Sn is quite an heavy atom we use the scalar relativistic option.

```
$ADFBIN/band << eor
DefaultsConvention 2014

Title SnO EFG

NumericalQuality Basic ! Only for speed

Relativistic ZORA

! useful for Moessbauer spectroscopy: density and coulomb pot. at nuclei
PropertiesAtNuclei
End

EFG
End

Units
   Length Angstrom
End

Coordinates Fractional

Define
   aaa=3.8029
   ccc=4.8382
   vvv=0.2369
End

Lattice
   aaa  0.0  0.0
   0.0  aaa  0.0
   0.0  0.0  ccc
End

Atoms
   O   0.0  0.0  0.0
   O   0.5  0.5  0.0
   Sn  0.0  0.5  vvv
   Sn  0.5  0.0  -vvv
End
```

```
BasisDefaults
   BasisType DZ
   Core none
End


End input
eor
```

13.8.5 Example: Phonons

`Download GraphenePhonons.run`

A phonon calculation should be performed at the equilibrium geometry.

In the first calculation we optimize the geometry, including the lattice vectors. We also set the criteria a bit more strict.

```
$ADFBIN/band << eor
Title Graphene geometry optimization

Units
  Length Angstrom
end

! For Graphene we need to use a symmetryc grid
KSpace 5 &
   Grid Symmetric
End

ATOMS
  C   0.0 0.0 0.0
  C   1.23 0.7101408312 0.0
END

Lattice
   2.46 0.000000 0
   1.23 2.130422493 0
End

GeoOpt
  OptimizeLattice true
  Converge grad=1e-5
End

BasisDefaults
  BasisType DZ
end

end input
eor

mv RUNKF Graphene.runkf
```

In the second calculation we use the pre-optimized geometry. (See details of the *Restart* (page 87) key block) Then we define a supercell and perform a phonon run by using the *RunType* (page 52) and *PhononConfig* (page 53) key blocks. Note that `KSpace` can be chosen a bit lower, since we now have a bigger unit cell.

```
$ADFBIN/band << eor
Title Graphene phonon calc

Units
  Length Angstrom
end

KSpace 3 &
   Grid Symmetric
End

Restart
  File Graphene.runkf
  Geometry
End

runtype
  phonons
end

phononConfig
  stepSize 0.0913
  superCell
    2 0
    0 2
  subend
end

ATOMS
  C   0.0 0.0 0.0
  C   1.23 0.7101408312 0.0
END

Lattice
  2.46 0.000000 0
  1.23 2.130422493 0
End

BasisDefaults
  BasisType DZ
end

end input
eor
```

13.9 Analysis

13.9.1 Example: CO absorption on a Cu slab: fragment option and densityplot

`Download Frags_COCu.run`

This example illustrates the usage of *fragments* (page 77) in a BAND calculation for analysis purposes. It takes two runs to do the DOS analysis in a fragment basis, and an extra two runs to get the deformation density with respect to the fragment densities.

The setup involves first the computation of the free CO overlayer, which is to be adsorbed on a Cu surface. To suppress

(most of the) interactions between the CO molecules, i.e. to effectively get the *molecular* CO, the `KSpace` parameter is set to 1 (= no dispersion), and the lattice parameters are set so large that the CO molecules are far apart. The standard result file RUNKF is saved under the name 'CO.runkf'.

```
# ---------------------------- CO molecule -------------------------

$ADFBIN/band << eor
DefaultsConvention pre2014

Title The CO fragment

Comment
 Technical
   Zero order k space integration
   Good real space integration accuracy
   Definitions of variables
 Features
   Lattice   : 2D, large lattice vectors
   Unit cell : 2 atoms, 1x1, quasi molecular
   Basis     : NO+STO w/ core
End

PRINT EIGENS

Kspace   1  ! neglect dispersion
Accuracy   4

Define
  bond=2.18
  far=25
End

Lattice     ! CO molecules far apart
  far  0.0
  0.0  far
End

Atoms
  C   0 0 0
  O   0 0 bond
End

BasisDefaults
BasisType DZ
Core Large
End

End Input
eor

mv RUNKF CO.runkf
```

Now we can use the result file to do a DOS analysis for CO on a copper surface treating the molecule as a fragment. With Fragment%Labels we assign names to the different symmetry orbitals. The Density-of-States analysis details are given with the keys DOS (energy grid, result file with DOS data) and, optionally, GrossPopulations and OverlapPopulations.

```
# ---------------------------- CO + Cu slab --------------------------

$ADFBIN/band << eor
DefaultsConvention pre2014

Title Cu slab with CO adsorbed

Comment
 Technical
   Quadratic K space integration (low)
   Good real space integration accuracy
   Definitions of variables
 Features
   Lattice   : 2D
   Unit cell : 3 atoms, 1x1
   Basis     : NO+STO w/ core
   Options   : Molecular fragment
               Analysis: DOS, PDOS, COOP
End

KSpace 3
Accuracy 4

! fragment specification

Fragment CO.runkf
  1 1
  2 2
Labels ! let us give them some labels
  2Sigma
  2Sigma*
  1Pi_x
  1Pi_y
  3Sigma
  1Pi_x*
  1Pi_y*
  3Sigma*
SubEnd
End

! use fragment basis in dos
DosBas
Fragment 1
End

DOS   ! Analysis
  File  pdos.CO_Cu
  Energies 500
  Min -0.750
  Max 0.300
End

GrossPopulations
  3 2    ! All metal d states
  Sum    ! ALl metal sp states
    3 0
    3 1
  EndSum
```

```
  Frag 1   ! All CO states
  Sum    ! CO 1pi
    FragFun 1 5
    FragFun 1 6
  EndSum
  FragFun 1 7  ! CO 5-sigma
End

OverlapPopulations
  Left   ! Metal d with CO
    3 2
  Right
    Frag 1
End

Define
  dist=3.44
  bond=2.18
End

Lattice
  4.822 0.0
  0.0   4.822
End

Atoms
  C    0 0 dist
  O    0 0 dist+bond
  Cu  0.0   0.0   0.0
End

BasisDefaults
BasisType DZ
Core Large
End

End Input
eor
```

After this run we copy the computed DOS data from the DOS result file to standard output. We also save the restart file for later use.

```
echo Contents of DOS file
cat pdos.CO_Cu

mv RUNKF COCu.runkf
```

Next we want to know the deformation density with respect to the two fragments: 1) The CO molecule and 2) the bare Cu surface. We haven't done the bare Cu surface yet, so that is what happens next.

```
# ---------------------------- Cu slab ------------------------

$ADFBIN/band << eor
DefaultsConvention pre2014

Title Cu slab
```

```
Comment
 Technical
   Quadratic K space integration (low)
   Good real space integration accuracy
   Definitions of variables
 Features
   Lattice   : 2D
   Unit cell : 3 atoms, 1x1
   Basis     : NO+STO w/ core
   Options   :
End

Kspace 3
Accuracy 4

Define
  dist=3.44
  bond=2.18
End

Lattice
  4.822 0.0
  0.0   4.822
End

Atoms
  Cu  0.0   0.0   0.0
End

BasisDefaults
BasisType DZ
Core Large
End

End Input
eor

mv RUNKF Cu.runkf
```

Now we are all set to do our final calculation. We have the two fragment files CO.runkf and Cu.runkf, and the restart file COCu.runkf. Next we want to know the deformation density with respect to the two fragments: 1) The CO molecule and 2) the bare Cu surface. The visualization options like OrbitalPlot and Densityplot require a regular set of points (a grid). Here is how it works

```
# ---------------------------- CO + Cu slab restart --------------------------

$ADFBIN/band -n 1 << eor
DefaultsConvention pre2014

Title Cu slab with CO adsorbed (restart density plot)

Kspace 3
Accuracy 4

Restart
  File COCu.runkf
  DensityPlot
End
```

```
Grid
  Type Coarse
End

DensityPlot
  rho(deformation/fit)
End

! fragment specification

Fragment CO.runkf
  1 1
  2 2
End

Fragment Cu.runkf
  1 3
End

Define
  dist=3.44
  bond=2.18
End

Lattice
  4.822 0.0
  0.0   4.822
End

Atoms
  C   0 0 dist
  O   0 0 dist+bond
  Cu  0.0   0.0   0.0
End

BasisDefaults
BasisType DZ
Core Large
End

End Input
eor
```

This particular restart options does not work in parallel, hence the "-n 1" on the first line.The result of the last run is a
file named TAPE41. Normally you would save that to COCu.t41

```
mv TAPE41 COCu.t41
```

and view it with **ADFview**. On the TAPE41 file are now three fields shown in **ADFview** as

- FITDENSITY_deformation_scf
- FITDENSITY_deformation_scf_frag1
- FITDENSITY_deformation_scf_frag2

being the deformation density of CO+Cu with respect to the atoms, and the same for the two fragments CO and the Cu
slab. In **ADFview** you can add the fields of the two fragments, and then create another field that holds the difference.

13.9.2 Example: CO absorption on a MgO slab: fragment option and PEDA

Download PEDA_MgO+CO.run

This example shall illustrate the use of the *Fragment* (page 77) keywords in combination with the *PEDA* (page 78) keyword to perform the PEDA. For this example two fragment calculations are necessary to calculate the unperturbed eigensystems of the MgO slab and CO fragment.

Fragment calculations

```
# ---------------------------- MgO slab -------------------------

$ADFBIN/band << eor
Title MgO surface

KSpace
  Grid 3 3
End

BeckeGrid
   quality basic
End

XC
   GGA PBE
End

Units
   length Angstrom
end

Atoms
   Mg  0.00000000      0.00000000      0.00000000
   Mg  1.50260191     -1.50260191     -2.12400000
   Mg  0.00000000      0.00000000     -4.24800000
   Mg  3.00520382      0.00000000      0.00000000
   Mg  1.50260191      1.50260191     -2.12400000
   Mg  3.00520382      0.00000000     -4.24800000
   O   1.50260191     -1.50260191      0.00200000
   O   0.00000000      0.00000000     -2.12400000
   O   1.50260191     -1.50260191     -4.25000000
   O   1.50260191      1.50260191      0.00200000
   O   3.00520382      0.00000000     -2.12400000
   O   1.50260191      1.50260191     -4.25000000
End

Lattice
     3.00520382     -3.00520382      0.00000000
     3.00520382      3.00520382      0.00000000
End

BasisDefaults
   BasisType TZP
   Core small
End

END INPUT
```

```
eor

mv RUNKF MgO.runkf
```

```
# --------------------------- CO fragment -----------------------

$ADFBIN/band << eor
Title CO molecule

KSpace
  Grid 3 3
End

BeckeGrid
  quality basic
End

XC
  GGA PBE
End

Units
   length Angstrom
end

Atoms
   C   0.00000000       0.00000000       2.61000000
   O   0.00000000       0.00000000       3.73700000
End

Lattice
      3.00520382      -3.00520382       0.00000000
      3.00520382       3.00520382       0.00000000
End

BasisDefaults
   BasisType TZP
   Core small
End

END INPUT
eor

mv RUNKF CO.runkf
```

PEDA calculation

The two result files, MgO.runkf and CO.runkf, can now be used to perform the PEDA. Here, the mapping of the atoms of the PEDA calculation and the fragment calculations is necessary. And the used grid points in reciprocal space have to be identical in all three calculations.

```
# --------------------------- PEDA calculation -----------------------

$ADFBIN/band << eor
Title PEDA
```

```
KSpace
   Grid 3 3
End

BeckeGrid
   quality basic
End

XC
   GGA PBE
End

fragment MgO.runkf
   1  1
   2  2
   3  3
   4  4
   5  5
   6  6
   7  7
   8  8
   9  9
   10 10
   11 11
   12 12
end

fragment CO.runkf
   2 13
   1 14
end

PEDA

units
   length Angstrom
end

Atoms
   Mg.frag_MgO  0.00000000       0.00000000       0.00000000
   Mg.frag_MgO  1.50260191      -1.50260191      -2.12400000
   Mg.frag_MgO  0.00000000       0.00000000      -4.24800000
   Mg.frag_MgO  3.00520382       0.00000000       0.00000000
   Mg.frag_MgO  1.50260191       1.50260191      -2.12400000
   Mg.frag_MgO  3.00520382       0.00000000      -4.24800000
   O.frag_MgO   1.50260191      -1.50260191       0.00200000
   O.frag_MgO   0.00000000       0.00000000      -2.12400000
   O.frag_MgO   1.50260191      -1.50260191      -4.25000000
   O.frag_MgO   1.50260191       1.50260191       0.00200000
   O.frag_MgO   3.00520382       0.00000000      -2.12400000
   O.frag_MgO   1.50260191       1.50260191      -4.25000000
   O.frag_CO    0.00000000       0.00000000       3.73700000
   C.frag_CO    0.00000000       0.00000000       2.61000000
End

Lattice
       3.00520382      -3.00520382       0.00000000
       3.00520382       3.00520382       0.00000000
```

```
End

BasisDefaults
   BasisType TZP
   Core small
End

END INPUT
eor
```

In the output file the results can be found in the PEDA block after the *Energy Analysis*.

13.9.3 Example: CO absorption on a MgO slab: fragment option, PEDA and PEDANOCV

Download PEDANOCV_MgO+CO.run

This example shall illustrate the use of the *Fragment* (page 77) keywords in combination with the *PEDA* (page 78) and *PEDANOCV* (page 79) keywords to perform the PEDANOCV calculation. For this example two fragment calculations are necessary to calculate the unperturbed eigensystems of the MgO slab and CO fragment. Here, the sampling of the reciprocal space is restricted to Γ point.

Fragment calculations

```
!----------------------------- MgO slab -----------------------

$ADFBIN/band << eor
Title MgO fragment

KSpace
   Grid 1 1
end

BeckeGrid
   quality basic
End

relativistic zora

XC
   GGA PBE
End

Units
   length Angstrom
end

Atoms
   Mg  0.00000000      0.00000000       0.00000000
   Mg  1.50260191     -1.50260191      -2.12400000
   Mg  0.00000000      0.00000000      -4.24800000
   Mg  3.00520382      0.00000000       0.00000000
   Mg  1.50260191      1.50260191      -2.12400000
   Mg  3.00520382      0.00000000      -4.24800000
   O   1.50260191     -1.50260191       0.00200000
```

```
   O    0.00000000       0.00000000      -2.12400000
   O    1.50260191      -1.50260191      -4.25000000
   O    1.50260191       1.50260191       0.00200000
   O    3.00520382       0.00000000      -2.12400000
   O    1.50260191       1.50260191      -4.25000000
End

Lattice
     3.00520382      -3.00520382       0.00000000
     3.00520382       3.00520382       0.00000000
End

BasisDefaults
   BasisType TZP
   Core none
End

END INPUT
eor

mv RUNKF MgO.runkf
```

```
!---------------------------- CO fragment ------------------------

$ADFBIN/band << eor
Title CO fragment

KSpace
   Grid 1 1
End

BeckeGrid
   quality basic
End

relativistic zora

XC
   GGA PBE
End

Units
   length Angstrom
end

Atoms
   C    0.00000000       0.00000000       2.61000000
   O    0.00000000       0.00000000       3.73700000
End

Lattice
     3.00520382      -3.00520382       0.00000000
     3.00520382       3.00520382       0.00000000
End

BasisDefaults
   BasisType TZP
   Core none
```

```
End

END INPUT
eor

mv RUNKF CO.runkf
```

PEDANOCV calculation

The two result files, MgO.runkf and CO.runkf, can now be used to perform the PEDANOCV. Here, the mapping of the atoms of the PEDA calculation and the fragment calculations is necessary. And the used grid points in reciprocal space have to be identical in all three calculations - in this case the Γ point for all calculations.

```
!---------------------------- PEDANOCV calculation -----------------------

$ADFBIN/band << eor
Title Mg+CO

KSpace
   Grid 1 1
End

BeckeGrid
   quality basic
End

relativistic zora

XC
   GGA PBE
End

fragment MgO.runkf
    1  1
    2  2
    3  3
    4  4
    5  5
    6  6
    7  7
    8  8
    9  9
   10 10
   11 11
   12 12
end

fragment CO.runkf
    1  13
    2  14
end

PEDA

PEDANOCV
   EigvalThresh 0.001
```

```
End

units
   length Angstrom
end

Atoms
   Mg.frag_MgO  0.00000000      0.00000000      0.00000000
   Mg.frag_MgO  1.50260191     -1.50260191     -2.12400000
   Mg.frag_MgO  0.00000000      0.00000000     -4.24800000
   Mg.frag_MgO  3.00520382      0.00000000      0.00000000
   Mg.frag_MgO  1.50260191      1.50260191     -2.12400000
   Mg.frag_MgO  3.00520382      0.00000000     -4.24800000
   O.frag_MgO   1.50260191     -1.50260191      0.00200000
   O.frag_MgO   0.00000000      0.00000000     -2.12400000
   O.frag_MgO   1.50260191     -1.50260191     -4.25000000
   O.frag_MgO   1.50260191      1.50260191      0.00200000
   O.frag_MgO   3.00520382      0.00000000     -2.12400000
   O.frag_MgO   1.50260191      1.50260191     -4.25000000
   C.frag_CO    0.00000000      0.00000000      2.61000000
   O.frag_CO    0.00000000      0.00000000      3.73700000
End

Lattice
      3.00520382     -3.00520382      0.00000000
      3.00520382      3.00520382      0.00000000
End

BasisDefaults
   BasisType TZP
   Core none
End

END INPUT
eor
mv RUNKF decomp.runkf
```

In the output file the results can be found in the PEDANOCV block after the Energy Analysis and PEDA block.

The NOCV orbitals and NOCV deformation densities can be visualized using **ADFview** or by a restart calculation. In the latter case, one adds the *Restart* (page 87) block key with the options `File decomp.runkf` and the `NOCVdRhoPlot` and `NOCVOrbitalPlot` keys. These will trigger the calculation of the plot properties. To specify which NOCV deformation densities and NOCV orbitals are plotted, one adds the *NOCVdRhoPlot* (page 90) and *NOCVOrbitalPlot* (page 89) block key. In both blocks the line `1 Band 1 5` means, that for k-point 1 the densities/orbitals 1 to 5 are calculated.

```
$ADFBIN/band << eor
Title Restart Calculation

Restart
   File decomp.runkf
   NOCVdRhoPlot
   NOCVOrbitalPlot
End

NOCVdRhoPlot
   1 Band 1 5
End
```

```
NOCVOrbitalPlot
   1 Band 1 5
End

Grid
   Type coarse
End

KSpace
   Grid 1 1
End

BeckeGrid
   quality basic
End

relativistic zora

XC
   GGA PBE
End

units
   length Angstrom
end

Atoms
   Mg  0.00000000      0.00000000       0.00000000
   Mg  1.50260191     -1.50260191      -2.12400000
   Mg  0.00000000      0.00000000      -4.24800000
   Mg  3.00520382      0.00000000       0.00000000
   Mg  1.50260191      1.50260191      -2.12400000
   Mg  3.00520382      0.00000000      -4.24800000
   O   1.50260191     -1.50260191       0.00200000
   O   0.00000000      0.00000000      -2.12400000
   O   1.50260191     -1.50260191      -4.25000000
   O   1.50260191      1.50260191       0.00200000
   O   3.00520382      0.00000000      -2.12400000
   O   1.50260191      1.50260191      -4.25000000
   C   0.00000000      0.00000000       2.61000000
   O   0.00000000      0.00000000       3.73700000
End

Lattice
      3.00520382     -3.00520382       0.00000000
      3.00520382      3.00520382       0.00000000
End

BasisDefaults
   BasisType TZP
   Core none
End

debug BlockPropertyModule

END INPUT
eor
```

The important output of this calculation is the TAPE41 file. Renaming it to foobar.t41 will allow **ADFview** to read and interpret the data stored on this file.

13.9.4 Example: Cu_slab: 2-dim. Finite temperature and orbital plot

```
Download Cu_slab.run
```

A two-dimensional infinite (periodic boundary conditions) slab calculation is performed for Cu. The dimensionality is simply defined by the number of records in the *Lattice* (page 12) data block. In a 2-dimensional calculation the lattice vectors are put in the xy-plane.

Slab calculations for metals frequently suffer from SCF convergence problems, as a result of the open valence band(s). To help the program converge it is often useful or even necessary to use some special features, such as the *ElectronicTemperature* (page 42) key. This particular key requires a numerical value (0.025 in the example) and implies that a finite-temperature electronic distribution is used, rather than a sharp cut-off at the Fermi level. The numerical value is the applied energy width, in Hartree units.

The so-modified electronic distribution also affects the energy. The 'true' zero-T energy is computed, approximately, by an interpolation formula. The interpolation is not very accurate and one should try to use as small as possible values for the *ElectronicTemperature* (page 42) key so as to avoid increasing uncertainty in the results. The program prints, in the energy section of the output file, the finite-T correction term that has been applied through the interpolation formula. This gives at least an indication of any remaining uncorrected deviation of the outcome from a true zero-T calculation.

In the second run the *RUNKF* file of the first run is used to do an orbital plot restart. Normally you would rename the resulting *TAPE41* to 'myslab.t41' and watch the orbitals with **ADFview**.

```
# -------------------------- first run -------------------------

$ADFBIN/band << eor
Title Cu slab

Comment
 Technical
   Quadratic K space integration
   Good real space integration accuracy
 Features
   Lattice   : 2D
   Unit cell : 1 atom, 1x1
   Basis     : NO+STO w/ core
   Options   : ElectronicTemperature (temperature effect)
End

Kspace 5
Accuracy 4

Convergence
   ElectronicTemperature  0.025
End

Lattice
   4.822 0.0
   0.0   4.822
End

Atoms
   Cu   0.0   0.0   0.0
End
```

```
BasisDefaults
   BasisType DZ
End

EndInput
eor

mv RUNKF CuSlab.runkf
rm Points

# ---------------------------- orbital plot --------------------------
export NSCM=1

$ADFBIN/band -n 1 << eor
Title Cu slab orbital plot

Comment
 Technical
   Quadratic K space integration
   Good real space integration accuracy
 Features
   Lattice   : 2D
   Unit cell : 1 atom, 1x1
   Basis     : NO+STO w/ core
   Options   : ElectronicTemperature (temperature effect)
End

Kspace 5
Accuracy 4

Restart
   File CuSlab.runkf
   OrbitalPlot
End

Grid
   Type Coarse
End

OrbitalPlot
   1 Band 2 4 ! k-point 1, bands 2 to 4
   3  -0.1 0.1 ! k-point 3 orbitals within 0.1 Hartree from Fermi Level
End

Convergence
   ElectronicTemperature  0.025
End

Lattice
   4.822  0.000
   0.000  4.822
End

Atoms
   Cu  0.0   0.0   0.0
End
```

```
BasisDefaults
   BasisType DZ
End

EndInput
eor

echo "Begin TOC of tape41"

$ADFBIN/dmpkf -n 1 TAPE41 --toc

echo "End TOC of tape41"
```

13.9.5 Example: Bader analysis

```
Download Li2O_Bader.run
```

To get the Quantum Theory of Atoms In Molecules and Crystals (QT-AIMAC) analysis use the *GridBasedAIM* (page 77) block key.

The grid-based AIM method is very fast, but a bit inaccurate. Hence, on has to make sure that the results are converged w.r.t. the real-space integration grid. (see eg. *NumericalQuality* (page 31) or *Accuracy* (page 39))

```
$ADFBIN/band << eor
DefaultsConvention pre2014

Title Li2O bulk (fluorite structure)

KSpace 3

accuracy 4

Integration
   accsph 6
   accpyr 6
end

GridBasedAIM
End

Dependency basis=1e-9 fit=1e-8

tails bas=0 core=0 fit=0

DIIS
   dimix 0.2
   ncycledamp 0
end

scf
   mixing 0.4
end

xc
   gga scf bp86
end
```

```
define
   ha=8.73/2
   qa=8.73/4
end

Lattice
   0  ha ha
   ha 0  ha
   ha ha 0
end

Atoms
   O   0.0   0.0  0.0
   Li  qa    qa   qa
   Li  3*qa  qa   qa
end

BasisDefaults
   BasisType TZ2P
   Core small
end

end input
eor
```

13.9.6 Example: Properties at nuclei

Download PropertiesAtNuclei.run

One can obtain the values of some properties near the nucleus. (see *PropertiesAtNuclei* (page 61))

Note: Instead of calculating the properties at a point in space an average is taken over a tiny sphere around this point.

```
$ADFBIN/band << EOF
DefaultsConvention pre2014

Title O2

PropertiesAtNuclei
   vxc[rho(fit)]
   rho(fit)
   rho(scf)
   v(coulomb/scf)
   rho(deformation/fit)
   rho(deformation/scf)
End

unrestricted

Accuracy 6

UNITS
   length Angstrom
   angle Degree
END

Atoms
```

```
   O   0.000  0.000  0.000
   O   0.000  0.000  1.208
end

BasisDefaults
   BasisType DZ
   Core None
End

XC
   gga always pbe
END

end input
EOF
```

13.9.7 Example: Band structure plot

Download Li_BZPlot.run

First run: automatic generation of band structure data (see *BZStruct* (page 75)) on the result file (to be viewed with the GUI). We use a little bit of interpolation for smoothened curves.

```
# first plot: automatic with interpolation

$ADFBIN/band << eor

Title Li Bulk

NumericalQuality Basic

units
   length angstrom
end

ATOMS
   Li 0.0 0.0 0.0
END

BZStruct
   Enabled   true
   Automatic true
   FatBands  false
end

Lattice
 -1.745  1.745  1.745
  1.745 -1.745  1.745
  1.745  1.745 -1.745
End

BasisDefaults
   BasisType DZ
   Core Large
end
```

```
EndInput
eor
```

Second run: specifying the path through the BZ zone by hand. We set automatic to false and then specify the path with the *BZPath* (page 76) key block, using one or more path subkeys. Here, the second run will produce exactly the same path as the automatic one.

```
# second plot: user specified

$ADFBIN/band << eor

Title Li Bulk

NumericalQuality Basic

units
   length angstrom
end

ATOMS
   Li 0.0 0.0 0.0
END

BZStruct
   Enabled   true
   Automatic false
   FatBands  false
end

bzpath
   path
      0.00  0.00  0.00  G
      0.50 -0.50  0.50  H
      0.00  0.00  0.50  N
      0.00  0.00  0.00  G
      0.25  0.25  0.25  P
      0.50 -0.50  0.50  H
   subend
   path
      0.25  0.25  0.25  P
      0.00  0.00  0.50  N
   subend
end

Lattice
  -1.745  1.745  1.745
   1.745 -1.745  1.745
   1.745  1.745 -1.745
End

BasisDefaults
   BasisType DZ
   Core Large
end

eor
```

13.9.8 Example: Effective Mass (electron mobility)

```
Download EffectiveMass.run
```

An effective mass calculation is about the curvature of band at the top of the valence band and the bottom of the conduction band. This is obtained by numerical differentiation.

It can be done for systems with 1D, 2D, or 3D translational symmetry.

The easiest way to use this feature is to specify an empty *EffectiveMass* (page 61) key block (so leave out the `NumAbove`, `NumBelow`, and `UniqueKPoints`).

Example 1D:

```
"$ADFBIN/band"  << eor
DefaultsConvention 2014

TITLE 1D Al Chain

EffectiveMass
   StepSize 0.001
   NumAbove 4
   NumBelow 2
   UniqueKPoints 1 2 3
End

UNITS
   length Angstrom
   angle Degree
END

ATOMS
   Al  0.0 0.0 0.0
END

Lattice
   2.12440502 -0.000000 0.0
End

BasisDefaults
   BasisType DZ
   Core Large
End

XC
   LDA SCF VWN
END

end input
eor

rm -f RUNKF
```

Example 2D:

```
"$ADFBIN/band"  << eor
DefaultsConvention 2014

TITLE MoS2Slab
```

```
relativistic zora

EffectiveMass
End

UNITS
   length Angstrom
   angle Degree
END

Atoms Mo
   Mo     -1.626960686      0.313108730      0.000000000
   S       0.000000000      1.252434919      1.547040825
   S       0.000000000      1.252434919     -1.547040825
End

Lattice
           1.626960686     -2.817978569      0.000000000
           1.626960686      2.817978569      0.000000000
End

BasisDefaults
   BasisType DZ
   Core Large
End

end input
eor

rm -f RUNKF
```

Example 3D:

```
"$ADFBIN/band"  << eor

DefaultsConvention 2014

NumericalQuality Basic

KSpace
   Quality Normal
End

TITLE ZnO

Tails bas=1e-8

EffectiveMass
   NumAbove 2
   NumBelow 2
End

UNITS
    length Angstrom
    angle Degree
END

ATOMS
```

```
   Zn   1.625 0.9381941876 0.0
   Zn   1.625 -0.9381941878 2.615
   O    1.625 0.9381941876 1.96125
   O    1.625 -0.9381941878 4.57625
END

Lattice
   1.625 -2.814582562 0.000000
   1.625 2.814582562 0.000000
   0.000000 0.000000 5.23
End

BasisDefaults
   BasisType DZ
   Core Large
End


end input
eor
```

13.9.9 Example: Generating an Excited State with and Electron Hole

```
Download Si_ElectronHole.run
```

There is the possibility define the excitation of an electron from a low lying, localized band to a virtual band. The *ElectronHole* (page 96) key does allow the specification of the original band and the spin of the electron. The *En-forcedSpinPolarization* (page 96) key allows to restrict the spin polarization of the whole system.

```
"$ADFBIN/band" <<eor

UNITS
    length Angstrom
    angle Degree
END

Atoms
  Si.frozen_core -0.67875 -0.67875 -0.67875
  Si              0.67875  0.67875  0.67875
End

BasisDefaults
  BasisType DZP
  core none
  Si.frozen_core  BasisType=DZP core=Large
End

Lattice
    0.000  2.715  2.715
    2.715  0.000  2.715
    2.715  2.715  0.000
End

XC
   LDA SCF VWN
END
```

```
UNRESTRICTED

ElectronHole
    BandIndex 1
    SpinIndex 1
End

EnforcedSpinPolarization 0

end input
eor
```

13.10 List of Examples

- *Restart a SCF* (page 128)
- *Silicon* (page 163)
- *Si_ElectronHole* (page 197)
- *SnO_EFG* (page 173)
- *TiF3a* (page 170)
- *TiF3g* (page 171)
- *ZnS_ModelPotential* (page 120)

REQUIRED CITATIONS

When you publish results in the scientific literature which were obtained with programs of the ADF package, you are required to include references to the program package with the appropriate release number, and a few key publications.

In addition references to special features are mandatory, in case you have used them.

14.1 General References

For calculations with the periodic structures BAND program, version 2017:

1. G. te Velde and E.J. Baerends, *Precise density-functional method for periodic structures*, Physical Review B 44, 7888 (1991) (http://dx.doi.org/10.1103/PhysRevB.44.7888).

2. G. Wiesenekker and E.J. Baerends, *Quadratic integration over the three-dimensional Brillouin zone*, Journal of Physics: Condensed Matter 3, 6721 (1991) (http://dx.doi.org/10.1088/0953-8984/3/35/005).

3. M. Franchini, P.H.T. Philipsen, L. Visscher, *The Becke Fuzzy Cells Integration Scheme in the Amsterdam Density Functional Program Suite*, Journal of Computational Chemistry 34, 1818 (2013) (http://dx.doi.org/10.1002/jcc.23323).

4. M. Franchini, P.H.T. Philipsen, E. van Lenthe, L. Visscher, *Accurate Coulomb Potentials for Periodic and Molecular Systems through Density Fitting*, Journal of Chemical Theory and Computation 10, 1994 (2014) (http://dx.doi.org/10.1021/ct500172n).

5. BAND2017, SCM, Theoretical Chemistry, Vrije Universiteit, Amsterdam, The Netherlands, http://www.scm.com Optionally, you may add the following list of authors and contributors: P.H.T. Philipsen, G. te Velde, E.J. Baerends, J.A. Berger, P.L. de Boeij, M. Franchini, J.A. Groeneveld, E.S. Kadantsev, R. Klooster, F. Kootstra, P. Romaniello, M. Raupach, D.G. Skachkov, J.G. Snijders, C.J.O. Verzijl, J.A. Celis Gil, J. M. Thijssen, G. Wiesenekker, T. Ziegler.

Note: if you have used a modified (by yourself, for instance) version of the code, you should mention in the citation that a modified version has been used.

14.2 Feature References

Lead See key references above, for all work with BAND

Suggested G. Wiesenekker, G. te Velde and E.J. Baerends, *Analytic quadratic integration over the two-dimensional Brillouin zone*, Journal of Physics C: Solid State Physics 21, 4263 (1988) (http://dx.doi.org/10.1088/0022-3719/21/23/012).

14.2.1 Geometry optimization

Lead E.S. Kadantsev, R. Klooster. P.L. de Boeij and T. Ziegler, *The Formulation and Implementation of Analytic Energy Gradients for Periodic Density Functional Calculations with STO/NAO Bloch Basis Set*, Molecular Physics 105, 2583 (2007) (http://dx.doi.org/10.1080/00268970701598063).

14.2.2 TDDFT

Lead F. Kootstra, P.L. de Boeij and J.G. Snijders, *Efficient real-space approach to time-dependent density functional theory for the dielectric response of nonmetallic crystals*, Journal of Chemical Physics 112, 6517 (2000) (http://dx.doi.org/10.1063/1.481315).

P. Romaniello and P.L. de Boeij, *Time-dependent current-density-functional theory for the metallic response of solids*, Physical Review B 71, 155108 (2005) (http://dx.doi.org/10.1103/PhysRevB.71.155108).

Main applications F. Kootstra, P.L. de Boeij, and J.G. Snijders, *Application of time-dependent density-functional theory to the dielectric function of various nonmetallic crystals*, Physical Review B 62, 7071 (2000) (http://dx.doi.org/10.1103/PhysRevB.62.7071).

P. Romaniello, P.L. de Boeij, F. Carbone, and D. van der Marel, *Optical properties of bcc transition metals in the range 0 - 40 eV*, Physical Review B 73, 075115 (2006) (http://dx.doi.org/10.1103/PhysRevB.73.075115).

Suggested book references F. Kootstra, Ph.D. thesis (http://www.scm.com/Doc/ft439.pdf), Rijksuniversiteit Groningen, Groningen (2001).

P. Romaniello, Ph.D. thesis (http://www.scm.com/Doc/Thesis_Pina.pdf), Rijksuniversiteit Groningen, Groningen (2006).

A. Berger, Ph.D. thesis (http://www.scm.com/Doc/Thesis_Arjan.pdf), Rijksuniversiteit Groningen, Groningen (2006).

14.2.3 Relativistic TDDFT

Lead P. Romaniello and P.L. de Boeij, *Relativistic two-component formulation of time-dependent current-density functional theory: Application to the linear response of solids*, Journal of Chemical Physics 127, 174111 (2007) (http://dx.doi.org/10.1063/1.2780146).

14.2.4 Vignale Kohn

Lead J.A. Berger, P.L. de Boeij and R. van Leeuwen, *Analysis of the viscoelastic coefficients in the Vignale-Kohn functional: The cases of one- and three-dimensional polyacetylene*, Physical Review B 71, 155104 (2005) (http://dx.doi.org/10.1103/PhysRevB.71.155104).

Applications J.A. Berger, P. Romaniello, R. van Leeuwen and P.L. de Boeij, *Performance of the Vignale-Kohn functional in the linear response of metals*, Physical Review B 74, 245117 (2006) (http://dx.doi.org/10.1103/PhysRevB.74.245117).

J.A. Berger, P.L. de Boeij, and R. van Leeuwen, *Analysis of the Vignale-Kohn current functional in the calculation of the optical spectra of semiconductors*, Physical Review B 75, 35116 (2007) (http://dx.doi.org/10.1103/PhysRevB.75.035116).

14.2.5 NMR

Lead D. Skachkov, M. Krykunov, E. Kadantsev, and T. Ziegler, *The Calculation of NMR Chemical Shifts in Periodic Systems Based on Gauge Including Atomic Orbitals and Density Functional Theory*, Journal of Chemical Theory and Computation 6, 1650 (2010) (http://dx.doi.org/10.1021/ct100046a)

D. Skachkov, M. Krykunov, and T. Ziegler, *An improved scheme for the calculation of NMR chemical shifts in periodic systems based on gauge including atomic orbitals and density functional theory*, Canadian Journal of Chemistry 89, 1150 (2011) (http://dx.doi.org/10.1139/v11-050).

14.2.6 ESR

A-tensor: Nuclear magnetic dipole hyperfine interaction E.S. Kadantsev and T. Ziegler, *Implementation of a Density Functional Theory-Based Method for the Calculation of the Hyperfine A-tensor in Periodic Systems with the Use of Numerical and Slater Type Atomic Orbitals: Application to Paramagnetic Defects*, Journal of Physical Chemistry A 112, 4521 (2008) (http://dx.doi.org/10.1021/jp800494m).

G-tensor: Zeeman interaction E.S. Kadantsev and T. Ziegler, *Implementation of a DFT Based Method for the Calculation of Zeeman g-tensor in Periodic Systems with the use of Numerical and Slater Type Atomic Orbitals*, Journal of Physical Chemistry A 113, 1327 (2009) (http://dx.doi.org/10.1021/jp805466c).

14.2.7 NEGF

Lead

3. (a) O. Verzijl and J. M. Thijssen *DFT-Based Molecular Transport Implementation in ADF/BAND*, J. Phys. Chem. C, 2012, 116 (46), pp 24393–24412 (http://dx.doi.org/10.1021/jp3044225).

14.3 External programs and Libraries

Click here for the list of programs and/or libraries used in the ADF package. On some platforms optimized libraries have been used and/or vendor specific MPI implementations.

REFERENCES

1. S.H. Vosko, L. Wilk and M. Nusair, *Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis.* Canadian Journal of Physics 58, 1200 (1980) (http://dx.doi.org/10.1139/p80-159).

2. H. Stoll, C.M.E. Pavlidou and H. Preuß, *On the calculation of correlation energies in the spin-density functional formalism.* Theoretica Chimica Acta 49, 143 (1978) (http://dx.doi.org/10.1007/PL00020511).

3. A.D. Becke, *Density-functional exchange-energy approximation with correct asymptotic behavior.* Physical Review A 38, 3098 (1988) (http://dx.doi.org/10.1103/PhysRevA.38.3098).

4. J.P. Perdew and Y. Wang, *Accurate and simple density functional for the electronic exchange energy: generalized gradient approximation.* Physical Review B 33, 8800 (1986) (http://dx.doi.org/10.1103/PhysRevB.33.8800).

5. J.P. Perdew, J.A. Chevary, S.H. Vosko, K.A. Jackson, M.R. Pederson, D.J. Singh and C. Fiolhais, *Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation.* Physical Review B 46, 6671 (1992) (http://dx.doi.org/10.1103/PhysRevB.46.6671).

6. J.P. Perdew, *Density-functional approximation for the correlation energy of the inhomogeneous electron gas.* Physical Review B 33, 8822 (1986) (http://dx.doi.org/10.1103/PhysRevB.33.8822).

7. C. Lee, W. Yang and R.G. Parr, *Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density.* Physical Review B 37, 785 (1988) (http://dx.doi.org/10.1103/PhysRevB.37.785).

8. B.G. Johnson, P.M.W. Gill and J.A. Pople, *The performance of a family of density functional methods.* Journal of Chemical Physics 98, 5612 (1993) (http://dx.doi.org/10.1063/1.464906).

9. T.V. Russo, R.L. Martin and P.J. Hay, *Density Functional calculations on first-row transition metals.* Journal of Chemical Physics 101, 7729 (1994) (http://dx.doi.org/10.1063/1.468265).

10. R. van Leeuwen and E.J. Baerends, *Exchange-correlation potential with correct asymptotic behavior.* Physical Review A 49, 2421 (1994) (http://dx.doi.org/10.1103/PhysRevA.49.2421).

11. R. Neumann, R.H. Nobes and N.C. Handy, *Exchange functionals and potentials.* Molecular Physics 87, 1 (1996) (http://dx.doi.org/10.1080/00268979600100011).

12. J.P. Perdew, K. Burke and M. Ernzerhof, *Generalized Gradient Approximation Made Simple.* Physical Review Letters 77, 3865 (1996) (http://dx.doi.org/10.1103/PhysRevLett.77.3865).

13. B. Hammer, L.B. Hansen, and J.K.Nørskov, *Improved adsorption energetics within density-functional theory using revised Perdew-Burke-Ernzerhof functionals.* Physical Review B 59, 7413 (1999) (http://dx.doi.org/10.1103/PhysRevB.59.7413).

14. J.P. Perdew, A. Ruzsinszky, G.I. Csonka, O.A. Vydrov, G.E. Scuseria, L.A. Constantin, X. Zhou and K. Burke, *Restoring the Density-Gradient Expansion for Exchange in Solids and Surfaces.* Physical Review Letters 100, 136406 (2008) (http://dx.doi.org/10.1103/PhysRevLett.100.136406).

15. J. Tao, J.P. Perdew, V.N. Staroverov and G.E. Scuseria, *Climbing the Density Functional Ladder: Nonempirical Meta-Generalized Gradient Approximation Designed for Molecules and Solids.* Physical Review Letters 91, 146401 (2003) (http://dx.doi.org/10.1103/PhysRevLett.91.146401).

16. Y. Zhao, D.G. Truhlar, *A new local density functional for main-group thermochemistry, transition metal bonding, thermochemical kinetics, and noncovalent interactions.* Journal of Chemical Physics 125, 194101 (2006) (http://dx.doi.org/10.1063/1.2370993).

17. P.H.T. Philipsen, E. van Lenthe, J.G. Snijders and E.J. Baerends, *Relativistic calculations on the adsorption of CO on the (111) surfaces of Ni, Pd, and Pt within the zeroth-order regular approximation.* Physical Review B 56, 13556 (1997) (http://dx.doi.org/10.1103/PhysRevB.56.13556).

18. P.H.T. Philipsen, and E.J. Baerends, *Relativistic calculations to assess the ability of the generalized gradient approximation to reproduce trends in cohesive properties of solids.* Physical Review B 61, 1773 (2000) (http://dx.doi.org/10.1103/PhysRevB.61.1773).

19. E.S. Kadantsev, R. Klooster. P.L. de Boeij and T. Ziegler, *The Formulation and Implementation of Analytic Energy Gradients for Periodic Density Functional Calculations with STO/NAO Bloch Basis Set.* Molecular Physics 105, 2583 (2007) (http://dx.doi.org/10.1080/00268970701598063).

20. E.S. Kadantsev and T. Ziegler, *Implementation of a Density Functional Theory-Based Method for the Calculation of the Hyperfine A-tensor in Periodic Systems with the Use of Numerical and Slater Type Atomic Orbitals: Application to Paramagnetic Defects.* Journal of Physical Chemistry A 112, 4521 (2008) (http://dx.doi.org/10.1021/jp800494m).

21. E.S. Kadantsev and T. Ziegler, *Implementation of a DFT Based Method for the Calculation of Zeeman g-tensor in Periodic Systems with the use of Numerical and Slater Type Atomic Orbitals.* Journal of Physical Chemistry A 113, 1327 (2009) (http://dx.doi.org/10.1021/jp805466c).

22. F. Kootstra, P.L. de Boeij and J.G. Snijders, *Efficient real-space approach to time-dependent density functional theory for the dielectric response of nonmetallic crystals.* Journal of Chemical Physics 112, 6517 (2000). (http://dx.doi.org/10.1063/1.481315)

23. P. Romaniello and P.L. de Boeij, *Time-dependent current-density-functional theory for the metallic response of solids.* Physical Review B 71, 155108 (2005) (http://dx.doi.org/10.1103/PhysRevB.71.155108).

24. J.A. Berger, P.L. de Boeij and R. van Leeuwen, *Analysis of the viscoelastic coefficients in the Vignale-Kohn functional: The cases of one- and three-dimensional polyacetylene.*, Physical Review B 71, 155104 (2005) (http://dx.doi.org/10.1103/PhysRevB.71.155104).

25. P. Romaniello and P.L. de Boeij, *Relativistic two-component formulation of time-dependent current-density functional theory: application to the linear response of solids.*, Journal of Chemical Physics 127, 174111 (2007) (http://dx.doi.org/10.1063/1.2780146).

26. J.P. Perdew, A. Ruzsinszky, G. I. Csonka, L. A. Constantin, and J. Sun, *Workhorse Semilocal Density Functional for Condensed Matter Physics and Quantum Chemistry.*, Physical Review Letters 103, 026403 (2009) (http://dx.doi.org/10.1103/PhysRevLett.103.026403).

27. C. Adamo and V. Barone, *Exchange functionals with improved long-range behavior and adiabatic connection methods without adjustable parameters: The mPW and mPW1PW models.* Journal of Chemical Physics 108, 664 (1998) (http://dx.doi.org/10.1063/1.475428).

28. Y. Zhang and W. Yang, *Comment on "Generalized Gradient Approximation Made Simple".* Physical Review Letters 80, 890 (1998) (http://dx.doi.org/10.1103/PhysRevLett.80.890).

29. C. Adamo and V. Barone, *Physically motivated density functionals with improved performances: The modified Perdew.Burke.Ernzerhof model.* Journal of Chemical Physics 116, 5933 (2002) (http://dx.doi.org/10.1063/1.1458927).

30. N.C. Handy and A.J. Cohen, *Left-right correlation energy.* Molecular Physics 99, 403 (2001) (http://dx.doi.org/10.1080/00268970010018431).

31. M. Swart, A.W. Ehlers and K. Lammertsma, *Performance of the OPBE exchange-correlation functional.* Molecular Physics 2004 102, 2467 (2004) (http://dx.doi.org/10.1080/0026897042000275017).

32. S. Grimme, *Semiempirical GGA-Type Density Functional Constructed with a Long-Range Dispersion Correction.* Journal of Computational Chemistry 27, 1787 (2006) (http://dx.doi.org/10.1002/jcc.20495).

33. J.I. Rodriguez, A.M. Köster, P.W. Ayers, A. Santos-Valle, A. Vela and G. Merino, *An efficient grid-based scheme to compute QTAIM atomic properties without explicit calculation of zero-flux surfaces.* Journal of Compututational Chemistry 30, 1082 (2009) (http://dx.doi.org/10.1002/jcc.21134).

34. J.I. Rodriguez, R.F.W. Bader, P.W. Ayers, C. Michel, A.W. Gotz and C. Bo, *A high performance grid-based algorithm for computing QTAIM properties.* Chemical Physics Letters 472, 149 (2009) (http://dx.doi.org/10.1016/j.cplett.2009.02.081).

35. J. Tersoff and D. R. Hamann, *Theory of the scanning tunneling microscope.* Physical Review B 31, 505 (1985) (http://dx.doi.org/10.1103/PhysRevB.31.805).

36. A. Klamt and G. Schüürmann, *COSMO: a new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradient.* Journal of the Chemical Society: Perkin Transactions 2, 799 (1993) (http://dx.doi.org/10.1039/P29930000799).

37. D. Skachkov, M. Krykunov, E. Kadantsev, and T. Ziegler, *The Calculation of NMR Chemical Shifts in Periodic Systems Based on Gauge Including Atomic Orbitals and Density Functional Theory.* Journal of Chemical Theory and Computation 6, 1650 (2010) (http://dx.doi.org/10.1021/ct100046a).

38. J.L. Pascual-ahuir, E. Silla and I. Tuñon, *GEPOL: An improved description of molecular surfaces. III. A new algorithm for the computation of a solvent-excluding surface.* Journal of Computational Chemistry 15, 1127 (1994) (http://dx.doi.org/10.1002/jcc.540151009).

39. B. Delley, *The conductor-like screening model for polymers and surfaces.* Molecular Simulation 32, 117 (2006) (http://dx.doi.org/10.1080/08927020600589684).

40. N.L. Allinger, X. Zhou, J. Bergsma, *Molecular mechanics parameters*, Journal of Molecular Structure: THEOCHEM 312, 69 (1994) (http://dx.doi.org/10.1016/S0166-1280(09)80008-0).

41. S. Grimme, J. Anthony, S. Ehrlich, and H. Krieg, *A consistent and accurate* ab initio *parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu*, The Journal of Chemical Physics 132, 154104 (2010) (http://dx.doi.org/10.1063/1.3382344).

42. S. Grimme, S. Ehrlich, and L. Goerigk, *Effect of the Damping Function in Dispersion Corrected Density Functional Theory*, Journal of Computational Chemistry 32, 1456 (2011) (http://dx.doi.org/10.1002/jcc.21759).

43. P. Haas, F. Tran, P. Blaha, and K. H. Schwarz, *Construction of an optimal GGA functional for molecules and solids*, Physical Review B 83, 205117 (2011) (http://dx.doi.org/10.1103/PhysRevB.83.205117).

44. F. Tran, and P. Blaha, *Accurate Band Gaps of Semiconductors and Insulators with a Semilocal Exchange-Correlation Potential*, Physical Review Letters 102, 226401 (2009) (http://dx.doi.org/10.1103/PhysRevLett.102.226401).

45. D. Alfè, *PHON: A program to calculate phonons using the small displacement method*, Computer Physics Communications 180, 2622 (2009) (http://dx.doi.org/10.1016/j.cpc.2009.03.010).

46. V.I. Anisimov, F. Aryasetiawan, and A.I. Lichtenstein, *First-principles calculations of the electronic structure and spectra of strongly correlated systems: the LDA + U method*, Journal Physics: Condensed Matter 9, 767 (1997) (http://dx.doi.org/10.1088/0953-8984/9/4/002).

47. M. Cococcioni, and S. de Gironcoli, *Linear response approach to the calculation of the effective interaction parameters in the LDA+U method*, Physical Review B 71, 035105 (2005) (http://dx.doi.org/10.1103/PhysRevB.71.035105).

48. D. Skachkov, M. Krykunov, and T. Ziegler, *An improved scheme for the calculation of NMR chemical shifts in periodic systems based on gauge including atomic orbitals and density functional theory*, Canadian Journal of Chemistry 89, 1150 (2011) (http://dx.doi.org/10.1139/V11-050).

49. M. Kuisma, J. Ojanen, J. Enkovaara, and T.T. Rantala, *Kohn-Sham potential with discontinuity for Band gap materials*, Physical review B 82, 115106 (2010) (http://dx.doi.org/10.1103/PhysRevB.82.115106).

50. L. Visscher, and K.G. Dyall, *Dirac-Fock atomic electronic structure calculations using different nuclear charge distributions*, Atomic Data and Nuclear Data Tables 67, 207 (1997) (http://dx.doi.org/10.1006/adnd.1997.0751).

51. A.D. Becke, *A multicenter numerical integration scheme for polyatomic molecules*, Journal of Chemical Physics 88, 2547 (1988) (http://dx.doi.org/10.1063/1.454033).

52. M. Franchini, P.H.T. Philipsen, L. Visscher, *The Becke Fuzzy Cells Integration Scheme in the Amsterdam Density Functional Program Suite*, Journal of Computational Chemistry 34, 1818 (2013) (http://dx.doi.org/10.1002/jcc.23323).

53. M. Franchini, P.H.T. Philipsen, E. van Lenthe, L. Visscher, *Accurate Coulomb Potentials for Periodic and Molecular Systems through Density Fitting*, Journal of Chemical Theory and Computation 10, 1994 (2014) (http://dx.doi.org/10.1021/ct500172n).

54. D. Koller, F. Tran, and P. Blaha, *Improving the Modified Becke-Johnson Exchange Potential.*, Physical Review B 83, 155109 (2012) (http://dx.doi.org/10.1103/PhysRevB.85.155109).

55. R. A.Jishi, O. B. Ta, and A. Sharif, *Modeling of Lead Halide Perovskites for Photovoltaic Applications.*, Archive (http://arxiv.org/abs/1405.1706).

56. M. Raupach and R. Tonner, *A periodic energy decomposition analysis method for the investigation of chemical bonding in extended systems*, The Journal of Chemical Physics 142, 194105 (2015) (http://dx.doi.org/10.1063/1.4919943).

57. X. Ren, P. Rinke, V. Blum, J. Wieferink, A. Tkatchenko, A. Sanfilippo, K. Reuter and M. Scheffler, *Resolution-of-identity approach to Hartree–Fock, hybrid density functionals, RPA, MP2 and GW with numeric atom-centered orbital basis functions*, New J. Phys. 14 053020 (http://dx.doi.org/10.1088/1367-2630/14/5/053020).

58. J.A. Berger, P. Romaniello, R. van Leeuwen and P.L. de Boeij, *Performance of the Vignale-Kohn functional in the linear response of metals*, Phys. Rev. B 74, 245117 (http://dx.doi.org/10.1103/PhysRevB.74.245117).

59. J.A. Berger, *Fully Parameter-Free Calculation of Optical Spectra for Insulators, Semiconductors, and Metals from a Simple Polarization Functional*, Phys. Rev. Lett. 115, 137402 (http://dx.doi.org/10.1103/PhysRevLett.115.137402).

60. Z. Qian and G. Vignale, *Dynamical exchange-correlation potentials for an electron liquid*, Phys. Rev. B 65, 235121 (http://dx.doi.org/10.1103/PhysRevB.65.235121).

61. S. Conti, R. Nifosì and M.P. Tosi, *The exchange - correlation potential for current-density functional theory of frequency-dependent linear response*, J. Phys. Condens. Matter 9, L475 (http://dx.doi.org/10.1088/0953-8984/9/34/004).

62. J. Heyd, G.E. Scuseria and M. Ernzerhof, *Hybrid functionals based on a screened Coulomb potential*, J. Chem. Phys. 118, 8207 (2003) (http://dx.doi.org/10.1063/1.1564060).

63. M.A.L. Marques, M.J.T. Oliveira, and T. Burnus, *Libxc: a library of exchange and correlation functionals for density functional theory*, Computer Physics Communications 183, 2272 (2012) (http://dx.doi.org/10.1016/j.cpc.2012.05.007).

64. (a) Raupach and R. Tonner, unpublished. Please contact the authors of reference 56.

65. W. Setyawan and S. Curtarolo, *High-throughput electronic band structure calculations: Challenges and tools*, Computational Materials Science 49 (2010) 299–312 (http://dx.doi.org/10.1016/j.commatsci.2010.05.010).

66. Rui Li, Jiaxing Zhang, Shimin Hou, Zekan Qian, Ziyong Shen, Xingyu Zhao, Zengquan Xue, *A corrected NEGF + DFT approach for calculating electronic transport through molecular devices: Filling bound states and patching the non-equilibrium integration*, Chemical Physics 336 (2007) 127-135 (http://dx.doi.org/10.1016/j.chemphys.2007.06.011).

67. Zeng-hui Yang, Haowei Peng, Jianwei Sun, and John P. Perdew, *More realistic band gaps from meta-generalized gradient approximations: Only in a generalized Kohn-Sham scheme*, Physical Review B 93, 205205 (2016) (http://dx.doi.org/10.1103/PhysRevB.93.205205).

68. C. J. O. Verzijl and J. M. Thijssen *DFT-Based Molecular Transport Implementation in ADF/BAND*, J. Phys. Chem. C, 2012, 116 (46), pp 24393–24412 (http://dx.doi.org/10.1021/jp3044225).

KEYWORDS

- *AIMCriticalPoints* (page 77)
- *ALLOW* (page 8)
- *ATENSOR* (page 60)
- *ATOMS* (page 13)
- *ATOMPROPS* (page 14)
- *ATOMTYPE* (page 34)
- *BASISDEFAULTS* (page 31)
- *BECKEGRID* (page 37)
- *BFIELD* (page 29)
- *BZPATH* (page 76)
- *BZSTRUCT* (page 75)
- *COMMENT* (page 7)
- *CONFINEMENT* (page 36)
- *CONSTRAINTS* (page 51)
- *CONVERGENCE* (page 42)
- *COORDINATES* (page 13)
- *CPVECTOR* (page 46)
- *DEBUG* (page 10)
- *DEFAULTSCONVENTION* (page 31)
- *DEFINE* (page 8)
- *DENSITYPLOT* (page 88)
- *DEPENDENCY* (page 45)
- *DIIS* (page 43)
- *DIRIS* (page 44)
- *DOS* (page 73)
- *EFFECTIVEMASS* (page 61)
- *EFG* (page 60)
- *EField* (page 29)
- *EIGTHRESHOLD* (page 9)
- *ELECTRONHOLE* (page 96)
- *ENFORCEDSPINPOLARIZATION* (page 96)
- *ESR* (page 60)
- *FatBands* (page 75)
- *FERMI* (page 46)
- *FORMFACTORS* (page 62)
- *FRAGMENT* (page 77)
- *FREQUENCIES* (page 50)
- *FuzzyPotential* (page 29)
- *GeometryFile* (page 15)
- *GEOOPT* (page 48)
- *GRADIENTS* (page 47)

- *XC* (page 17)
- *ZLMFIT* (page 40)