BAND Manual

Amsterdam Modeling Suite 2025.1

www.scm.com

Apr 04, 2025

CONTENTS

CHAPTER

ONE

GENERAL

1.1 Introduction

The periodic DFT program **BAND** can be used for calculations on periodic systems, i.e. polymers, slabs and crystals. It uses Density Functional Theory (DFT) in the Kohn-Sham approach. BAND shares many of the core algorithms with ADF, although important differences remain (a noteworthy difference is that BAND uses numerical atomic orbitals as basis functions).

This User's Manual describes how to use the program, how input is structured, what files are produced, and so on. The *Examples section* (page 209) explains the most popular features in detail, by commenting on the input and output files in the $AMSHOME/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.

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.

Graphical User Interface (GUI) tutorials: GUI overview tutorials, BAND-GUI tutorials

This manual 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* (page 327).

1.2 Feature List

1.2.1 Model Hamiltonians

- *XC energy functionals and potentials* (page 15)
  - *LDA* (page 16), *GGA* (page 16), *meta-GGA* (page 18), *Model potentials* (page 21)
  - *Range-separated Hybrids* (page 24)
  - *GGA+U (Hubbard)* (page 25)
  - *LibXC library* (page 22)
  - *Grimme dispersion corrections* (page 19)
- *Relativistic effects: ZORA and spin-orbit coupling* (page 33) (including non-collinear magnetization)
- *COSMO* (page 34) solvation model
- Homogeneous *electric* (page 49) and *magnetic* (page 49) fields

1.2.2 Structure and Reactivity

- Geometry optimization, transition state search, linear transit, PES-scan, molecular dynamics via **AMS**. See the AMS Manual for details.
- Formation energy with respect to isolated atoms (which are computed with a fully numerical Herman-Skillman type subprogram)

1.2.3 Spectroscopy and Properties

- Normal modes, phonon dispersion curves (and related thermodynamic properties) and elastic tensor via **AMS** See the AMS Manual for details.
- Frequency-dependent dielectric function of systems periodic in one, two and three dimensions in the *Time-dependent Current-DFT* (page 113) (TD-CDFT) formalism
- *ESR and EPR* (page 124) (electron paramagnetic resonance) and *EFG* (page 126) (Nuclear Quadrupole Interaction)
- *Form factors* (page 130) (X-ray structures)
- *NMR shielding tensor* (page 127)

1.2.4 Charge transport

- *Non-Equilibrium Green's Function* (page 165) (NEGF) for calculating transmission function and current
- *Effective mass* (page 128) for electrons and holes mobility

1.2.5 Analysis

- *Various Atomic charges* (page 153), including Mulliken, Hirshfeld, CM5 and Voronoi
- Mulliken populations for basis functions, overlap populations between atoms or between basis functions
- *Densities-of-States* (page 143): DOS, PDOS and OPWDOS/COOP (see also: Band Structure and COOP tutorial)
- *Local Densities-of-States* (page 187) LDOS (STM images)
- *3D filed plotting of various properties* (page 159), such as orbitals (Bloch-waves), deformation densities, Coulomb potentials, …
- *Band Structure plot* (page 148) along edges of the Brillouin zone
- *Fermi surface* (page 152): (view with amsbands)
- *Fragment* (page 156) orbitals and a Mulliken type population analysis in terms of the fragment orbitals
- *Quantum Theory of Atoms In Molecules* (page 153) (QT-AIM, aka Bader Analysis). Atomic charges and critical points
- Electron Localization Function (*ELF* (page 183))
- Fragment based Periodic Energy Decomposition Analysis (*PEDA* (page 157))
- PEDA combined with Natural Orbitals for Chemical Valency (NOCV) to decompose the orbital relaxation (*PEDA-NOCV* (page 158))

1.3 What's new in Band 2025.1

- Full hybrids (like B3PW) for band gaps.
- *updated Libxc library to libxc7.0.0* (page 22)
- General purpose LAK meta gga, by Lebeda and coworkers.
- For the EffectiveMass the positon of the HOMO and LUMO are determined along the high symmetry path. To get the pre 2025 behavior set useBandStructureInfoFromPath=No.
- *DFT-D3 updated library (s-dftd3 1.2.1)* (page 19) and *DFT-D4 updated library (dftd4 3.7.0)* (page 19)
  - inclusion of actinide parameters (Z = 87 (Fr) - 103 (Lr))

1.4 What's new in Band 2024.1

- The unit of the DOS and PDOS is now states-per-unit-cell/energy. Before ams2024 this was multiplied by the energy interval, and had the unit states-per-unit-cell. To reproduce the old behavior use DOS%CompensateDeltaE=No.
- Restart of the DOS: i.e. calculate it from a previous calculation. This may be useful when you forgot to request it, or want to change the energy window/grid. In addition, a better k sampling can be used for the DOS only (compared to the k-grid used in the previous SCF calculation). See also the missing DOS problem addressed in *restarting the DOS* (page 189).
- Restart of the BandStructure. This can be useful if you forgot to request it, or want to have it with a finer `BandStructure%DeltaK` key.

1.5 What's new in Band 2023.1

- Improved SCF convergence with the MultiStepper.
- Special (all electron) basis sets for comparison with literature data. Pure *STO* (page 59) basis sets for direct comparison with ADF, and pure *GTO* (page 59) basis sets as used by many other codes. Large GTO basis sets can also be useful for accurate MP2/RPA/GW calculations.
- MP2/RPA/GW for molecules
- TASKCC functional from Lebeda and coworkers (https://doi.org/10.1103/PhysRevResearch.4.023061). Compared to the TASKXC functional (good for band gaps) this improves atomization energies (the energy of open shell systems).

1.6 What's new in Band 2022.1

- Mulliken analysis (including PDOS) with spin-orbit coupling, see tutorial
- *Alternative elements* (page 66) (model impurities, fractional site occupancies)
- *Electron energy density function* (page 188), also known as X.
- *Fermi surface* (page 152), see tutorial

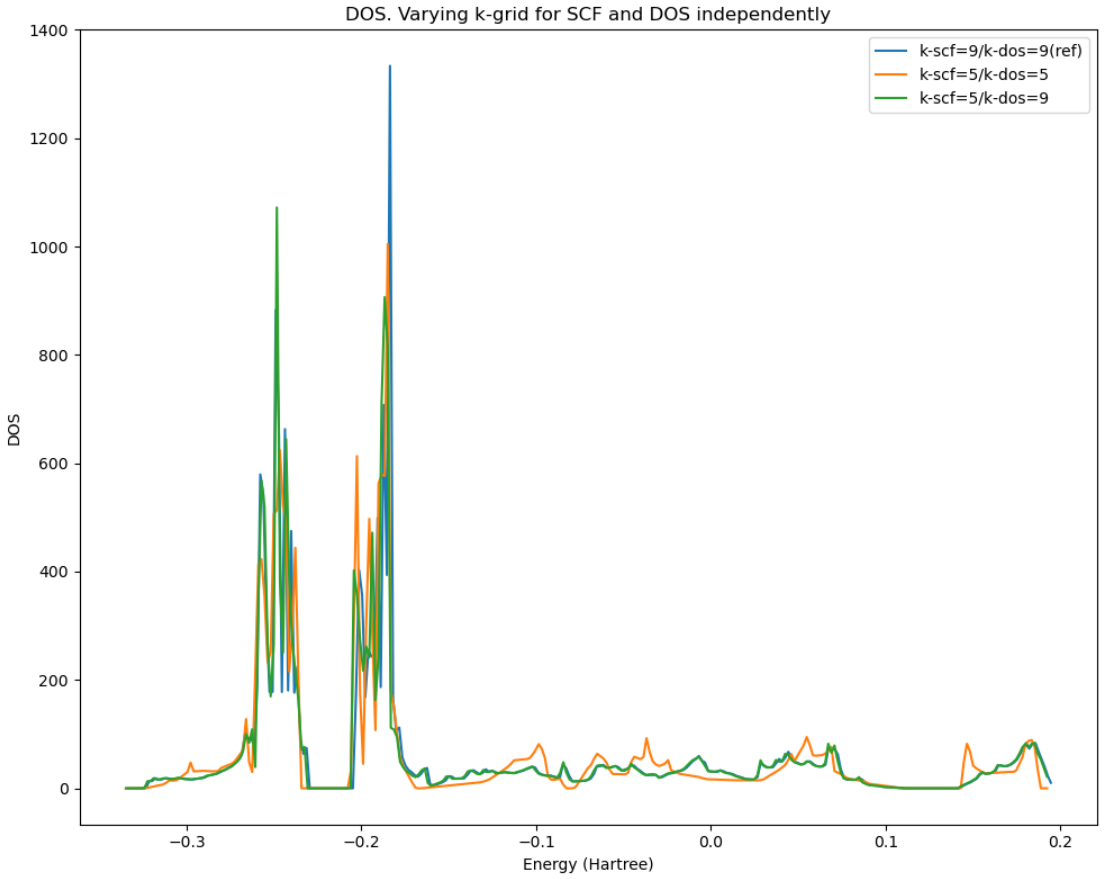


Fig. 1.1: DOS of a CuNiSlab. The best result is when using a 9x9 k-grid for both the SCF and the DOS calculation (blue curve). Using a worse 5x5 grid for both the SCF and the DOS produces a quite different DOS (amber). Doing the SCF with the coarser 5x5 grid and restarting the DOS with the finer 9x9 grid gives the green DOS, matching closely, and mostly hides, the best DOS (blue). See *Example: DOS and BandStructure from a previous calculation* (page 249).

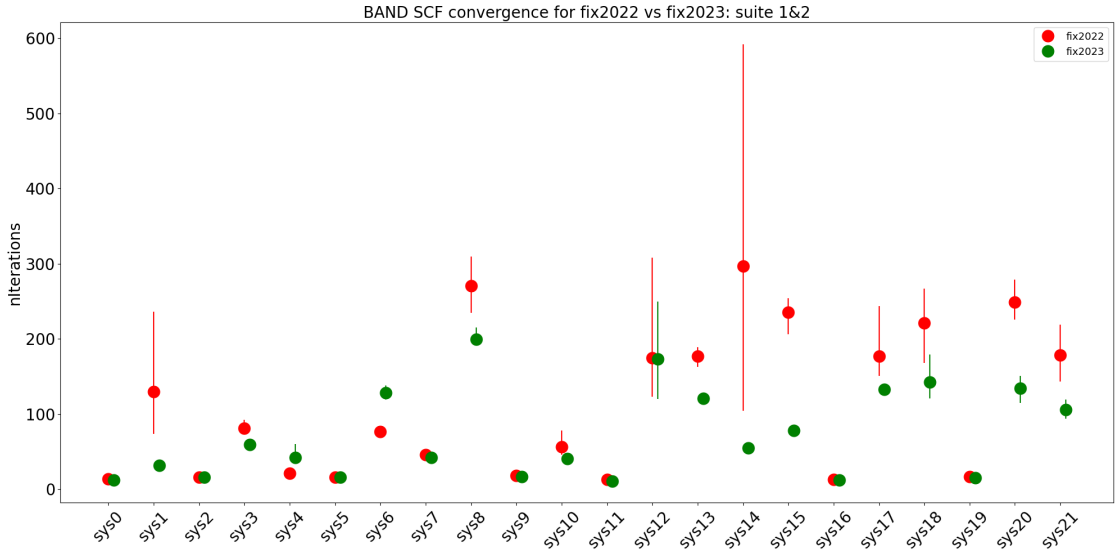


Fig. 1.2: Comparison of the number of SCF cycles needed. For easy systems there is not much difference, for more difficult systems, however, the fix2023 (green) is an improvement over the fix2022 (red). As there can be some randomness in the number of iterations (butterfly effect) the calculations are repeated five times (using a different number of cores), the dot is the average number of cycles used, and the vertical lines show the spread in the number of iterations (if any). The maximum number of iterations was set to 1000.

1.7 What's new in Band 2021.1

- The *D4(EEQ) dispersion correction* (page 19) can now be used for periodic systems.
- *r2SCAN-D4* (page 22) XC functional
- New AMS driver features can be used in combination with BAND. See the page What's new in the AMS driver for more details.

1.8 What's new in Band 2020

- New default: *Scalar relativistic ZORA* (page 33) is now enabled by default (in previous versions "non-relativistic" was the default).
- *Basis sets for Uue (Z=119) and Ubn (Z=120) added* (page 54)
- BAND can be used in QM/MM calculations with the new Hybrid Engine. See the Hybrid Engine manual and tutorial for more details.
- New AMS driver features can be used in combination with BAND. See the page What's new in the AMS2020 driver for more details (particularly important for BAND are the new strain-constraints and the periodic geometry optimizer improvements).

1.9 Input

The input options for Band are specified in a text file consisting of a series of key-value pairs, possibly nested in blocks. The input is usually embedded in an executable shell script. This is the content of a typical shell script for running a Band calculation:

```
#!/bin/sh

$AMSBIN/ams <<eor
   # This is the beginning of the input.
   # The input consists of key-value pairs and blocks.
   # Here we define the input option for the AMS driver:

   Task GeometryOptimization

   System
      Atoms
         H 0.0 0.0 0.0
         H 0.9 0.0 0.0
      End
   End

   # Next comes the Band "Engine" block. The input options for Band, which are
   # described in this manual, should be specified in this block:

   Engine Band
      Basis
         Type DZP
      End

      XC
         GGA PBE
      End
   EndEngine
eor
```

To run the calculation above from command-line you should:

1. Create a text file called, for example, `test.run` and copy-paste the content of the script above
2. Make the script executable by typing in your shell `chmod u+x test.run`
3. Execute the script and redirect the output to a file: `./test.run > out`

The program will create a directory called `ams.results`. Inside it, you will find the *logfile* `ams.log` (which can be used to monitor the progress of the calculation) and the binary results files `ams.rkf` and `band.rkf`. After the calculation is completed, you can examine the output file `out`. For more details, see the AMS documentation.

See also:

The *Examples* (page 209) section contains a large number of input examples.

Important: All options described in this manual should be specified in the Band Engine block:

```
# All Band keywords should be specified inside the 'Engine Band' block
Engine Band
   Basis
      Type DZP
```

(continues on next page)

(continued from previous page)

```
   End

   XC
      GGA PBE
   End
EndEngine
```

1.9.1 General remarks on input structure and parsing

- Most keys are optionals. Defaults values will be used for keys that are not specified in the input
- Keys/blocks can either be *unique* (*i.e.* they can appear in the input only once) or *non-unique*. (i.e. they can appear multiple times in the input)
- The order in which keys or blocks are specified in the input does not matter. Possible exceptions to this rule are a) the content of non-standard blocks b) some non-unique keys/blocks)
- Comments in the input file start with one of the following characters: `#`, `!`, `::`:

```
# this is a comment
! this is also a comment
:: yet another comment
```

- Empty lines are ignored
- The input parsing is **case insensitive** (except for string values):

```
# this:
UseSymmetry false
# is equivalent to this:
USESYMMETRY FALSE
```

- Indentation does not matter and multiple spaces are treaded as a single space (except for string values):

```
# this:
     UseSymmetry     false
# is equivalent to this:
UseSymmetry false
```

- Environment variables in the input will be replaced by their value. We only support simple substitution of variables (optionally surrounded by `{}`). All other parameter expansion features of the POSIX shell (https://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html#tag_18_06_02) (e.g. default values) are **not** supported. Expansion of a variable can be prevented by prefixing the leading dollar (`$`) with a backslash (`\`). Note that variables which are not set will expand to an empty string, just like in a shell:

```
File $SOME_DIRECTORY/file.txt
File ${SOME_DIRECTORY}_BACKUP/file.txt
Key  \$WILL_NOT_EXPAND
```

1.9.2 Keys

Key-value pairs have the following structure:

```
KeyName Value
```

Possible types of keys:

bool key
The value is a single Boolean (logical) value. The value can be `True` (equivalently `Yes`) or `False` (equivalently `No`.). Not specifying any value is equivalent to specifying `True`. Example:

```
KeyName Yes
```

integer key
The value is a single integer number. Example:

```
KeyName 3
```

float key
The value is a single float number. For scientific notation, the E-notation is used (e.g. -2.5×10^{-3} can be expressed as `-2.5E-3`). The decimal separator should be a dot (`.`), and **not** a comma (`,`). Example:

```
KeyName -2.5E-3
```

Note that fractions (of integers) can also be used:

```
KeyName 1/3    (equivalent to: 0.33333333333...)
```

string key
The value is a string, which can include white spaces. Only ASCII characters are allowed. Example:

```
KeyName Lorem ipsum dolor sit amet
```

multiple_choice key
The value should be a single word among the list options for that key (the options are listed in the documentation of the key). Example:

```
KeyName SomeOption
```

integer_list key
The value is list of integer numbers. Example:

```
KeyName 1 6 0 9 -10
```

Note that one can also specify ranges of integers by specifying the interval and (optionally) the step size separated by colons:

```
KeyName 1:5       (equivalent to: 1 2 3 4 5)
KeyName 2:10:2    (equivalent to: 2 4 6 8 10)
KeyName 20:10:-2  (equivalent to: 20 18 16 14 12 10)
```

Note also that ranges can be freely combined with individual numbers:

```
KeyName 1:5 10 20  (equivalent to: 1 2 3 4 5 10 20)
```

float_list key
The value is list of float numbers. The convention for float numbers is the same as for Float keys. Example:

```
KeyName 0.1 1.0E-2 1.3
```

Float lists can also be specified as a range with equidistant points, by specifying the interval's boundaries (inclusive) as well as the number of desired subintervals separated by colons:

```
KeyName 1.0:1.5:5  (equivalent to: 1.0 1.1 1.2 1.3 1.4 1.5)
```

Range specifications can be freely combined with each other and single numbers:

```
KeyName 0.0 1.0:1.5:5 2.0:3.0:10
```

1.9.3 Blocks

Blocks give a hierarchical structure to the input, grouping together related keys (and possibly sub-blocks). In the input, blocks generally span multiple lines, and have the following structure:

```
BlockName
   KeyName1 value1
   KeyName2 value2
   ...
End
```

Headers

For some blocks it is possible (or necessary) to specify a *header* next to the block name:

```
BlockName someHeader
   KeyName1 value1
   KeyName2 value2
   ...
End
```

Compact notation

It is possible to specify multiple key-value pairs of a block on a single line using the following notation:

```
# This:
BlockName KeyName1=value1 KeyName2=value2

# is equivalent to this:
BlockName
   KeyName1 value1
   KeyName2 value2
End
```

Notes on compact notation:

- The compact notation cannot be used for blocks with headers.
- Spaces (blanks) between the key, the equal sign and the value are ignored. However, if a value itself needs to contain spaces (e.g. because it is a list, or a number followed by a unit), the entire value must be put in either single or double quotes:

```
# This is OK:
BlockName Key1=value Key2 = "5.6 [eV]" Key3='5 7 3 2'
# ... and equivalent to:
```

(continues on next page)

(continued from previous page)

```
BlockName
   Key1  value
   Key2  5.6 [eV]
   Key3  5 7 3 2
End

# This is NOT OK:
BlockName Key1=value Key2 = 5.6 [eV] Key3=5 7 3 2
```

Non-standard Blocks

A special type of block is the *non-standard block*. These blocks are used for parts of the input that do not follow the usual key-value paradigm.

A notable example of a non-standard block is the `Atoms` block (in which the atomic coordinates and atom types are defined).

1.9.4 Including an external file

You can include an external ASCII file in the input with the `@include` directive:

```
@include FileName.in
@include "file name with spaces.in"
```

The file name should include the path, either absolute or relative to the run-directory. The content of the file is included in the input at the point where the `@include` directive occurs. The `@include` directive may occur any number of times in the input.

The `@include` feature makes it easy to pack your preferred settings in one file and use them in every run with minimum input-typing effort.

1.9.5 Units

Some keys have a default unit associated (not all keys have units). For such keys, the default unit is mentioned in the key documentation. One can specify a different unit within square brackets at the end of the line:

```
KeyName value [unit]
```

For example, assuming the key `EnergyThreshold` has as default unit `hartree`, then the following definitions are equivalent:

```
# Use defaults unit:
EnergyThreshold 1.0

# use eV as unit:
EnergyThreshold 27.211 [eV]

# use kcal/mol as unit:
EnergyThreshold 627.5 [kcal/mol]

# hartree is the atomic unit of energy:
EnergyThreshold 1.0 [hartree]
```

Available units:

- **Energy**: `hartree`, `joule`, `eV`, `kJ/mol`, `kcal/mol`, `cm1`, `MHz`
- **Forces**: `hartree/bohr`, `eV/bohr`, `hartree/angstrom`, `eV/angstrom`, `kcal/mol/angstrom`, `kJ/mol/angstrom`
- **Length**: `bohr`, `angstrom`, `meter`
- **Angles**: `radian`, `degree`
- **Mass**: `el`, `proton`, `atomic`, `kg`
- **Pressure**: `atm`, `pascal`, `GPa`, `a.u.`, `bar`, `kbar`
- **Electric field**: `V/angstrom`, `V/meter`, `a.u.`

CHAPTER

TWO

AMS DRIVER'S TASKS AND PROPERTIES

BAND is an engine used by the AMS driver. While BAND's specific options and properties are described in this manual, the definition of the system, the selection of the task and certain (PES-related) properties are documented in the AMS driver's manual.

In this page you will find useful links to the relevant sections of the AMS driver's Manual.

2.1 Geometry, System definition

The definition of the system, i.e. the atom types and atomic coordinates (and optionally, the systems' net charge, the lattice vector, the input bond orders, external homogeneous electric field, external point charges, atomic masses for isotopes) are part of the AMS driver input. See the System definition section of the AMS manual.

2.2 Tasks: exploring the PES

The job of the AMS driver is to handle all changes in the simulated system's geometry, e.g. during a geometry optimization or molecular dynamics calculation, using energy and forces calculated by the engine.

These are the tasks available in the AMS driver:

- Single Point
- Geometry Optimization
- Transition State Search
- IRC (Intrinsic Reaction Coordinate)
- PESScan (Potential Energy Surface Scan, including linear transit)
- NEB (Nudged Elastic Band)
- Vibrational Analysis
- Molecular Dynamics

2.3 Properties in the AMS driver

The following properties can be requested to the BAND engine in the AMS driver's input:

- Atomic charges
- Dipole Moment (see also *Dipole moment and Berry Phase* (page 131))
- Dipole Gradients
- Elastic tensor
- Nuclear Gradients / Forces
- Hessian
- Infrared (IR) spectra / Normal Modes
- Thermodynamic properties
- PES point character
- Phonons
- Stress tensor
- Elastic tensor
- VCD (Vibrational Circular Dichroism)

CHAPTER

THREE

MODEL HAMILTONIANS

3.1 Density Functional (XC)

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.

The density functional approximation is controlled by the XC key.

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 24).

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}}
   {Dispersion Grimme4 {s6=...} {s8=...} {a1=...} {a2=...}}
   {Model [LB94|TB-mBJ|KTB-mBJ|JTS-MTB-MBJ|GLLB-SC|BGLLB-VWN|BGLLB-LYP]}
   {SpinOrbitMagnetization [None|NonCollinear|CollinearX|CollinearY|CollinearZ]}
   {LibXC {Functional}}
```

(continues on next page)

(continued from previous page)

```
   {DOUBLEHYBRID doublehybrid}
   {RPA {option}}
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**)

3.1.1 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 parametrization of electron gas data given by Vosko, Wilk and Nusair (ref[1], 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] 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]
- **PW86x**: the correction advocated in 1986 by Perdew-Wang[4]
- **PW91x**: the exchange correction proposed in 1991 by Perdew-Wang[5]
- **mPWx**: the modified PW91 exchange correction proposed in 1998 by Adamo-Barone[6]

[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) (https://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) (https://doi.org/10.1007/PL00020511).

[3] A.D. Becke, *Density-functional exchange-energy approximation with correct asymptotic behavior.* Physical Review A 38, 3098 (1988) (https://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) (https://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) (https://doi.org/10.1103/PhysRevB.46.6671).

[6] 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) (https://doi.org/10.1063/1.475428).

- **PBEx**: the exchange correction proposed in 1996 by Perdew-Burke-Ernzerhof[7]
- **HTBSx**: the HTBS exchange functional[8]
- **RPBEx**: the revised PBE exchange correction proposed in 1999 by Hammer-Hansen-Norskov[9]
- **revPBEx**: the revised PBE exchange correction proposed in 1998 by Zhang-Yang[10]
- **mPBEx**: the modified PBE exchange correction proposed in 2002 by Adamo-Barone[11]
- **OPTX**: the OPTX exchange correction proposed in 2001 by Handy-Cohen[12]

For the correlation part the options are:

- **Perdew**: the correlation term presented in 1986 by Perdew[13]
- **PBEc**: the correlation term presented in 1996 by Perdew-Burke-Ernzerhof[7]
- **PW91c**: the correlation correction of Perdew-Wang (1991), see[Page 16, 51617]
- **LYP**: the Lee-Yang-Parr 1988 correlation correction[18]

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)[19]

[7] J.P. Perdew, K. Burke and M. Ernzerhof, *Generalized Gradient Approximation Made Simple.* Physical Review Letters 77, 3865 (1996) (https://doi.org/10.1103/PhysRevLett.77.3865).

[8] 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) (https://doi.org/10.1103/PhysRevB.83.205117).

[9] 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) (https://doi.org/10.1103/PhysRevB.59.7413).

[10] Y. Zhang and W. Yang, *Comment on "Generalized Gradient Approximation Made Simple".* Physical Review Letters 80, 890 (1998) (https://doi.org/10.1103/PhysRevLett.80.890).

[11] 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) (https://doi.org/10.1063/1.1458927).

[12] N.C. Handy and A.J. Cohen, *Left-right correlation energy.* Molecular Physics 99, 403 (2001) (https://doi.org/10.1080/00268970010018431).

[13] J.P. Perdew, *Density-functional approximation for the correlation energy of the inhomogeneous electron gas.* Physical Review B 33, 8822 (1986) (https://doi.org/10.1103/PhysRevB.33.8822).

[16] 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) (https://doi.org/10.1063/1.464906).

[17] 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) (https://doi.org/10.1063/1.468265).

[18] 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) (https://doi.org/10.1103/PhysRevB.37.785).

[19] M. Swart, A.W. Ehlers and K. Lammertsma, *Performance of the OPBE exchange-correlation functional.* Molecular Physics 2004 102, 2467 (2004) (https://doi.org/10.1080/00268970420002750 17).

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[20].

The available functionals of this type are:

- **TPSS**: The 2003 meta-GGA[21]
- **M06L**: The meta-GGA as developed by the Minesota group[22]
- **revTPSS**: The 2009 revised meta-GGA[23]
- **MVS**: Functional by Sun-Perdew-Ruzsinszky[24]
- **MS0**: Functional by Sun et al.[25]
- **MS1**: Functional by Sun et al.[26]
- **MS2**: Functional by Sun et al.[Page 18, 26]
- **SCAN**: Functional by Sun et al.[27]
- **TASKxc**: by Aschebrock et al (https://journals.aps.org/prresearch/abstract/10.1103/PhysRevResearch.1.033082). Intended for band gaps and charge transfer systems.
- **TASKCC**: Functional by Lebeda et al.[37], improves TASKxc atomization energies
- **LAK**: Functional by Lebeda and coworkers.[47], it is a general purpose functional.

Note: For Meta-GGA XC functionals, it is recommended to use `small` or `none` *frozen core* (page 54) (the frozen orbitals are computed using LDA and not the selected Meta-GGA)

[20] R. Neumann, R.H. Nobes and N.C. Handy, *Exchange functionals and potentials.* Molecular Physics 87, 1 (1996) (https://doi.org/10.1080/00268979600100011).

[21] 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) (https://doi.org/10.1103/PhysRevLett.91.146401).

[22] 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) (https://doi.org/10.1063/1.2370993).

[23] 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) (https://doi.org/10.1103/PhysRevLett.103.026403).

[24] J. Sun, J.P. Perdew, and A. Ruzsinszky, *Semilocal density functional obeying a strongly tightened bound for exchange*, Proceedings of the National Academy of Sciences 112, 685 (2015) (https://doi.org/10.1073/pnas.1423145112)

[25] J. Sun, B. Xiao, A. Ruzsinszky, *Communication: Effect of the orbital-overlap dependence in the meta generalized gradient approximation*, Journal of Chemical Physics 137, 051101 (2012) (https://doi.org/10.1063/1.4742312).

[26] J. Sun, R. Haunschild, B. Xiao, I.W. Bulik, G.E. Scuseria, J.P. Perdew, *Semilocal and hybrid meta-generalized gradient approximations based on the understanding of the kinetic-energy-density dependence*, Journal of Chemical Physics 138, 044113 (2013) (https://doi.org/10.1063/1.4789414).

[27] J. Sun, A. Ruzsinszky, J.P. Perdew, *Strongly Constrained and Appropriately Normed Semilocal Density Functional*, Physical Review Letters 115, 036402 (2015) (https://doi.org/10.1103/PhysRevLett.115.036402).

[37] T. Lebeda, T. Aschebrock, and S. Kümmel, *First steps towards achieving both ultranonlocality and a reliable description of electronic binding in a meta-generalized gradient approximation*, Phys. Rev. Research 4, 023061 (2022) (https://doi.org/10.1103/PhysRevResearch.4.023061)

[47] T. Lebeda, T. Aschebrock, and S. Kümmel, *Balancing the contributions to the gradient expansion: Accurate binding and band gaps with a nonempirical meta-GGA*, Phys. Rev. Letters, 023061 (2024) (accepted for publication) (https://journals.aps.org/prl/accepted/d2072Y38Fff15b8b58146dd51ea85cccafaf9f3dc)

3.1.2 Dispersion Correction

BAND supports the new *D4(EEQ)* (page 19) as well as the older *D3 and D3-BJ* (page 19) dispersion corrections from the group of Stefan Grimme:

D4(EEQ)

Version 3.7.0 of dftd4 is used (https://github.com/dftd4/dftd4), which includes parameters for actinides. For elements for which no parameters are present (103<Z<112,Z>118), the parameters for the element Z'=Z-32 will be used.

`Dispersion Grimme4 {s6=...} {s8=...} {a1=...} {a2=...}`

If `Dispersion Grimme4` is present in the `XC` block the D4(EEQ) dispersion correction (with the electronegativity equilibrium model) by the Grimme group[33][34] will be added to the total bonding energy, gradient and second derivatives, where applicable.

The D4(EEQ) model has four parameters: s_6, s_8, a_1 and a_2 and their value should depend on the XC functional used. For many functionals the D4(EEQ) parameters are **predefined**, like B1B95, B3LYP, B3PW91, B97-D, BLYP, BP86, CAM-B3LYP, HartreeFock, OLYP, OPBE, PBE, PBE0, PW6B95, REVPBE, RPBE, TPSS, TPSSH. For such functionals it is enough to specify `Dispersion Grimme4` in the input block. E.g.:

```
XC
  GGA BLYP
  Dispersion Grimme4
END
```

For many other functionals you should explicitly specify the D4(EEQ) parameters in the `Dispersion` key (otherwise the PBE parameters will be used). For example, for the PW91 functional you could use the following input (although this is not necessary, since the parameters for PW91 are predefined):

```
XC
  GGA PW91
  Dispersion Grimme4 s6=1.0 s8=0.7728 a1=0.3958 a2=4.9341
END
```

The D4(EEQ) parameters for many functionals can be found in the supporting information of the following paper:[Page 19, 33], see also https://github.com/dftd4/dftd4.

D3 and D3-BJ

In BAND parameters for *Grimme3* and *Grimme3 BJDAMP* were updated according to version 1.2.1 of s-dftd3 (https://github.com/dftd3/simple-dftd3).

`DISPERSION Grimme3 BJDAMP {PAR1=par1 PAR2=par2 PAR3=par3 PAR4=par4}`

If this key is present a dispersion correction (DFT-D3-BJ) by Grimme[29] will be added to the total bonding energy, gradient and second derivatives, where applicable.

For many XC functionals, parametrizations are implemented e.g. for B3LYP, TPSS, BP86, BLYP, PBE, PBEsol, RPBE, and SCAN. For example, this is the input block for specifying the PBE functional with Grimme3 BJDAMP dispersion correction (PBE-D3(BJ)):

[33] E. Caldeweyher, S. Ehlert, A. Hansen, H. Neugebauer, S. Spicher, C. Bannwarth, S. Grimme, *A Generally Applicable Atomic-Charge Dependent London Dispersion Correction Scheme*, J. Chem. Phys., 2019, 150, 154122 (https://doi.org/10.1063/1.5090222)

[34] E. Caldeweyher, J.-M. Mewes, S. Ehlert, S. Grimme, *Extension and evaluation of the D4 London-dispersion model for periodic systems*, Phys. Chem. Chem. Phys., 2020, 22, 8499-8512 (https://doi.org/10.1039/D0CP00502A)

[29] 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) (https://doi.org/10.1002/jcc.21759).

```
XC
  GGA PBE
  DISPERSION Grimme3 BJDAMP
End
```

The parametrization has four general parameters that depend on the XC functional. 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 https://www.chemie.uni-bonn.de/grimme/de/software/dft-d3/bj_damping
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[31] 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[28], and RPBE, and will be automatically set if one of these functionals is used. For SCAN parameters from Ref.[30] are used. For all other functionals, PBE-D3 parameters are used as default. You can explicitly specify the three parameters.

variable	variable on https://www.chemie.uni-bonn.de/grimme/de/software/dft-d3/zero_damping
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]. 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[Page 20, 32].

[31] 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) (https://doi.org/10.1063/1.3382344).

[28] 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) (https://doi.org/10.1103/PhysRevLett.100.136406).

[30] J.G. Brandenburg, J.E. Bates, J. Sun, and J.P. Perdew, *Benchmark tests of a strongly constrained semilocal functional with a long-range dispersion correction*, Physical Review B 94, 115144 (2016) (https://doi.org/10.1103/PhysRevB.94.115144)

[32] S. Grimme, *Semiempirical GGA-Type Density Functional Constructed with a Long-Range Dispersion Correction.* Journal of Computational Chemistry 27, 1787 (2006) (https://doi.org/10.1002/jcc.20495).

3.1.3 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[35].

TB-mBJ

This model potential can be used to correct for the band gap problem with GGAs for bulk systems[36]. 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[36]	-0.012	1.023	0.5
KTB-mBJ[38]	0.267	0.656	1.0
JTS-mTB-mBJ[39]	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 functional uses a model for the exchange response potential (based on J. Krieger, Y. Li and G. Iafrate response potential[41]) from which the derivative discontinuity follows[40]. This is an accurate functional for band gap predictions and Electric Field Gradient calculations. It is also a fast method and a very good compromise between accuracy and computational cost. This functional is composed of the GLLB exchange response potential and the PBESOL exchange hole and the correlation potential[40].

BGLLB-VWN

This functional is a variation of the GLLB-SC functional using the B88 exchange hole potential and the VWN correlation potential. This functional gives good results for Group I-VII and II-VI semi conductors.

BGLLB-LYP

This functional is a variation of the GLLB-SC functional using the B88 exchange hole potential and the LYP correlation potential. This functional gives good results for large band gap insulators.

One can change the K parameter for the GLLB functionals with the `GLLBKParameter` key:

[35] R. van Leeuwen and E.J. Baerends, *Exchange-correlation potential with correct asymptotic behavior*. Physical Review A 49, 2421 (1994) (https://doi.org/10.1103/PhysRevA.49.2421).

[36] F. Tran, and P. Blaha, *Accurate Band Gaps of Semiconductors and Insulators with a Semilocal Exchange-Correlation Potential*, Physical Review Letters 102, 226401 (2009) (https://doi.org/10.1103/PhysRevLett.102.226401).

[38] D. Koller, F. Tran, and P. Blaha, *Improving the Modified Becke-Johnson Exchange Potential.*, Physical Review B 83, 155109 (2012) (https://doi.org/10.1103/PhysRevB.85.155109).

[39] R. A.Jishi, O. B. Ta, and A. Sharif, *Modeling of Lead Halide Perovskites for Photovoltaic Applications.*, Archive (http://arxiv.org/abs/1405.1706).

[41] J.B. Krieger, Yan Li, G.J. Iafrate, *Derivation and application of an accurate Kohn-Sham potential with integer discontinuity*, Physics Letters A 8, 146 (1990) (https://doi.org/10.1016/0375-9601(90)90975-T)

[40] 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) (https://doi.org/10.1103/PhysRevB.82.115106).

```
XC
   Model [GLLB-SC|BGLLB-VWN|BGLLB-LYP]
   GLLBKParameter val
End
```

The default value is K=0.382 (value obtained from the electron gas model in the original publication).

3.1.4 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 33) 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.

3.1.5 LibXC Library Integration

Any publication employing calculations carried out with LibXC need to cite the current literature reference of LibXC, which is at the moment Ref. [S. Lehtola, C. Steigemann, M.J.T. Oliveira, M.A.L. Marques, SoftwareX 7, 1 (2018) (https://doi.org/10.1016/j.softx.2017.11.002)][14]. Benchmark papers employing functionals from LibXC should especially include a reference to the employed DFT library, since as discussed in Ref. [S. Lehtola, M.A.L. Marques, J. Chem. Phys. 159, 114116 (2023) (https://doi.org/10.1063/5.0167763)][15], the implementations of a given functionals in various program packages may not lead to the same result, even at the complete basis set limit.

LibXC functional

LibXC is a library of approximate XC functionals, see Ref.[42 Page 22, 14]. Version 7.0.0 of LibXC is used. See the LibXC website for the complete list of functionals: https://libxc.gitlab.io/functionals.

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, XLYP, XPBE, HLE16
- **MetaGGA:** M06-L, M11-L, MN12-L, MS0, MS1, MS2, MVS, PKZB, TPSS, HLE17, rSCAN, r2SCAN
- **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

[14] S. Lehtola, C. Steigemann, M.J.T. Oliveira, M.A.L. Marques, *Recent developments in Libxc – A comprehensive library of functionals for density functional theory*, SoftwareX 7, 1 (2018) (https://doi.org/10.1016/j.softx.2017.11.002)

[15] S. Lehtola, M.A.L. Marques, *Reproducibility of density functional approximations: how new functionals should be reported*, Journal of Chemical Physics 159, 114116 (2023) (https://doi.org/10.1063/5.0167763)

[42] 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) (https://doi.org/10.1016/j.cpc.2012.05.007).

- **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 24)): CAM-B3LYP, CAMY-B3LYP, HJS-PBE, HJS-PBESOL, HJS-B97X, HSE03, HSE06, LRC_WPBE, LRC_WPBEH, LCY-BLYP, LCY-PBE, M11, MN12-SX, N12-SX, TUNED-CAM-B3LYP, WB97, WB97X

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 54)).
- For periodic systems only short range-separated functionals can be used (see *Range-separated hybrid functionals* (page 24))
- In case of LibXC the output of the BAND calculation will give the reference for the used functional, see also the LibXC website https://libxc.gitlab.io/functionals.
- 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.
- BAND can not calculate VV10 dispersion dependent LibXC functionals, like VV10, LC-VV10, B97M-V, WB97X-V, WB97M-V. Before BAND2025 BAND would stop if one tries to calculate a VV10 dispersion dependent functional. Starting from BAND2025 BAND will not stop if one includes another dispersion method (with the DISPERSION key), in which case the VV10 dispersion dependent part of the functional will be replaced with the other dispersion method that is specified.
- If you use a GGA functional via LibXC then BAND will calculate the XC potential, similar to what happens with the normal GGA key. For some GGA functionals, however, this gives a problem for certain elements (during the RADIAL part of the code). This has been tested for all elements and no such problems are observed for functionals in this white list: "AM05 BCGP BP86 HCTH-93 HCTH-120 HCTH-147 HCTH-407 HCTH-P14 PBEINT MPBE MPW PBE PBESOL PW91 SOGGA TH-FL XPBE HLE16". When using a GGA not in the white list BAND will automatically switch to another strategy. This means that core orbitals will no longer correspond to the GGA, and also the calculation will be less efficient. If that happens this warning will be printed: WARNING: requested libxc gga functional may give numerical problems for certain atom types (in the RADIAL section))

3.1.6 Hartree-Fock

Pure Hartree-Fock is triggered by the key

```
XC
    HartreeFock true
End
```

- The Hartree-Fock exchange matrix is calculated through a procedure known as Resolution of the Identity (RI). See *RIHartreeFock* (page 79) key.

There are some shortcomings to this option

- It does not work with a frozen core
- There are no gradients available
- Does not work for periodic systems

3.1.7 Range-separated hybrid functionals

Short range-separated hybrid functionals, like the **HSE03** functional[43], can be useful for prediction of more accurate band gaps compared to GGAs. These must be specified via the *LibXC* (page 22) 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 (https://libxc.gitlab.io/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[Page 24, 43]).

Notes:

- Hybrid functionals can only be used in combination with all-electron basis sets (see `CORE NONE` in section *Basis set* (page 54)).
- The Hartree-Fock exchange matrix is calculated through a procedure known as Resolution of the Identity (RI). See *RIHartreeFock* (page 79) 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
```

3.1.8 Double hybrids, MP2 and RPA

Starting from AMS2023, Many-body perturbation theory functionals (double hybrids, MP2, and RPA) as implemented in ADF can also be used in the BAND engine for the molecular (non-periodic) case, see the ADF documentation for an overview over the implemented functionals.

- RPA and double hybrids

Generally, BAND supports the same MBPT functionality as ADF. However, in Band, MP2, RPA, and double hybrids can not be used icw spin-orbit coupling.

[43] J. Heyd, G.E. Scuseria and M. Ernzerhof, *Hybrid functionals based on a screened Coulomb potential*, J. Chem. Phys. 118, 8207 (2003) (https://doi.org/10.1063/1.1564060).

It is recommended to adjust the **Basis** key inside the **Dependency** block and the **DependencyThreshold** inside the **RIHartreeFock** block. Typically, MBPT calculations require a large basis set (See the MBPT documentation inside ADF), ideally of quadruple-zeta quality or larger. In this case, the following settings are recommended.

Usage example:

```
XC
   DoubleHybrid B2PLYP
End

Dependency
   AllowBasisDependency True
   Basis 1e-4
End

Softconfinement
   Quality Excellent
End

RIHartreeFock
   DependencyThreshold 1e-3
End
```

3.1.9 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).

3.1.10 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[45]. See also the review article[44].

```
HubbardU
   Atom
      Element string
      LValue [s | p | d | f]
```

(continues on next page)

[45] 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) (https://doi.org/10.1103/PhysRevB.71.035105).

[44] 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) (https://doi.org/10.1088/0953-8984/9/4/002).

(continued from previous page)

```
      UValue float
   End
   PrintOccupations Yes/No
   Region
      LValue [s | p | d | f]
      Name string
      UValue float
   End
End
```

HubbardU

Type
Block

Description
Options for Hubbard-corrected DFT calculations.

Atom

Type
Block

Recurring
True

Description
Specify Hubbard parameters (U,l) for a certain element

Element

Type
String

Description
Name of the element, such as Cu or Zn

LValue

Type
Multiple Choice

Default value
s

Options
[s, p, d, f]

Description
L value of the shell to apply the Hubbard model to

UValue

Type
Float

Default value
0.0

Unit
Hartree

Description
Hubbard U value.

`PrintOccupations`

Type
Bool

Default value
Yes

Description
Whether or not to print the occupations during the SCF.

`Region`

Type
Block

Recurring
True

Description
Specify Hubbard parameters (U,l) for all atoms in a certain region

`LValue`

Type
Multiple Choice

Default value
s

Options
[s, p, d, f]

Description
L value of the shell to apply the Hubbard model to

`Name`

Type
String

Description
Name of the region

`UValue`

Type
Float

Default value
0.0

Unit
Hartree

Description
Hubbard U value.

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

```
...
Atoms
   Ni 0.000  0.000  0.000
   O 2.085  2.085  2.085
End
...

...
HubbardU
   printOccupations true
   Atom   element=Ni UValue=0.3  lValue=2
End
...
```

It is also possible to specify U per region.

See also:

Example: Fixing the Band gap of NiO with GGA+U (page 222)

3.1.11 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[46]).

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.

[46] 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) (https://doi.org/10.1103/PhysRevB.93.205205).

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.

3.1.12 DFT-1/2

The DFT-1/2 method due to Slater has been extended by Ferreira (PRB,78,125116,2008 (https://doi.org/10.1103/PhysRevB.78.125116)) to address the band gap problem. DFT-1/2 can be used in combination with any XC functional (this method is also referred to as LDA-1/2 or GGA-1/2, depending on the functional used).

The physical picture is that the hole is localized having substantial self energy. Adding an electron to the solid is assumed to go to a very delocalized state with little or no self energy. The method amounts to adding attractive spherical potentials at atomic sites and optimizing the screening parameter for maximal band gap, and can be used on top of any functional, relativistic option and spin option. From this viewpoint the only freedom in the method is the list of active atom types, the ones for which we will add the potential and optimize the gap. The l-dependent potential option from Ferreira is currently not supported.

The simplest approach is to optimize all the atom types. However, one can also look at the character of the top of the valence band, and determine which atoms are contributing to the PDOS there. This can be done by hand by using the bandstructure GUI module. In band there is an option to analyze this automatically, see the Prepare=true sub option.

See also:

Example: DFT-1/2 method for Silicon (page 227)

```
XC
   DFTHalf
      ActiveAtomType
         AtomType string
         IonicCharge float
         ScreeningCutOffs float_list
```

(continues on next page)

(continued from previous page)

```
      End
      Enabled Yes/No
      Prepare Yes/No
      SelfConsistent Yes/No
   End
End
```

XC

DFTHalf

Type
Block

Description
DFT-1/2 method for band gaps. See PRB vol 78,125116 2008. This method can be used in combination with any functional. For each active atom type (see ActiveAtomType) Band will perform SCF calculations at different screening cut-off values (see ScreeningCutOffs) and pick the cut-off value that maximizes the band gap. If multiple atom types are active, the screening cut-off optimizations are done one type at the time (in the same order as the ActiveAtomType blocks appear in the input).

ActiveAtomType

Type
Block

Recurring
True

Description
Use the DFT-1/2 method for the atom-type specified in this block.

AtomType

Type
String

Description
Atom-type to use. You can activate all atom-types by specifying 'All'.

IonicCharge

Type
Float

Default value
0.5

Description
The amount of charge to be removed from the atomic HOMO.

ScreeningCutOffs

Type
Float List

Default value
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0]

Unit
Bohr

Description
List of screening cut-offs (to screen the asymptotic IonicCharge/r potential). Band will loop over these values and find the cut-off that maximizes the band-gap. If only one number is provided, Band will simply use that value.

`Enabled`

Type
Bool

Default value
No

GUI name
Use method

Description
Whether the DFT-1/2 method will be used.

`Prepare`

Type
Bool

Default value
No

Description
Analyze the band structure to determine reasonable settings for an DFT-1/2 calculation. If this is possible the list of active atom types is written to the output. This can be used in a next run as the values for ActiveAtomType. The DFTHalf%Enabled key should be set to false

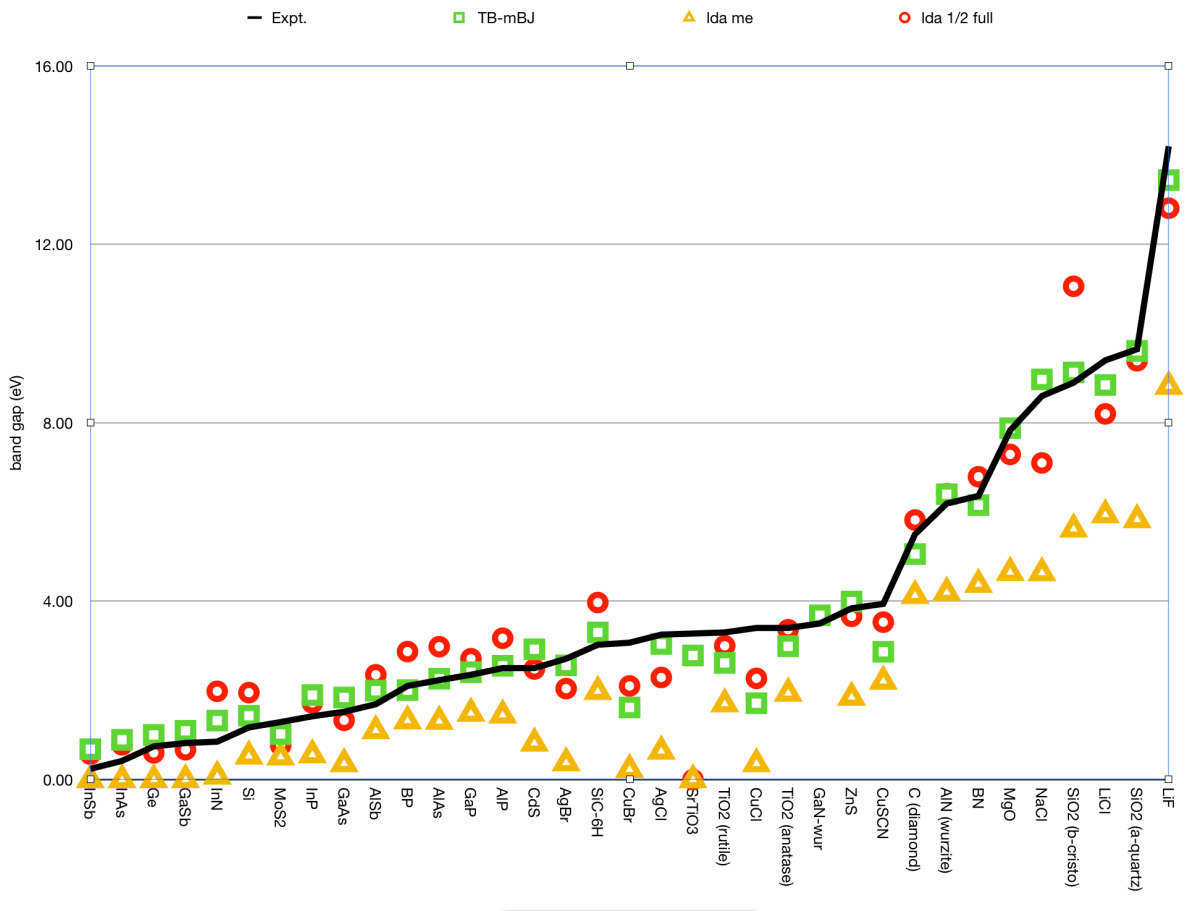
`SelfConsistent`

Type
Bool

Default value
Yes

Description
Apply the extra potential during the SCF, or only afterwards. Applying DFT-1/2 only post SCF increases the band gap, compared to the self-consistent one.

Here are some results showing that LDA-1/2 can work quite well, but that the TB-mBJ functional works better for this set.



3.2 Relativistic Effects and Spin

3.2.1 Spin polarization

By default Band calculations are spin-restricted. You can instruct Band to perform a spin-unrestricted via the `Unrestricted` key:

```
Unrestricted Yes/No
```

`Unrestricted`

Type
Bool

Default value
No

Description
Controls whether Band should perform a spin-unrestricted calculation. Spin-unrestricted calculations are computationally roughly twice as expensive as spin-restricted.

The orbitals are occupied according to the aufbau principle.

If you want to enforce a specific spin-polarization (instead of occupying according to the aufbau principle) you can use the `EnforcedSpinPolarization` key:

```
EnforcedSpinPolarization float
```

EnforcedSpinPolarization

Type
Float

GUI name
Spin polarization

Description
Enforce a specific spin-polarization instead of occupying according to the aufbau principle. The spin-polarization is the difference between the number of alpha and beta electron.

Thus, a value of 1 means that there is one more alpha electron than beta electrons.

The number may be anything, including zero, which may be of interest when searching for a spin-flipped pair, that may otherwise end up in the (more stable) parallel solution.

3.2.2 Relativistic Effects

Relativistic effects are treated with the accurate and efficient ZORA approach[1][2], controlled by the `Relativity` 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).

```
Relativity
   Level [None | Scalar | Spin-Orbit]
End
```

Relativity

Type
Block

Description
Options for relativistic effects.

Level

Type
Multiple Choice

Default value
Scalar

Options
[None, Scalar, Spin-Orbit]

GUI name
Relativity (ZORA)

Description
None: No relativistic effects.

Scalar: Scalar relativistic ZORA.

[1] 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) (https://doi.org/10.1103/PhysRevB.56.13556).

[2] 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) (https://doi.org/10.1103/PhysRevB.61.1773).

This option comes at very little cost.

SpinOrbit: 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 key.

See also the *SpinOrbitMagnetization* (page 22) key.

3.3 Solvation

Band offers two implicit solvent models, COSMO and SM12.

3.3.1 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[1].

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.

The solvent information is specified in the `solvent` key of the `solvation` block. The simplest option is to use one of the pre-defined solvents:

```
Solvation
   Enabled Yes/No
   Solvent
      Name [...]
   End
End
```

`Solvation`

Type
Block

Description
Options for the COSMO (Conductor like Screening Model) solvation model.

`Enabled`

Type
Bool

Default value
No

[1] 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) (https://doi.org/10.1039/P29930000799).

GUI name
Include COSMO solvation

Description
Use the Conductor like Screening Model (COSMO) to include solvent effects.

`Solvent`

Type
Block

Description
Solvent details

`Name`

Type
Multiple Choice

Default value
Water

Options
[AceticAcid, Acetone, Acetonitrile, Ammonia, Aniline, Benzene, BenzylAlcohol, Bromoform, Butanol, isoButanol, tertButanol, CarbonDisulfide, CarbonTetrachloride, Chloroform, Cyclohexane, Cyclohexanone, Dichlorobenzene, DiethylEther, Dioxane, DMFA, DMSO, Ethanol, EthylAcetate, Dichloroethane, EthyleneGlycol, Formamide, FormicAcid, Glycerol, HexamethylPhosphoramide, Hexane, Hydrazine, Methanol, MethylEthylKetone, Dichloromethane, Methylformamide, Methypyrrolidinone, Nitrobenzene, Nitrogen, Nitromethane, PhosphorylChloride, IsoPropanol, Pyridine, Sulfolane, Tetrahydrofuran, Toluene, Triethylamine, TrifluoroaceticAcid, Water]

GUI name
Solvent

Description
Name of a pre-defined solvent. A solvent is characterized by the dielectric constant (Eps) and the solvent radius (Rad).

This is the list of possible solvents and their corresponding `Eps` and `Rad` values:

Solvent 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

continues on next page

Table 3.1 – continued from previous page

Dichlorobenzene	C6H4Cl2	9.8	3.54
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	8.55	3.12
Water	H2O	78.39	1.93

Several other options can be defined in the `Solvation` block:

```
Solvation
   CVec [EXACT | FITPOT]
   Charge
      Conv float
      Corr Yes/No
      Iter integer
      Method [CONJ | INVER]
   End
   Enabled Yes/No
   Radii # Non-standard block. See details.
      ...
   End
   SCF [VAR | PERT | NONE]
   Solvent
      Del float
      Emp float
      Eps float
      Name [...]
```

(continues on next page)

(continued from previous page)

```
      Rad float
   End
   Surf [Delley | Wsurf | Asurf | Esurf | Klamt]
End
```

Solvation

Type
Block

Description
Options for the COSMO (Conductor like Screening Model) solvation model.

CVec

Type
Multiple Choice

Default value
EXACT

Options
[EXACT, FITPOT]

GUI name
Calculate Coulomb interaction

Description
Choose how to calculate the Coulomb interaction matrix between the molecule and the point charges on the surface:

- EXACT: use exact density, and integrate against the potential of the point charges. This may have inaccuracies when integration points are close to the point charges.

- FITPOT: evaluate the molecular potential at the positions of the point charges, and multiply with these charges.

Charge

Type
Block

Description
Select the algorithm to determine the charges.

Conv

Type
Float

Default value
1e-08

Description
Charge convergence threshold in iterative COSMO solution.

Corr

Type
Bool

Default value
Yes

GUI name
Correct for outlying charge

Description
Correct for outlying charge.

`Iter`

Type
Integer

Default value
1000

Description
Maximum number of iterations to solve COSMO equations.

`Method`

Type
Multiple Choice

Default value
CONJ

Options
[CONJ, INVER]

GUI name
Charge determination method

Description
INVER: matrix inversion, CONJ: biconjugate gradient method.

The CONJ method is guaranteed to converge with small memory requirements and is normally the preferred method.

`Enabled`

Type
Bool

Default value
No

GUI name
Include COSMO solvation

Description
Use the Conductor like Screening Model (COSMO) to include solvent effects.

`Radii`

Type
Non-standard block

Description
The values are the radii of the atomic spheres. If not specified the default values are those by Allinger. Format: 'AtomType value'. e.g.: 'H 0.7'

`SCF`

Type
Multiple Choice

Default value
VAR

Options
[VAR, PERT, NONE]

GUI name
Handle charges

Description
Determine the point charges either Variational (VAR) or after the SCF as a Perturbation (PERT).

`Solvent`

Type
Block

Description
Solvent details

`Del`

Type
Float

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

`Emp`

Type
Float

Description
Emp is the empirical scaling factor x for the energy scaling.

`Eps`

Type
Float

Description
User-defined dielectric constant of the solvent (overrides the Eps value of the solvent defined in 'Name')

`Name`

Type
Multiple Choice

Default value
Water

Options
[AceticAcid, Acetone, Acetonitrile, Ammonia, Aniline, Benzene, BenzylAlcohol, Bromoform, Butanol, isoButanol, tertButanol, CarbonDisulfide, CarbonTetrachloride, Chloroform, Cyclohexane, Cyclohexanone, Dichlorobenzene, DiethylEther, Dioxane, DMFA, DMSO, Ethanol, EthylAcetate, Dichloroethane, EthyleneGlycol, Formamide, FormicAcid, Glycerol, HexamethylPhosphoramide, Hexane, Hydrazine, Methanol, MethylEthylKetone, Dichloromethane, Methylformamide, Methypyrrolidinone, Nitrobenzene, Nitrogen, Nitromethane, PhosphorylChloride, IsoPropanol, Pyridine, Sulfolane, Tetrahydrofuran, Toluene, Triethylamine, TrifluoroaceticAcid, Water]

GUI name
Solvent

Description
Name of a pre-defined solvent. A solvent is characterized by the dielectric constant (Eps) and the solvent radius (Rad).

`Rad`

Type
Float

Unit
Angstrom

Description
User-defined radius of the solvent molecule (overrides the Rad value of the solvent defined in 'Name').

`Surf`

Type
Multiple Choice

Default value
Delley

Options
[Delley, Wsurf, Asurf, Esurf, Klamt]

GUI name
Surface type

Description
Within the COSMO model the molecule is contained in a molecule shaped cavity.

Select one of the following surfaces to define the cavity:

- Wsurf: Van der Waals surface

- Asurf: solvent accessible surface

- Esurf: solvent excluding surface

- Klamt: Klamt surface

- Delley: Delley surface.

3.3.2 Additional keys for periodic systems

For the simulation of periodic structures ICW solvation, you may specify the following options:

```
PeriodicSolvation
   RemovePointsWithNegativeZ Yes/No
   NStar integer
End
```

`PeriodicSolvation`

Type
Block

Description
Additional options for simulations of periodic structures with solvation.

RemovePointsWithNegativeZ

Type
Bool

Default value
No

GUI name
Only above slab

Description
Whether the 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 (in the Z direction), then this option should be set to 'True'

NStar

Type
Integer

Default value
4

Description
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 108) block). 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.

3.3.3 SM12: Solvation Model 12

Continuum solvation can be done with the Minnesota's Solvation Model 12 (SM12) (JCTC,9,609,2013 (https://pubs.acs.org/doi/10.1021/ct300900e)). Details on the implementation of SM12 in ADF can be found in Ref. (JCTC,12,4033,2016 (https://pubs.acs.org/doi/10.1021/acs.jctc.6b00410)). The energetics of solvation is calculated using:

$$\Delta G_S^{\otimes} = \Delta E_E + G_P + G_{CDS} + \Delta G_N + \Delta G_{conc}^{\otimes}$$

where the symbol $\otimes$ denotes an arbitrary choice of standard states, ΔE_E is the change in the solute's internal electronic energy in transferring from the gas phase to the liquid phase at the same geometry, G_P is the polarization free energy of the solute–solvent system when the solute is inserted, G_{CDS} is the component of the free energy that is nominally associated with cavitation, dispersion, and solvent structure, ΔG_N is the change in ΔG_S^0 due to a change in nuclear coordinates, and $\Delta G_{conc}^{\otimes}$ accounts for the difference in concentrations, if any, in the gas-phase standard state and the solution-phase one. In case of 1 M concentration in both solution and gas, then $\Delta G_S^{\otimes}$ = 0 kcal/mol, which yields ΔG_S^*. If the same geometry is used in solution and gas phase calculation, then ΔG_N is zero.

SM12 makes use of the Generalized Born approximation to calculate the bulk electrostatic contribution. This is comprised of several terms that are together known as the ENP (Electronic, Nuclear, and Polarization) term G_P. The SM12 model in Band uses CM5. CM5 is a class 4 charge model, making use of both empirical and density related terms. It is comprised of Hirshfeld charges, a simple bond order calculation, atomic distances, and atom specific parameters. The covalent radii utilized are based on the atomic covalent radius from the Handbook of Chemistry and Physics. The Coulomb integral is described with the use of an approximation from Still et al.. Several parameters go into describing this, which include:

inter atomic distance, an empirical Born constant, and the Born area, which is calculated with the Analytical Surface Area (ASA) algorithm. The Born area is calculated using Legendre-Gauss quadrature from the atomic radii to a sphere that encapsulates the entire molecule.

The ASA algorithm is also used to calculate the solvent accessible surface area (SASA), which is computed within the CDS (Cavitation, Dispersion, Solvation) term of SM12. The CDS term depends on three terms:

- SASA (ASA Algorithm)
- Atomic surface tension
- Macroscopic surface tension

Atomic surface tension is based on atom-atom distances and the solvent. Macroscopic surface tension is solvent specific. The SM12 implementation in Band reports energies in an atom specific way. You can attribute exact CDS and polarization energies to each atom in your solute. The parameters for SM12 are derived to explicitly incorporate organic elements (N, C, O, F, Si, P, S, Cl, Br, I), with less emphasis on non-organics. Also, while most solvents have a generic atomic surface tension reliance for atoms, water has it's own explicit set of parameters to better describe it.

Input

The minimal input for the SM12 method is the following:

```
SolvationSM12
   Enabled Yes/No
   Solv [...]
End
```

SolvationSM12

Type
Block

Description
Options for Solvation Model 12 (SM12).

Enabled

Type
Bool

Default value
No

GUI name
Include SM12 solvation

Description
Whether to use the Solvation Model 12 (SM12) in the calculation.

Solv

Type
Multiple Choice

Default value
WATER

Options
[ACETICACID, ACETONITRILE, ACETOPHENONE, ANILINE, ANISOLE, BENZENE, BENZONITRILE, BENZYLALCOHOL, BROMOBENZENE, BROMOETHANE, BROMOFORM, BROMOOCTANE, N-BUTANOL, SEC-BUTANOL, BUTANONE,

BUTYLACETATE, N-BUTYLBENZENE, SEC-BUTYLBENZENE, T-BUTYLBENZENE, CARBONDISULFIDE, CARBONTETRACHLORIDE, CHLOROBENZENE, CHLOROFORM, CHLOROHEXANE, M-CRESOL, CYCLOHEXANE, CYCLOHEXANONE, DECALIN, DECANE, DECANOL, 1-2-DIBROMOETHANE, DIBUTYLETHER, O-DICHLOROBENZENE, 1-2-DICHLOROETHANE, DIETHYLETHER, DIISOPROPYLETHER, N-N-DIMETHYLACETAMIDE, N-N-DIMETHYLFORMAMIDE, 2-6-DIMETHYLPYRIDINE, DIMETHYLSULFOXIDE, DODECANE, ETHANOL, ETHOXYBENZENE, ETHYLACETATE, ETHYLBENZENE, FLUOROBENZENE, 1-FLUORO-N-OCTANE, HEPTANE, HEPTANOL, HEXADECANE, HEXADECYLIODIDE, HEXANE, HEXANOL, IODOBENZENE, ISOBUTANOL, ISOOCTANE, ISOPROPANOL, ISOPROPYLBENZENE, P-ISOPROPYLTOLUENE, MESITYLENE, METHANOL, METHOXYETHANOL, METHYLENECHLORIDE, N-METHYLFORMAMIDE, 2-METHYLPYRIDINE, 4-METHYL-2-PENTANONE, NITROBENZENE, NITROETHANE, NITROMETHANE, O-NITROTOLUENE, NONANE, NONANOL, OCTANE, OCTANOL, PENTADECANE, PENTANE, PENTANOL, PERFLUOROBENZENE, PHENYLETHER, PROPANOL, PYRIDINE, TETRACHLOROETHENE, TETRAHYDROFURAN, TETRAHYDROTHIOPHENEDIOXIDE, TETRALIN, TOLUENE, TRIBUTYLPHOSPHATE, TRIETHYLAMINE, 1-2-4-TRIMETHYLBENZENE, UNDECANE, WATER, XYLENE, 1-2-DIBROMOETHANE_WATER, 1-2-DICHLOROETHANE_WATER, BENZENE_WATER, CARBONTETRACHLORIDE_WATER, CHLOROBENZENE_WATER, CHLOROFORM_WATER, CYCLOHEXANE_WATER, DIBUTYLETHER_WATER, DIETHYLETHER_WATER, ETHYLACETATE_WATER, HEPTANE_WATER, HEXANE_WATER, NITROBENZENE_WATER, OCTANOL_WATER]

GUI name
Solvent

Description
List of predefined solvents

This is the full list of input options for the SM12 method:

```
SolvationSM12
   ARO float
   Acid float
   Base float
   BornC float
   BornRadiusConfig
      MaxCellDistance float
      PointsPerBohr integer
      UseLegendreGrid Yes/No
   End
   Chgal float
   Cust string
   Debug string
   EPS float
   Enabled Yes/No
   HALO float
   Kappa float
   PostSCF Yes/No
   PrintSM12 Yes/No
   RadSolv float
   Ref float
   Solv [...]
   Tens float
   TopologicalExtrapolation
```

(continues on next page)

(continued from previous page)

```
      FirstCell integer
      LastCell integer
      Order integer
   End
End
```

SolvationSM12

Type
Block

Description
Options for Solvation Model 12 (SM12).

ARO

Type
Float

Default value
0.0

Description
Square of the fraction of non-hydrogen atoms in the solvent that are aromatic carbon atoms (carbon aromaticity)

Acid

Type
Float

Default value
0.82

Description
Abraham hydrogen bond acidity parameter

Base

Type
Float

Default value
0.35

Description
Abraham hydrogen bond basicity parameter

BornC

Type
Float

Default value
3.7

Description
Coulomb constant for General Born Approximation

BornRadiusConfig

Type
Block

Description

MaxCellDistance

Type
Float

Default value
30.0

Unit
Bohr

Description
Max distance from the centra cell used when computing the Born radii for periodic systems

PointsPerBohr

Type
Integer

Default value
10

Description

UseLegendreGrid

Type
Bool

Default value
Yes

Description

Chgal

Type
Float

Default value
2.474

Description
Exponential of Pauli's bond order

Cust

Type
String

Description
Custom solvent input

Debug

Type
String

Description
Prints a lot of information about every pass on CDS and ENP code, keywords: ENP, CDS

EPS

Type
Float

Default value
78.36

Description
The dielectric constant

`Enabled`

Type
Bool

Default value
No

GUI name
Include SM12 solvation

Description
Whether to use the Solvation Model 12 (SM12) in the calculation.

`HALO`

Type
Float

Default value
0.0

Description
Square of the fraction of non-hydrogen atoms in the solvent molecule that are F, Cl, or Br (electronegative halogenicity)

`Kappa`

Type
Float

Default value
0.0

Description
Factor for Debye screening

`PostSCF`

Type
Bool

Default value
No

Description
Whether to apply the solvation potential during the SCF or only calculate the solvation energy after the SCF.

`PrintSM12`

Type
Bool

Default value
No

Description
Prints out an in-depth breakdown of solvation energies

`RadSolv`

Type
Float

Default value
0.4

Description
The radius distance between the solute and solvent

`Ref`

Type
Float

Default value
1.3328

Description
Refractive index of solvent

`Solv`

Type
Multiple Choice

Default value
WATER

Options
[ACETICACID, ACETONITRILE, ACETOPHENONE, ANILINE, ANISOLE, BENZENE, BENZONITRILE, BENZYLALCOHOL, BROMOBENZENE, BROMOETHANE, BROMOFORM, BROMOOCTANE, N-BUTANOL, SEC-BUTANOL, BUTANONE, BUTYLACETATE, N-BUTYLBENZENE, SEC-BUTYLBENZENE, T-BUTYLBENZENE, CARBONDISULFIDE, CARBONTETRACHLORIDE, CHLOROBENZENE, CHLOROFORM, CHLOROHEXANE, M-CRESOL, CYCLOHEXANE, CYCLOHEXANONE, DECALIN, DECANE, DECANOL, 1-2-DIBROMOETHANE, DIBUTYLETHER, O-DICHLOROBENZENE, 1-2-DICHLOROETHANE, DIETHYLETHER, DIISOPROPYLETHER, N-N-DIMETHYLACETAMIDE, N-N-DIMETHYLFORMAMIDE, 2-6-DIMETHYLPYRIDINE, DIMETHYLSULFOXIDE, DODECANE, ETHANOL, ETHOXYBENZENE, ETHYLACETATE, ETHYLBENZENE, FLUOROBENZENE, 1-FLUORO-N-OCTANE, HEPTANE, HEPTANOL, HEXADECANE, HEXADECYLIODIDE, HEXANE, HEXANOL, IODOBENZENE, ISOBUTANOL, ISOOCTANE, ISOPROPANOL, ISOPROPYLBENZENE, P-ISOPROPYLTOLUENE, MESITYLENE, METHANOL, METHOXYETHANOL, METHYLENECHLORIDE, N-METHYLFORMAMIDE, 2-METHYLPYRIDINE, 4-METHYL-2-PENTANONE, NITROBENZENE, NITROETHANE, NITROMETHANE, O-NITROTOLUENE, NONANE, NONANOL, OCTANE, OCTANOL, PENTADECANE, PENTANE, PENTANOL, PERFLUOROBENZENE, PHENYLETHER, PROPANOL, PYRIDINE, TETRACHLOROETHENE, TETRAHYDROFURAN, TETRAHYDROTHIOPHENEDIOXIDE, TETRALIN, TOLUENE, TRIBUTYLPHOSPHATE, TRIETHYLAMINE, 1-2-4-TRIMETHYLBENZENE, UNDECANE, WATER, XYLENE, 1-2-DIBROMOETHANE_WATER, 1-2-DICHLOROETHANE_WATER, BENZENE_WATER, CARBONTETRACHLORIDE_WATER, CHLOROBENZENE_WATER, CHLOROFORM_WATER, CYCLOHEXANE_WATER, DIBUTYLETHER_WATER,

DIETHYLETHER_WATER, ETHYLACETATE_WATER, HEPTANE_WATER, HEXANE_WATER, NITROBENZENE_WATER, OCTANOL_WATER]

GUI name
Solvent

Description
List of predefined solvents

`Tens`

Type
Float

Default value
103.62

Description
Macroscopic surface tension of the solvent at the air/solvent interface at 298K (cal*mol^-1*Ang^-2)

`TopologicalExtrapolation`

Type
Block

Description
Method to extrapolate the long range Coulomb potential, needed for periodic calculations

`FirstCell`

Type
Integer

Default value
5

Description
First cell for the topological extrapolation of the long range part of the Coulomb Potential.

`LastCell`

Type
Integer

Default value
10

Description
Last cell for the topological extrapolation of the long range part of the Coulomb Potential.

`Order`

Type
Integer

Default value
3

Description
Order of the topological extrapolation of the long range part of the Coulomb Potential.

3.4 Electric and Magnetic Fields

3.4.1 Electric Field

The external electric field is handled at the AMS level, see the documentation there.

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 32) calculations, the spin is assumed to be aligned with the z-axis.

3.4.2 Magnetic Field

```
BField
   Bx float
   By float
   Bz float
   Dipole Yes/No
   DipoleAtom integer
   Method [NR_SDOTB | NR_LDOTB | NR_SDOTB_LDOTB]
   Unit [tesla | a.u.]
End
```

BField

Type
Block

Description
The effect of a magnetic filed can be approximated by the following potential: mu * sigma_i * B, where mu is the Bohr magneton, sigma_i are the Pauli matrices and B is the magnetic field

Bx

Type
Float

Default value
0.0

Unit
Tesla

Description
Value of the x component of the BField

By

Type
Float

Default value
0.0

Unit
Tesla

Description
Value of the y component of the BField

Bz

Type
Float

Default value
0.0

Unit
Tesla

Description
Value of the z component of the BField

Dipole

Type
Bool

Default value
No

GUI name
Bfield is: Atomic dipole

Description
Use an atomic dipole as magnetic field instead of a uniform magnetic field.

DipoleAtom

Type
Integer

Default value
1

GUI name
on atom number

Description
Atom on which the magnetic dipole should be centered (if using the dipole option)

Method

Type
Multiple Choice

Default value
NR_SDOTB

Options
[NR_SDOTB, NR_LDOTB, NR_SDOTB_LDOTB]

Description
There are two terms coupling to an external magnetic field.

One is the intrinsic spin of the electron, called S-dot-B, the other one is the orbital momentum call L-dot-B.

The L.B is implemented non-relativistically, using GIAOs in the case of a homogeneous magnetic field (not for the dipole case).

Unit

Type
Multiple Choice

Default value
tesla

Options
[tesla, a.u.]

Description
Unit of magnetic filed. The a.u. is the SI version of a.u.

3.4.3 Atom-wise fuzzy potential

```
FuzzyPotential # Non-standard block. See details.
   ...
End
```

FuzzyPotential

Type
Non-standard block

Description
Atomic (fuzzy cell) based, external, electric potential. See example.

Example:

```
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 71). 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.)

3.5 Nuclear Model

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

`NuclearModel`

Type
Multiple Choice

Default value
PointCharge

Options
[PointCharge, Gaussian, Uniform]

Description
Specify what model to use for the nucleus.

For the Gaussian model the nuclear radius is calculated according to the work of Visscher and Dyall (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))

CHAPTER

FOUR

ACCURACY AND EFFICIENCY

Given a *Model Hamiltonian* (page 15), the most important aspects determining the accuracy of a Band calculation are:

- *Basis set* (page 54)
- *K-Space* (page 66)

Also important, but to a lesser degree, are the following aspects:

- *Numerical Integration (BeckeGrid)* (page 71)
- *Density fitting (ZlmFit)* (page 75)
- *Basis-set confinement (SoftConfinement)* (page 62)
- *SCF convergence (Convergence)* (page 84)
- *Hartree–Fock Resolution of the Identity (RIHartreeFock)* (page 79) (only for hybrid functionals)
- *For Many-body-perturbation theory (MBPT) options* (page 100) (only for MBPT calculations)

The CPU time and memory requirements strongly depend on these options, as does the accuracy of the results.

General NumericalQuality

A simple way of tweaking the accuracy of the calculation is via the **NumericalQuality** key. This sets the quality of several technical aspects of a Band calculation (with the notable exception of the *basis set* (page 54))

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

`NumericalQuality`

Type
Multiple Choice

Default value
Normal

Options
[Basic, Normal, Good, VeryGood, Excellent]

Description
Set the quality of several important technical aspects of a BAND calculation (with the notable exception of the basis set). It sets the quality of: BeckeGrid (numerical integration), ZlmFit (density fitting), KSpace (reciprocal space integration), and SoftConfinement (basis set confinement). Note: the quality defined in the block of a specific technical aspects supersedes the value defined in NumericalQuality (e.g. if I specify 'NumericalQuality Basic' and 'BeckeGrid%Quality Good', the quality of the BeckeGrid will be 'Good')

4.1 Basis set

Band represents the single-determinant electronic wave functions as a linear combinations of atom-centered basis functions (the basis set). See also: Wikipedia page on Basis Sets (https://en.wikipedia.org/wiki/Basis_set_(chemistry)).

The basis sets in Band consists of **NAOs** (Numerical Atomic Orbitals, obtained by solving numerically the Kohn-Sham equations for the isolated spherical atoms) augmented by a set of **STOs** (Slater Type Orbitals).

The choice of basis set is very important, as it influences heavily the accuracy, the CPU time and the memory usage of the calculation. Band comes with 6 predefined types of basis sets: **SZ**, **DZ**, **DZP**, **TZP**, **TZ2P**, **QZ4P** (SZ: Single Zeta, DZ: Double Zeta, DZP: Double Zeta + Polarization, TZP: Triple Zeta + Polarization, TZ2P: Triple Zeta + Double Polarization, QZ4P: Quadruple Zeta + Quadruple Polarization). See the sections *Which basis set should I use?* (page 55) and *Available Basis Sets* (page 58) for more details.

To speed up the calculation, Band can use the **frozen core approximation** in which core orbitals are kept frozen during the SCF procedure (and the valence orbitals are orthogonalized against the frozen orbitals). One can run an **all electron** calculation by specifying `Core None` in the Basis input block. Note: some features, such as *Hybrid functionals* (page 24), are incompatible with the frozen-core approximation, and require an all electron (i.e. `Core None`) basis set.

4.1.1 Basis input block

You can specify which basis set to use in the `Basis` input block.

```
Basis
   Type [SZ | DZ | DZP | TZP | TZ2P | QZ4P | STO/TZ2P | STO/SZ | STO/DZ | STO/DZP |␣
↪STO/QZ4P |
         CORR/QZ6P | CORR/TZ3P | GTO/CC-PV5Z | GTO/DEF2-QZVPPD | GTO/CC-PV6Z | GTO/CC-
↪PVQZ |
         GTO/CC-PVTZ | GTO/CC-PVDZ | GTO/DEF2-SVP | GTO/DEF2-TZVP | GTO/DEF2-TZVPP |␣
↪GTO/DEF2-QZVP |
         GTO/AUG-CC-PVDZ | GTO/AUG-CC-PVTZ | GTO/AUG-CC-PVQZ | GTO/POB-TZVP]
   Core [None | Small | Medium | Large]
End
```

`Basis`

Type
Block

Description
Definition of the basis set

`Type`

Type
Multiple Choice

Default value
DZ

Options
[SZ, DZ, DZP, TZP, TZ2P, QZ4P, STO/TZ2P, STO/SZ, STO/DZ, STO/DZP, STO/QZ4P, CORR/QZ6P, CORR/TZ3P, GTO/CC-PV5Z, GTO/DEF2-QZVPPD, GTO/CC-PV6Z, GTO/CC-PVQZ, GTO/CC-PVTZ, GTO/CC-PVDZ, GTO/DEF2-SVP, GTO/DEF2-TZVP, GTO/DEF2-TZVPP, GTO/DEF2-QZVP, GTO/AUG-CC-PVDZ, GTO/AUG-CC-PVTZ, GTO/AUG-CC-PVQZ, GTO/POB-TZVP]

GUI name
Basis set

Description
Select the basis set to use.

SZ : Single Z

DZ : Double Z

DZP : Double Z, 1 polarization function

TZP : Triple Z, 1 polarization function

TZ2P : Triple Z, 2 polarization functions

QZ4P : Quadruple Z, 4 polarization function

The basis set chosen will apply to all atoms in your structure. If a matching basis is not found a better type might be used.

`Core`

Type
Multiple Choice

Default value
Large

Options
[None, Small, Medium, Large]

GUI name
Frozen core

Description
Select the size of the frozen core you want to use.

Small, Medium, and Large will be interpreted within the basis sets available (of the selected quality), and might refer to the same core in some cases.

4.1.2 Which basis set should I use?

The hierarchy of basis sets, from the smallest and least accurate (SZ) to the largest and most accurate (QZ4P), is **SZ < DZ < DZP < TZP < TZ2P < QZ4P**.

The choice of basis set is in general a trade off between accuracy and computation time: the more accurate the basis set, the more computationally demanding the calculation will be (both in term of CPU time and the memory usage).

As an example, in the following table we compare accuracy v.s. CPU time for a (24,24) carbon nanotube using different basis sets. "Energy error" is defined as the absolute error in the formation energy per atom using the QZ4P results as reference.[1]

[1] Computational details: Single Point calculation, 'Good' NumericalQuality, no frozen core, 7 k-points, XC functional: GGA:PBE. Calculation performed on a 24 cores compute node. 96 atoms in the unit cell.

Table 4.1: Accuracy and CPU time ratio for a (24,24) carbon nanotube using different basis sets

Basis Set	Energy Error [eV]	CPU time ratio (relative to SZ)
SZ	1.8	1
DZ	0.46	1.5
DZP	0.16	2.5
TZP	0.048	3.8
TZ2P	0.016	6.1
QZ4P	reference	14.3

It is worthwhile noting that the error in formation energies are to some extend systematic, and they partially cancel each other out when taking energy differences. For example, if one considers the difference in energy between two carbon nanotubes variants ((24,24) and (24-0)) with the same number of atoms, the basis set error is smaller than 1 milli-eV/atom already with a DZP basis set, which is much smaller than the absolute error in the individual energies. The same consideration holds for reactions barriers: the error in the energy difference between different conformations is much smaller than the error in the absolute energies themselves.

Band gaps:

The following figure shows the convergence WRT basis set of band gaps (XC:PBE). While DZ is often inaccurate (since DZ lacks any polarization function, the description of the virtual orbital space is very poor), TZP captures the trends very well.

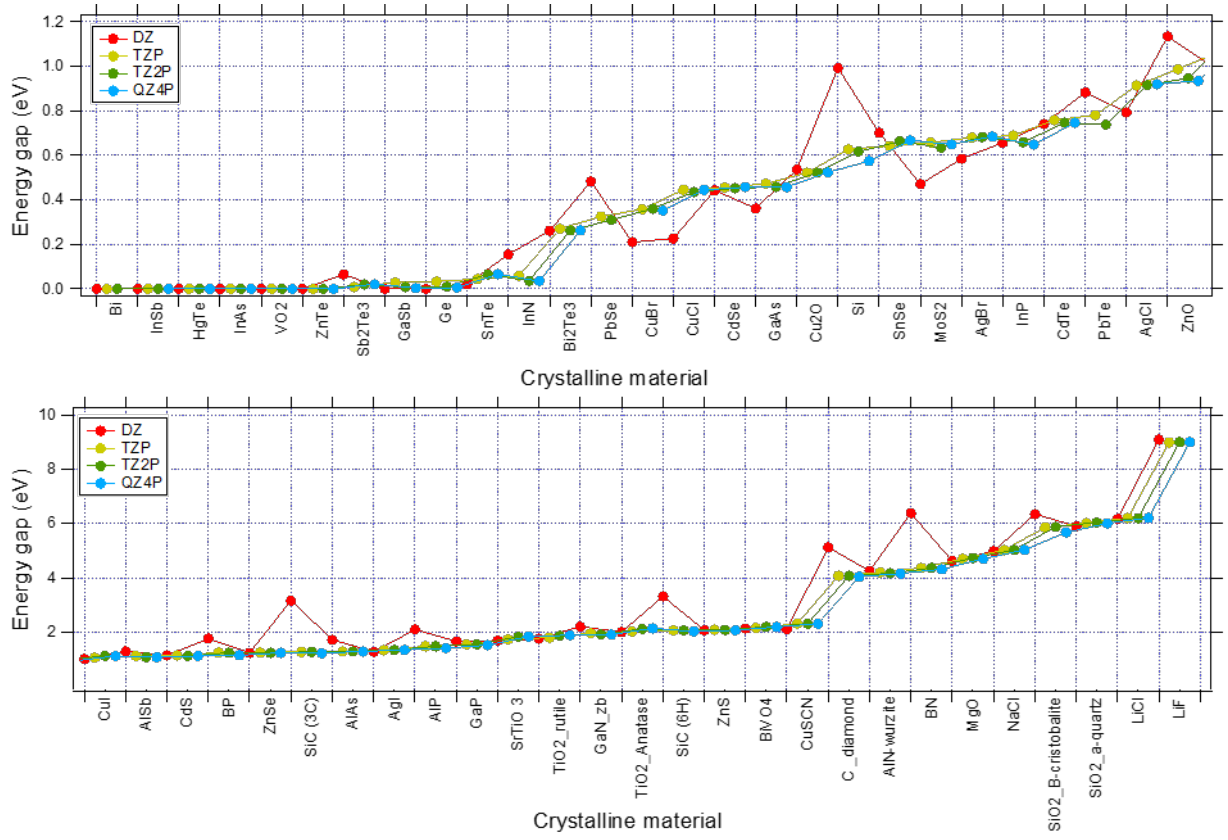


Fig. 4.1: Convergence of Band Gaps WRT basis set for various system. (XC:PBE, K-Space Quality: Good)

In general, since the basis set might have different effects on different properties, it is advisable to run a few simple

calculations to get an idea of the effect of the basis set with your property of interest.

A summary of the basis sets:

- **SZ**: Single Zeta, the minimal basis set (only NAOs), serves mostly a technical purpose. The results are rather inaccurate, but it's computationally efficient. It can be useful for running a very quick test calculation. See also *SZ-SCF-Restart* (page 199).
- **DZ**: The Double Zeta basis set is computationally very efficient. It can be used for pre-optimization of structures (that should then be further optimized with a better basis set). Since it has no polarization functions, properties depending on the virtual orbital space will be rather inaccurate.
- **DZP**: Double zeta plus polarization function. Only available for main group elements up to Krypton. For other elements a TZP basis set will be used **automatically**. This is a reasonably good basis set for geometry optimizations of organic systems.
- **TZP**: The Triple Zeta plus Polarization basis set offers the best balance between performance and accuracy. This is the basis set we would generally recommend.
- **TZ2P**: The Triple Zeta plus Double Polarization basis set is an accurate basis set. It is qualitatively similar to TZP but quantitatively better. It should be used when a good description of the virtual orbital space is needed.
- **QZ4P**: Quadruple zeta plus Quadruple Polarization. This is the biggest basis set available. It can be used for benchmarking.

4.1.3 Frozen core

It is recommended to use the frozen core approximation, as it is faster, in particular for heavy elements.

In general, the frozen core approximation does not influence the results significantly (especially if one uses a `small` frozen core). For accurate results on certain properties (like *Properties at Nuclei* (page 130)) all electron basis sets are needed on the atoms of interest.

- For Meta-GGA XC functionals, it is recommended to use `small` or `none` frozen core (the frozen orbitals are computed using LDA and not the selected Meta-GGA)
- For optimizations under pressure, use `small` or `none` frozen core

Small, Medium, and Large frozen cores:

The possible choices for the `Basis%Core` keyword are `None`, `Small`, `Medium`, and `Large`. How does this map to the actual *Available Basis Sets* (page 58)? Obviously this depends on the element. As all elements have an all electron basis set defined, `None` maps always to this set. For the element H there are no frozen core sets available and all options can only lead to the all electron basis set. For elements like C there is a single frozen core available and the three choices (`Small`, `Medium`, `Large`) lead to the same choice: C.1s. Most elements have two frozen cores defined. In such a case the choices `Large` and `Medium` point to the larger one, and `Small` to the smallest one. With four choices `Large` maps to the largest one, `Medium` to the one below it, and `Small` to the smallest one. This table summarizes the logic

Table 4.2: Mapping of the `Basis%Core` keyword to available cores (if any)

#Available frozen cores	Example	None	Small	Medium	Large
0	H	H	H	H	H
1	C C.1s	C	C.1s	C.1s	C.1s
2	Na Na.1s Na.2p	Na	Na.1s	Na.2p	Na.2p
3	Rb Rb.3p Rb.3d Rb.4p	Rb	Rb.3p	Rb.3d	Rb.4p
4	Pb Pb.4d Pb.4f Pb.5p Pb.5d	Pb	Pb.4d	Pb.5p	Pb.5d

It is possible to control both the basis and core per atom type and per region, see *More Basis input options* (page 60).

4.1.4 Available Basis Sets

The basis set files, containing the definition of the basis set, are located in `$AMSHOME/atomicdata/Band`.

The next table gives an indication frozen core (*fc*) standard basis sets are available for the different elements in BAND. Note that all electron (*ae*) basis set are available for all basis sets types.

Table 4.3: Available standard basis sets for non-relativistic and ZORA calculations H-Ubn (Z=1-120)

Element	frozen core	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 .1s .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-Og (Z=104-118)	ae .5d .5f	Yes		Yes
Uue-Ubn (Z=119-120)	ae .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

Note: Not all combinations of basis set `Type` and `Core` are available for all elements. If a specific combination is not available, Band will pick the first *better* basis set. E.g. if an element is not in the DZP database, it is taken from the TZP basis set.

4.1.5 Pure STO and GTO basis sets

Starting from AMS2023, BAND includes many basis sets to reproduce literature data. The basis sets are suitable (and recommended) for well converged *GW* (page 132), *RPA* (page 24), and *double hybrid* (page 24) calculations in which electron-electron interaction effects are accounted for explicitly beyond DFT. Available basis sets are:

pure STO type basis sets

The TZ2P and QZ4P basis sets from ADF can now be used in BAND calculations as well. To get the same results as ADF, it is additionally necessary to disable the confinement of the basis functions *Confinement of basis functions* (page 62). Additionally, the CORR/TZ3P and the CORR/QZ6P basis sets of ADF can also be used with BAND. Band also includes the Corr/TZ3P_G and the Corr/QZ6P_GH basis sets which includes higher-angular momentum functions (g-functions fo TZ3P_G, g-,h- (and sometimes i-)functions for QZ6P_GH).

```
Basis Type=STO/TZ2PSTO
SoftConfinement Quality=Excellent # disable radial basis function confinement
```

Table 4.4: STO (all electron) basis sets for band

Basis Set	ADF
STO/TZ2PSTO	ZORA/TZ2P
STO/QZ4PSTO	ZORA/QZ4P
CORR/QZ6P	CORR/QZ6P
CORR/QZ6P_GH	
CORR/TZ3P	CORR/TZ3P
CORR/TZ3P_G	

Correlation consistent Gaussian type basis sets

See also:

Examples GTO (page 239)

BAND includes the correlation-consistent (CC) Dunning Gaussian type (GTO) basis sets from double Zeta (DZ) to hextuple Zeta (6Z) quality. GTO basis sets can only be used in non-relativistic calculations and can not be used with the frozen core approximation. The basis sets cc-pVDZ to cc-pV5Z are only available for the first four rows of the periodic table, with the exception of K (potassium) (see also: the information on the Basis Set Exchange (https://www.basissetexchange.org/). The cc-pV6Z basis set is only available for the elements of the first three rows of the periodic table with the exception of Li, Na, and Mg.

```
Basis Type=GTO/CC-PVTZ
SoftConfinement Quality=Excellent # disable radial basis function confinement
Relativity Level=None   # relativistic effects are currently not supported for GTOs
```

Table 4.5: GTO (all electron) basis sets for band

Basis Set	comment
GTO/CC-PV5Z	
GTO/CC-PV6Z	
GTO/CC-PVDZ	
GTO/CC-PVQZ	
GTO/CC-PVTZ	
GTO/CC-PWCVQZ	

GTO type basis sets can be used to calculate high-quality RPA and MP2/double-hybrid energies as well as GW QP

energies (MBPT based methods), often in combination with complete basis set (CBS) limit extrapolation of the form

$$E_{\text{CBS}} = \frac{Y^3 E_Y - X^3 E_X}{Y^3 - X^3}$$

where E_Y denotes an energy or QP energy calculated with the cc-pVYZ basis set and E_X denotes an energy or QP energy calculated with the cc-pVXZ basis set, and $Y > X$. Typically, an extrapolation from a cc-pVTZ and a cc-pVQZ calculation will give a result of cc-pV5Z quality at lower computational cost.

When used in conjunction with MBPT based methods, a highly accurate auxiliary fit set is needed (Also see the recommendations in the *MBPT section* (page 100)). It is recommended to generate the auxiliary fit sets directly from products of basis functions to saturate all higher angular momentum channels. See the *RIHartreeFock documentation* (page 79).

4.1.6 More Basis input options

```
Basis
   Folder string
   PerAtomType
      Core [None | Small | Medium | Large]
      File string
      Symbol string
      Type [SZ | DZ | DZP | TZP | TZ2P | QZ4P]
   End
   PerRegion
      Core [None | Small | Medium | Large]
      Region string
      Type [SZ | DZ | DZP | TZP | TZ2P | QZ4P]
   End
End
```

Basis

Type
Block

Description
Definition of the basis set

Folder

Type
String

Description
Path to a folder containing the basis set files. This can be used for special use-defined basis sets. Cannot be used in combination with 'Type'

PerAtomType

Type
Block

Recurring
True

Description
Defines the basis set for all atoms of a particular type.

Core

Type
Multiple Choice

Options
[None, Small, Medium, Large]

Description
Size of the frozen core.

File

Type
String

Description
The path to the basis set file. The path can be absolute or relative to $AMSRESOURCES/Band. Specifying the path to the basis file explicitly overrides the automatic basis file selection via the Type and Core subkeys.

Symbol

Type
String

Description
The symbol for which to define the basis set.

Type

Type
Multiple Choice

Options
[SZ, DZ, DZP, TZP, TZ2P, QZ4P]

Description
The basis sets to be used.

PerRegion

Type
Block

Recurring
True

Description
Defines the basis set for all atoms in a region. If specified, this overwrites the values set with the Basis%Type and Basis%PerAtomType keywords for atoms in that region. Note that if this keyword is used multiple times, the chosen regions may not overlap.

Core

Type
Multiple Choice

Default value
Large

Options
[None, Small, Medium, Large]

Description
Size of the frozen core.

Region

Type
String

Description
The identifier of the region for which to define the basis set. Note that this may also be a region expression, e.g. 'myregion+myotherregion' (the union of two regions).

Type

Type
Multiple Choice

Default value
DZ

Options
[SZ, DZ, DZP, TZP, TZ2P, QZ4P]

Description
The basis sets to be used.

See also: *Example: Multiresolution* (page 230), and *Frozen core* (page 57).

4.1.7 Confinement of basis functions

It is possible to alter the radial part of the basis functions in order to make them more compact, which will in turn speeds up the calculation.

```
SoftConfinement
   Quality [Auto | Basic | Normal | Good | VeryGood | Excellent]
   Radius float
   Delta float
End
```

SoftConfinement

Type
Block

Description
In order to make the basis functions more compact, the radial part of the basis functions is multiplied by a Fermi-Dirac (FD) function (this 'confinement' is done for efficiency and numerical stability reasons). 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.

Quality

Type
Multiple Choice

Default value
Auto

Options
[Auto, Basic, Normal, Good, VeryGood, Excellent]

GUI name
Confinement

Description
In order to make the basis functions more compact, the radial part of the basis functions is multiplied by a Fermi-Dirac (FD) function (this 'confinement' is done for efficiency and numerical stability reasons). 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.

This key sets the two parameters 'Radius' and 'Delta'.

Basic: Radius=7.0, Delta=0.7;

Normal: Radius=10.0, Delta=1.0;

Good: Radius=20.0, Delta=2.0;

VeryGood and Excellent: no confinement at all.

If 'Auto', the quality defined in the 'NumericalQuality' will be used.

`Radius`

Type
Float

Unit
Bohr

Description
Explicitly specify the radius parameter of the Fermi-Dirac function.

`Delta`

Type
Float

Unit
Bohr

Description
Explicitly specify the delta parameter of the Fermi-Dirac function (if not specified, it will be 0.1*Radius).

- For geometry optimizations under pressure, `Basic` soft confinement is recommended.

4.1.8 Manually specifying AtomTypes (expert option)

`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 n^{th} type, which in turn has atoms at positions defined by the n^{th} `Atoms` key.

```
AtomType ElementSymbol
   Dirac ChemSym
      {option}
      ...
      shells cores
      shell_specification {occupation_number}
      ...
   End
```

(continues on next page)

(continued from previous page)

```
   {BasisFunctions
      shell_specification STO_exponent
      ...
   End}
   FitFunctions
      shell_specification STO_exponent
      ...
   End
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
```

(continues on next page)

(continued from previous page)

```
      2s
      2p 2.0
   End
   BasisFunctions
      1s 1.7
      ...
   End
   FitFunctions
      1s 13.5
      2s 11.0
      ...
   End
End
```

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.

4.1.9 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

$E(A+B) - E(\text{A with B as ghost}) - E(\text{B with A as ghost})$

The BSSE correction is

$E(A) - E(\text{A with B as ghost}) + E(B) - E(\text{B with A as ghost})$

See also:

Example: BSSE correction (page 232)

4.1.10 Alternative elements / Virtual crystal approximation

It is possible to define an alternative nuclear charge for an element. If a certain site in a crystal has, say, a 50% occupation of C (Z=6) and a 50% occupation of B (Z=5) you can use one alternative atom with Z=5.5

Example:

```
Atoms
   Si 0.0 0.0 0.0
   C  0.0 0.0 0.0 nuclear_charge=5.5   ! this site has a mixture of 50% C and B
End
```

In this example the basis set is taken from the C atom, but you could equally well specify the B atom here. (In fact any atom type can be specified here, but why would you like to use an Au basis set for this situation.). Defining such an average element is in the spirit of the Virtual Crystal Approximation (VCA), however, the fractional nuclear charge approach does not work well when for instance the fractional z is near the value of a noble gas. For instance when mixing Si (Z=14) and C(Z=8) atoms you may get near Ne (Z=10), and the corresponding lattice will be way too diffuse.

If you want to perform a scan it can be useful to use the ModifyAlternativeElement option

Example:

```
System
   ModifyAlternativeElements true

   Atoms
       Si 0.0 0.0 0.0
       H  0.0 0.0 0.0 nuclear_charge=5.6   ! Element H is ignored and will be "rounded
↪" to nearest atom (for the basis set) in this case C
   End
```

In band an alternative element works well with the frozen core approximation, using a smaller or no core has little effect. The VCA relies on defining an average pseudopotential (commonly used in plane wave programs) and is not identical to the alternative element approach (defining an average nuclear charge).

4.2 K-Space

The K-Space sampling (i.e., the k-points used to sample the Brillouin Zone) is an important technical aspect of Band, as it influences heavily the accuracy, the CPU time and the memory usage of the calculation (see section *Recommendations for k-space* (page 69)).

4.2.1 KSpace input block

The K-Space can be controlled via the `KSpace` input block. Two different k-space integration methods are available: the *Regular Grid* (**default**) and the *Symmetric Grid*.

```
KSpace
   Type [Regular | Symmetric]
   Quality [Auto | GammaOnly | Basic | Normal | Good | VeryGood | Excellent]
End
```

`KSpace`

Type
Block

Description
Options for the k-space integration (i.e. the grid used to sample the Brillouin zone)

`Type`

Type
Multiple Choice

Default value
Regular

Options
[Regular, Symmetric]

GUI name
K-space grid type

Description
The type of k-space integration grid used to sample the Brillouin zone (BZ) used.

'Regular': simple regular grid.

'Symmetric': symmetric grid for the irreducible wedge of the first BZ (useful when high-symmetry points in the BZ are needed to capture the correct physics of the system, graphene being a notable example).

`Quality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, GammaOnly, Basic, Normal, Good, VeryGood, Excellent]

GUI name
K-space

Description
Select the quality of the K-space grid used to sample the Brillouin Zone. If 'Auto', the quality defined in the 'NumericalQuality' will be used. If 'GammaOnly', only one point (the gamma point) will be used.

The actual number of K points generated depends on this option and on the size of the unit cell. The larger the real space cell, the fewer K points will be generated.

The CPU-time and accuracy strongly depend on this option.

Regular K-Space grid

By default, Band will look at the size of a lattice vectors and the KSpace quality to determine the number of k-points. The larger the lattice vector in real space, the smaller the reciprocal space vectors are, and as a result fewer k-points are needed. 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 k-points will be used along a lattice vector.

Lattice vector length	Basic	Normal	Good	VeryGood	Excellent
0-5 Bohr	5	9	13	17	21
5-10 Bohr	3	5	9	13	17
10-20 Bohr	1	3	5	9	13
20-50 Bohr	1	1	3	5	9
50- Bohr…	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.

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

```
KSpace
   Regular
      NumberOfPoints integer_list
   End
End
```

KSpace

Type
Block

Description
Options for the k-space integration (i.e. the grid used to sample the Brillouin zone)

Regular

Type
Block

Description
Options for the regular k-space integration grid.

NumberOfPoints

Type
Integer List

Description
Use a regular grid with the specified number of k-points along each reciprocal lattice vector.

For 1D periodic systems you should specify only one number, for 2D systems two numbers, and for 3D systems three numbers.

Symmetric K-Space grid (tetrahedron method)

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

The number of k-points in the symmetric grid depends on the KSpace quality and on the length of the shortest lattice vector.

It is also possible to manually specify the symmetric k-space integration parameter:

```
KSpace
   Symmetric
      KInteg integer
```

(continues on next page)

(continued from previous page)

```
   End
End
```

KSpace

Type
Block

Description
Options for the k-space integration (i.e. the grid used to sample the Brillouin zone)

Symmetric

Type
Block

Description
Options for the symmetric k-space integration grid.

KInteg

Type
Integer

GUI name
Accuracy

Description
Specify the accuracy for the Symmetric method.

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

2: linear tetrahedron method, coarsest spacing

3: quadratic tetrahedron method, coarsest spacing

4,6,… (even): linear tetrahedron method

5,7…. (odd): quadratic method

The tetrahedron method is usually by far inferior.

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.

4.2.2 Recommendations for k-space

Which K-Space quality to use depends very much a) the system you are studying and b) the property you are interested in. We strongly recommend you to test the effect of different K-Space qualities on your system and properties of interest.

As an example, in the following table we list the errors on formation energy and band gap for diamond using regular k-space grids of different qualities (using Excellent kSpace quality as reference).

Table 4.6: Accuracy of formation energy for diamond (primitive unit cell) using different KSpace grids

KSpace quality	Energy error / atom [eV]	CPU time ratio
Gamma-Only	3.3	1
Basic	0.6	2
Normal	0.03	6
Good	0.002	16
VeryGood	0.0001	35
Excellent	reference	64

It is worthwhile noting that the errors due to finite k-space sampling in formation energies are to some extend systematic, and they partially cancel each other out when taking energy differences.

In general, metals (or narrow-gap semiconductor) require higher K-Space sampling than insulators. For insulators and wide-gap semiconductors, Normal K-Space quality often suffices. For Narrow-gap semiconductor and metals, Good K-Space quality is highly recommended. For geometry optimizations under pressure, Good K-Space quality is recommended.

Furthermore for certain properties, such as band gaps, Normal K-Space quality might not be enough to obtain reliable results. For example, in following figure we see how Normal K-Space quality is often not enough for computing band gaps (especially for the narrow-gap semiconductor of the top panel). For band gap prediction, it is recommended to use Good K-Space quality.

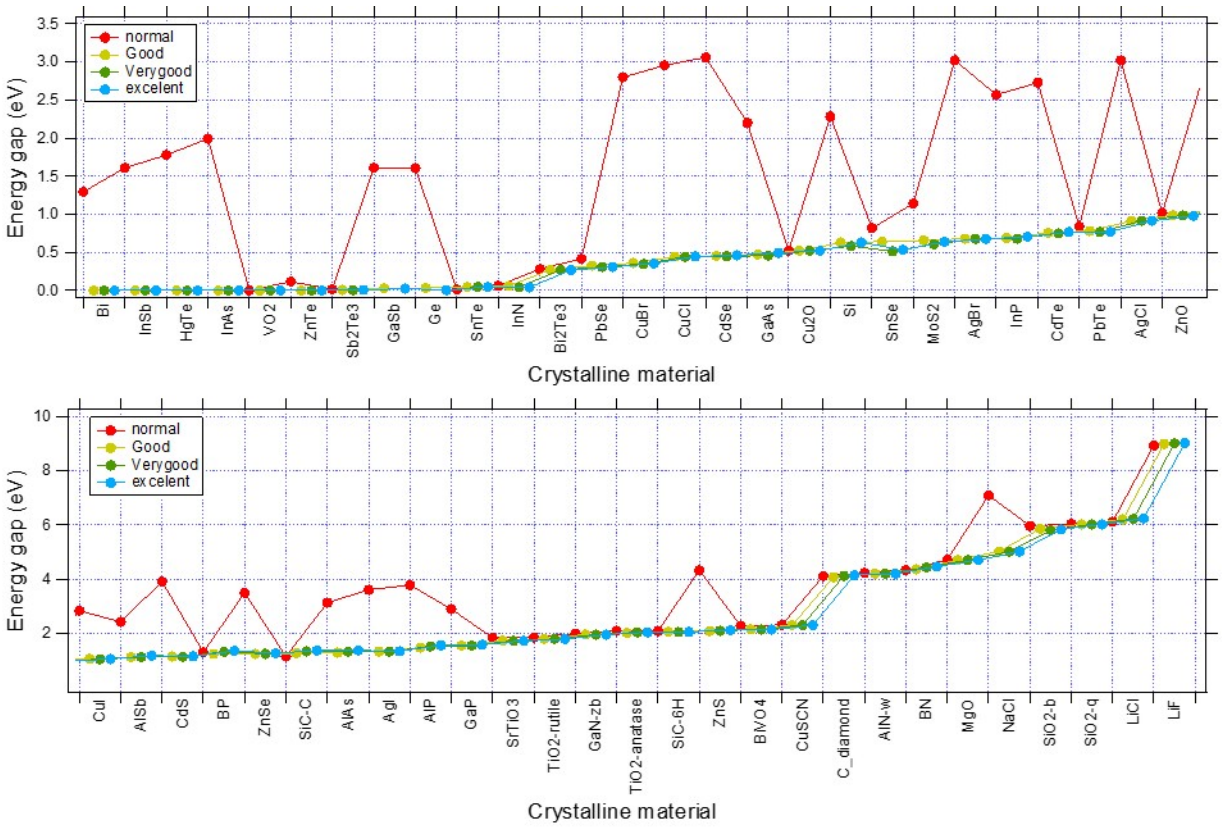


Fig. 4.2: Convergence of band gaps WRT the k-space quality set for various system. (XC:PBE, Basis:TZP)

High symmetry points and the regular grid

Using the symmetric k-grid it is ensured that points with high symmetry are included. However, for the default regular grid this is not the case. An important example is graphene. Its band structure has a conical intersection at the point labeled "K", in all other k-points there is a gap.

Table 4.7: The magical values for which the regular k-grid includes the high symmetry point "K" (with fractional coordinates 1/3,1/3) for a honeycomb lattice.

grid	point "K" included?	K-grid quality for Graphene
5x5	No	Normal
7x7	Yes	
9x9	No	Good
11x11	No	
13x13	Yes	VeryGood
15x15	No	

But in general it makes more sense to use the symmetric k-grid if points of high symmetry are deemed important.

4.3 Numerical Integration

Many of the integrals needed by Band are computed via numerical integration. See also: Wikipedia page on Numerical Integration (https://en.wikipedia.org/wiki/Numerical_integration).

4.3.1 Becke Grid

The numerical integration grid is a refined version of the fuzzy cells integration scheme developed by Becke[1]. The implementation in BAND is described in Ref.[2].

The quality of the Becke integration grid can be changed within the `BeckeGrid` block:

```
BeckeGrid
   Quality [Auto | Basic | Normal | Good | VeryGood | Excellent]
   RadialGridBoost float
   QualityPerRegion
      Quality [Basic | Normal | Good | VeryGood | Excellent]
      Region string
   End
End
```

`BeckeGrid`

Type
Block

Description
Options for the numerical integration grid, which is a refined version of the fuzzy cells integration scheme developed by Becke.

[1] A.D. Becke, *A multicenter numerical integration scheme for polyatomic molecules*, Journal of Chemical Physics 88, 2547 (1988) (https://doi.org/10.1063/1.454033).

[2] 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) (https://doi.org/10.1002/jcc.23323).

`Quality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, Basic, Normal, Good, VeryGood, Excellent]

Description
Quality of the integration grid. For a description of the various qualities and the associated numerical accuracy see reference. If 'Auto', the quality defined in the 'NumericalQuality' will be used.

`RadialGridBoost`

Type
Float

Default value
1.0

Description
The number of radial grid points will be boosted by this factor. Some XC functionals require very accurate radial integration grids, so BAND will automatically boost the radial grid by a factor 3 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.

`QualityPerRegion`

Type
Block

Recurring
True

Description
Sets the grid quality for all atoms in a region. If specified, this overwrites the globally set quality.

`Quality`

Type
Multiple Choice

Options
[Basic, Normal, Good, VeryGood, Excellent]

Description
The region's integration grid quality.

`Region`

Type
String

Description
The identifier of the region for which to set the quality.

Example: Multiresolution (page 230) illustrates how to use the `QualityPerRegion` option.

Notes:

- The space-partition function used in BAND differs from the one described in Ref.[Page 71, 2]. 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-Ubn):

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

- 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.

4.3.2 Radial grid of NAOs

```
RadialDefaults
   NR integer
   NRPerType integer_list
   RMax float
   RMin float
   RMinPerType float_list
End
```

`RadialDefaults`

Type
Block

Description
Options for the logarithmic radial grid of the basis functions used in the subprogram Dirac

`NR`

Type
Integer

Default value
3000

Description
Number of radial points. With very high values (like 30000) the Dirac subprogram may not converge.

`NRPerType`

Type
Integer List

Description
If present overrides NR. The list needs to be as long as there are atom types

`RMax`

Type
Float

Default value
100.0

Unit
Bohr

Description
Upper bound of the logarithmic radial grid

RMin

Type
Float

Default value
1e-06

Unit
Bohr

Description
Lower bound of the logarithmic radial grid

RMinPerType

Type
Float List

Unit
Bohr

Description
If specified overrides RMin. The list needs to be as long as there are atom types (different elements)

4.3.3 Voronoi grid (deprecated)

It is possible to use an alternative numerical integration scheme to the Becke Grid, namely the Voronoi Grid.

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

IntegrationMethod

Type
Multiple Choice

Default value
Becke

Options
[Becke, Voronoi]

Description
Choose the real-space numerical integration method. Note: the Voronoi integration scheme is deprecated.

The options for the Voronoi Grid are specified in the `Integration` block:

```
Integration
   AccInt float
End
```

Integration

Type
Block

Description
Options for the Voronoi numerical integration scheme. Deprecated. Use BeckeGrid instead.

`AccInt`

Type
Float

Default value
3.5

Description
General parameter controlling the accuracy of the Voronoi integration grid. A value of 3 would be basic quality and a value of 7 would be good quality.

4.4 Density Fitting

The Coulomb potential in Band is computed using a method called density fitting. The density fitting scheme in BAND is called **Zlm Fit**, and it is described in reference[1]. The ZlmFit is also used to compute (when needed) the gradient and hessian of the electron density.

4.4.1 Zlm Fit

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

```
ZlmFit
   Quality [Auto | Basic | Normal | Good | VeryGood | Excellent]
   QualityPerRegion
      Quality [Basic | Normal | Good | VeryGood | Excellent]
      Region string
   End
End
```

`ZlmFit`

Type
Block

Description
Options for the density fitting scheme 'ZlmFit'.

`Quality`

Type
Multiple Choice

Default value
Auto

[1] 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) (https://doi.org/10.1021/ct500172n).

Options
[Auto, Basic, Normal, Good, VeryGood, Excellent]

GUI name
Spline Zlm fit

Description
Quality of the density-fitting approximation. For a description of the various qualities and the associated numerical accuracy see reference. If 'Auto', the quality defined in the 'NumericalQuality' will be used.

`QualityPerRegion`

Type
Block

Recurring
True

Description
Sets the ZlmFit quality for all atoms in a region. If specified, this overwrites the globally set quality.

`Quality`

Type
Multiple Choice

Options
[Basic, Normal, Good, VeryGood, Excellent]

Description
The region's quality of the ZlmFit.

`Region`

Type
String

Description
The identifier of the region for which to set the quality.

Example: Multiresolution (page 230) illustrates how to use the `QualityPerRegion` option.

Expert options

```
ZlmFit
   LMargin integer
   AllowBoost Yes/No
   DensityThreshold float
   PartitionFunThreshold float
   FGaussianW float
   FGridSpacing float
   FKSpaceCutOff float
   FirstTopoCell integer
   LastTopoCell integer
   OrderTopoTrick integer
   NumStarsPartitionFun integer
End
```

`ZlmFit`

Type
Block

Description
Options for the density fitting scheme 'ZlmFit'.

`LMargin`

Type
Integer

Description
User-defined l-margin, i.e., l_max for fitting is max(lMargin + l_max_basis_function, 2*l_max_basis_function). Depends on Quality and normally is 4

`AllowBoost`

Type
Bool

Default value
Yes

Description
Allow automatic atom-dependent tuning of maximum l of spherical harmonics expansion. Whether or not this boost is needed for a given atom is based on an heuristic estimate of how complex the density around that atom is.

`DensityThreshold`

Type
Float

Description
Threshold below which the electron density is considered to be negligible. Depends on Quality and is normally 1.0e-7

`PartitionFunThreshold`

Type
Float

Default value
0.0

Description
Threshold for the partition functions: if an integration point has a partition function weight smaller than this threshold, it will be discarded.

`FGaussianW`

Type
Float

Default value
1.0

Description
Only for 3D periodic systems. Width of the Gaussian functions replacing the S and P Zlms for Fourier transform.

`FGridSpacing`

Type
Float

Description
Only for 3D periodic systems. Spacing for the Fourier grid. By default, this depends on the quality.

FKSpaceCutOff

Type
Float

Description
Only for 3D periodic systems. Cut-off of the grid in k-space for the Fourier transform.

FirstTopoCell

Type
Integer

Default value
5

Description
First cell for the topological extrapolation of the long range part of the Coulomb Potential.

LastTopoCell

Type
Integer

Default value
10

Description
Last cell for the topological extrapolation of the long range part of the Coulomb Potential.

OrderTopoTrick

Type
Integer

Default value
3

Description
Order of the topological extrapolation of the long range part of the Coulomb Potential.

NumStarsPartitionFun

Type
Integer

Default value
5

Description
Number of cell stars to consider when computing the partition function.

4.4.2 STO Fit (Deprecated)

In previous version of BAND (pre2014) this was the default option, which is now replaced by Zlm Fit. It is still used in the context of `NMR` and `OldResponse` calculations.

4.5 Hartree–Fock RI

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.*[1]. For more information on hybrid functionals in BAND, see the *XC section* (page 15).

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

```
RIHartreeFock
   Quality [Auto | VeryBasic | Basic | Normal | Good | VeryGood | Excellent]
   FitSetQuality [Auto | VeryBasic | Basic | Normal | Good | VeryGood | Excellent |␣
↪FromBasisProducts]
   DependencyThreshold float
   QualityPerRegion
      Quality [VeryBasic | Basic | Normal | Good | VeryGood | Excellent]
      Region string
   End
End
```

`RIHartreeFock`

Type
Block

Description

`Quality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
Numerical accuracy of the RI procedure. If 'Auto', the quality specified in the 'NumericalQuality' will be used.

`FitSetQuality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, VeryBasic, Basic, Normal, Good, VeryGood, Excellent, FromBasisProducts]

[1] 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 (https://doi.org/10.1088/1367-2630/14/5/053020).

Description
The quality of auxiliary fit set employed in the RI scheme.

If 'Auto', the value of the RIHartreeFock Quality option will be used.

Normal quality is generally sufficient for basis sets up to and including TZ2P.

For larger basis sets (or for benchmarking purposes) a VeryGood fit set is recommended.

Note that the FitSetQuality heavily influences the computational cost of the calculation.

`DependencyThreshold`

Type
Float

Default value
0.001

Description
To improve numerical stability, almost linearly-dependent combination of basis functions are removed from the Hartree-Fock exchange matrix.

If you obtain unphysically large bond energy in an Hybrid calculation, or an unphysically low correlation energy in an RPA, MP2, or double hybrid calculation, you might try setting the DependencyThreshold to a larger value (e.g. 3.0E-3)

Note, that in GW calculations and GW-BSE calculations the default for this key is 5.0e-3.

`QualityPerRegion`

Type
Block

Recurring
True

Description
Sets the fit-set quality for all atoms in a region. If specified, this overwrites the globally set quality.

`Quality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
This region's quality of the auxiliary fit set employed in the RI scheme.

`Region`

Type
String

Description
The identifier of the region for which to set the quality.

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

```
SoftConfinement
  Quality Basic
End
```

See the *Confinement of basis functions* (page 62) section for more info.

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

Note:

- In AMS2019.3 the fit set for `FitSetQuality Good` has been improved.

4.6 Self Consistent Field (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.

$$\text{err} = \sqrt{\int dx\ (\rho_{\text{out}}(x) - \rho_{\text{in}}(x))^2}$$

When the SCF error is below a certain criterion, controlled by subkey `Criterion` of block key `Convergence`, convergence is reached. The default criterion depends on the `NumericalQuality` key.

Table 4.8: The default `Convergence%Criterion` depends on the `NumericalQuality` and on the number of atoms in the unit cell.

NumericalQuality	Convergence%Criterion
Basic	1e-5 $\sqrt{N_{\text{atoms}}}$
Normal	1e-6 $\sqrt{N_{\text{atoms}}}$
Good	1e-7 $\sqrt{N_{\text{atoms}}}$
VeryGood	1e-8 $\sqrt{N_{\text{atoms}}}$

The default method is the so-called MultiStepper. The MultiStepper is flexible, but somewhat harder to control by the user.

See also:

Troubleshooting: *SCF does not converge* (page 199)

4.6.1 SCF block

```
SCF
   Eigenstates Yes/No
   Iterations integer
   Method [DIIS | MultiSecant | MultiStepper]
   Mixing float
   MultiStepperPresetPath string
   PMatrix Yes/No
   PrintAllOccupiedBands Yes/No
   PrintAllVirtualBands Yes/No
   PrintAlwaysBandRanges Yes/No
   Rate float
   VSplit float
End
```

SCF

Type
Block

Description
Controls technical SCF parameters.

`Eigenstates`

Type
Bool

Description
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).

`Iterations`

Type
Integer

Default value
300

GUI name
Maximum number of cycles

Description
The maximum number of SCF iterations to be performed.

`Method`

Type
Multiple Choice

Default value
MultiStepper

Options
[DIIS, MultiSecant, MultiStepper]

Description
Choose the general scheme used to converge the density in the SCF. In case of scf problems one can try the MultiSecant alternative at no extra cost per SCF cycle. For more details see the DIIS and MultiSecantConfig block.

`Mixing`

Type
Float

Default value
0.075

Description
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.

`MultiStepperPresetPath`

Type
String

Default value
DFTB/default2023.inc

Description
Name of file containing a SCFMultiStepper key block. This will be used if no Explicit SCF-MultiStepper block is in the input, and Method=MultiStepper.

If the path is not absolute, it is relative to $AMSHOME/data/presets/multi_stepper'

`PMatrix`

Type
Bool

Description
If present, evaluate the charge density from the P-matrix. See also the key Eigenstates.

`PrintAllOccupiedBands`

Type
Bool

Default value
Yes

Description
When printing the ranges of the bands, include all occupied ones.

`PrintAllVirtualBands`

Type
Bool

Default value
No

Description
When printing the ranges of the bands, include all virtual ones.

`PrintAlwaysBandRanges`

Type
Bool

Default value
No

Description
Normally the ranges of the bands are only printed at the last SCF cycle

`Rate`

Type
Float

Default value
0.99

Description
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 key Degenerate of block Convergence) or, if everything seems to fail, it will stop

VSplit

Type
Float

Default value
0.05

Description
To disturb degeneracy of alpha and beta spin MOs the value of this key is added to the beta spin potential at the startup.

4.6.2 Convergence

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 float
   CriterionFactor float
   Degenerate string
   ElectronicTemperature float
   InitialDensity [rho | psi | frompot]
   LessDegenerate Yes/No
   ModestCriterion float
   NoDegenerate Yes/No
   NumBoltz integer
   SpinFlip integer_list
   SpinFlipEnabled Yes/No
   SpinFlipRegion string
   StartWithMaxSpin Yes/No
   StartWithMaxSpinForSO Yes/No
End
```

Convergence

Type
Block

Description
Options and parameters related to the convergence behavior of the SCF procedure.

Criterion

Type
Float

Description
Criterion for termination of the SCF procedure. The default depends on the NumericalQuality and on the number of atoms in the system.

Can be used for EngineAutomations

CriterionFactor

Type
Float

Default value
1.0

Description
Multiply Criterion (which depends on system and quality) with this factor.

Can be used for EngineAutomations

`Degenerate`

Type
String

Default value
default

Description
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.

`ElectronicTemperature`

Type
Float

Default value
0.0

Unit
Hartree

Description
(KT) Specify this key for a gradient independent electronic temperature

`InitialDensity`

Type
Multiple Choice

Default value
rho

Options
[rho, psi, frompot]

Description
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`

Type
Bool

Default value
No

Description
If smoothing of occupations over nearly degenerate orbitals is applied (see Degenerate 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.

ModestCriterion

Type
Float

Default value
-1.0

Description
If this is specified band will consider the SCF converged if the error is below this criterion (after using the maximum number of iterations).

NoDegenerate

Type
Bool

Default value
No

Description
This key prevents any internal automatic setting of the key DEGENERATE.

NumBoltz

Type
Integer

Default value
10

Description
The electronic temperature is done with a Riemann Stieltjes numerical integration, between zero and one occupation. This defines the number of points to be used.

SpinFlip

Type
Integer List

GUI name
Flip spin for atoms

Description
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.

SpinFlipEnabled

Type
Bool

Default value
Yes

Description
If set to False, the keys SpinFlip and SpinFlipRegion are ignored. Only useful/convenient when trying to compare in a script the effect of spin flip.

SpinFlipRegion

Type
String

Recurring
True

GUI name
Flip spin for region

Description
Specify here the region 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

Type
Bool

Default value
Yes

Description
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 key.

StartWithMaxSpinForSO

Type
Bool

Default value
No

Description
Same as the StartWithMaxSpin option. In case of spin-orbit band always used to split the potential. Now will use maxspin in case of SpinFlip. With this option it will always do that.

4.6.3 DIIS

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

```
DIIS
   Adaptable Yes/No
   CHuge float
   CLarge float
   Condition float
   DiMix float
   DiMixMax float
   DiMixMin float
   NCycleDamp integer
   NVctrx integer
   Variant [DIIS | LISTi | LISTb | LISTd]
End
```

DIIS

Type
Block

Description
Parameters for the DIIS procedure to obtain the SCF solution

Adaptable

Type
Bool

Default value
Yes

Description
Change automatically the value of dimix during the SCF.

CHuge

Type
Float

Default value
20.0

GUI name
No DIIS (but damping) when coefs >

Description
When the largest coefficient in the DIIS expansion exceeds this value, damping is applied

CLarge

Type
Float

Default value
20.0

GUI name
Reduce DIIS space when coefs >

Description
When the largest DIIS coefficient exceeds this value, the oldest DIIS vector is removed and the procedure re-applied

Condition

Type
Float

Default value
1000000.0

Description
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.

DiMix

Type
Float

Default value
0.2

GUI name
Bias DIIS towards latest vector with

Description
Mixing parameter for the DIIS procedure

`DiMixMax`

Type
Float

Default value
-1.0

Description
For adaptive diis: A negative value means automatic, see DiMixatnvctrx. If positive it is an absolute upper bound for (adaptive) dimix

`DiMixMin`

Type
Float

Default value
0.01

Description
An absolute lower bound for adaptive dimix.

`NCycleDamp`

Type
Integer

Default value
1

GUI name
Do not start DIIS before cycle

Description
Number of initial iterations where damping is applied, before any DIIS is considered

`NVctrx`

Type
Integer

Default value
20

GUI name
Size of DIIS space

Description
Maximum number of DIIS expansion vectors

`Variant`

Type
Multiple Choice

Default value
DIIS

Options
[DIIS, LISTi, LISTb, LISTd]

Description
Which variant to use. In case of problematic SCF convergence, first try MultiSecant, and if that does not work the LISTi is the advised method. Note: LIST is computationally more expensive per SCF iteration than DIIS.

4.6.4 Multisecant

For more detais on the multisecant method see ref[1].

```
MultiSecantConfig
   CMax float
   InitialSigmaN float
   MaxSigmaN float
   MaxVectors integer
   MinSigmaN float
End
```

`MultiSecantConfig`

Type
Block

Description
Parameters for the Multi-secant SCF convergence method.

`CMax`

Type
Float

Default value
20.0

GUI name
Max coeff

Description
Maximum coefficient allowed in expansion

`InitialSigmaN`

Type
Float

Default value
0.1

GUI name
Initial

Description
This is a lot like a mix factor: bigger means bolder

[1] L. D. Marks and D. R. Luke, *Robust mixing for ab initio quantum mechanical calculations*, Phys. Rev. B 78, 075114 (2008) (https://doi.org/10.1103/PhysRevB.78.075114)

MaxSigmaN

Type
Float

Default value
0.3

GUI name
Max

Description
Upper bound for the SigmaN parameter

MaxVectors

Type
Integer

Default value
20

GUI name
Number of cycles to use

Description
Maximum number of previous cycles to be used

MinSigmaN

Type
Float

Default value
0.01

GUI name
Min

Description
Lower bound for the SigmaN parameter

4.6.5 MultiStepper

The MultiStepper introduces the concept of alternating between different steppers (methods). Methods are not switched at every SCF cycle, but rather after a sequence of them, called a stint. At the end of a stint it is considered whether it makes sense to try another stepper.

The key component is the Stepper. This wraps the type of the Stepper, say DIIS or SimpleMixing. Another important component is the MixAdapter. A step is controlled by a mix factor σ, also often called greed. The next guess density is a linear combination of previous input and output densities

$\rho^{\text{guess}} = \sum_i c_{i-1}^N (\rho_i^{\text{in}} + \sigma(\rho_i^{\text{out}} - \rho_i^{\text{in}}))$

The larger the mix factor the more aggressive the algorithm. Choosing it too small may simply stall the progress and choosing it too large can cause the error to grow. That is why using a MixAdapter is useful. It tries to predict a reasonable mix value, based on the progress of the error and also based on the number of previous iterations N that can be used without running into numerical problems.

A whole SCFMultiStepper block can be loaded from a file as a preset, and many reside in `$AMSHOME/data/presets/multi_stepper`. Normal users are not recommended to try to improve the standard preset. Which

preset to loaded is controlled by the `SCF%MultiStepperPresetPath` key, and this may be an absolute path to your own preset.

The the log file (ams.log) shows the active stepper and mix factor.

```
<Nov22-2022> <15:24:28>  cyc=  0 err=0.00E+00 cpu=  75s ela=  76s
<Nov22-2022> <15:25:26>  cyc=  1 err=4.26E+00 meth=1 nvec= 1 mix=0.0750 cpu=  57s␣
↪ela=  58s fit=7.06E-02
<Nov22-2022> <15:26:26>  cyc=  2 err=8.33E+00 meth=1 nvec= 2 mix=0.1455 cpu=  59s␣
↪ela=  60s fit=6.49E-02
<Nov22-2022> <15:27:23>  cyc=  3 err=7.85E+00 meth=1 nvec= 3 mix=0.1499 cpu=  56s␣
↪ela=  57s fit=6.42E-02
<Nov22-2022> <15:28:24>  cyc=  4 err=7.09E+00 meth=1 nvec= 4 mix=0.1544 cpu=  60s␣
↪ela=  61s fit=6.37E-02
<Nov22-2022> <15:29:21>  cyc=  5 err=9.49E+00 meth=2 nvec= 1 mix=0.0060 cpu=  57s␣
↪ela=  57s fit=7.91E-02
<Nov22-2022> <15:30:20>  cyc=  6 err=2.63E+00 meth=2 nvec= 2 mix=0.0062 cpu=  59s␣
↪ela=  59s fit=7.88E-02
<Nov22-2022> <15:31:18>  cyc=  7 err=3.82E+00 meth=2 nvec= 3 mix=0.0060 cpu=  57s␣
↪ela=  58s fit=7.84E-02
<Nov22-2022> <15:32:16>  cyc=  8 err=3.53E+00 meth=2 nvec= 4 mix=0.0062 cpu=  58s␣
↪ela=  58s fit=7.81E-02
```

From cycle 5 (cyc=5) on the second stepper is tried (meth=2), in this case because the error has grown too much since the start. Furthermore it restarts from the first density, not shown in the log file, using only one older density (nvec=1). Note that the second stepper starts with using a much more conservative mix factor (mix=0.006).

```
SCF
   SCFMultiStepper
      AlwaysChangeStepper Yes/No
      ErrorGrowthAbortFactor float
      FractionalStepFactor float
      MinStintCyclesForAbort integer
      Stepper header
         AbortSlope float
         DIISStepper
            EDIISAlpha float
            MaxCoefficient float
            MaxVectors integer
            MinVectors integer
            Mix float
         End
         ErrorGrowthAbortFactor float
         ExpectedSlope float
         FractionalStepFactor float
         MaxInitialError float
         MaxIterationNumber integer
         MaxStintNumber integer
         MinInitialError float
         MinIterationNumber integer
         MinStintCyclesForAbort integer
         MinStintNumber integer
         MixAdapter
            ErrorGrowthPanicFactor float
            GrowthFactor float
            MaxMix float
            MinMix float
            NTrialMixFactors integer
```

(continues on next page)

(continued from previous page)

```
            TrialMode [CurrentMixCentered | FullRange]
            Type [Error | Energy | UnpredictedStep | Trial]
         End
         MixStepper
            Mix float
         End
         MultiSecantStepper
            MaxCoefficient float
            MaxVectors integer
            Mix float
            Variant [MSB1 | MSB2 | MSR1 | MSR1s]
         End
         StintLength integer
      End
      StintLength integer
      UsePreviousStintForErrorGrowthAbort Yes/No
   End
End
```

SCF

SCFMultiStepper

Type
Block

Description
To solve the self-consistent problem multiple steppers can be tried during stints using the ones that give the best progress.

AlwaysChangeStepper

Type
Bool

Default value
No

Description
When the progress is fine there is no reason to change the stepper. In practice this is always set to true, because also the Stepper%ExpectedSlope can be used to achieve similar behavior.

ErrorGrowthAbortFactor

Type
Float

Default value
1000.0

Description
Abort stint when the error grows too much, compared to the error at the start of the stint.

FractionalStepFactor

Type
Float

Default value
-1.0

Description
Multiply the step by this factor. If smaller than zero this is not used.

`MinStintCyclesForAbort`

Type
Integer

Default value
0

Description
Look at ErrorGrowthAbortFactor only when a number of steps has been completed since the start of the stint. A value of 0 means always.

`Stepper`

Type
Block

Recurring
True

Description
??

`AbortSlope`

Type
Float

Default value
100.0

Description
If the slope (at the end of a stint) is larger than this: abort the stepper

`DIISStepper`

Type
Block

Description
DIIS stepper

`EDIISAlpha`

Type
Float

Default value
0.01

Description
The extra energy vector is weighed by this factor. .

`MaxCoefficient`

Type
Float

Default value
20.0

Description
The largest allowed value of the expansion coefficients. If exceed the number of vectors is reduces until the criterion is met.

MaxVectors

Type
Integer

Default value
10

Description
Maximum number of previous densities to be used (size of the history).

MinVectors

Type
Integer

Default value
-1

Description
Try to prevent to make nVectors shrink below this value, by allowing for significantly larger coefficients.

Mix

Type
Float

Default value
0.2

Description
Also known as greed. It determines the amount of output density to be used. May be changed by the MixAdapter.

ErrorGrowthAbortFactor

Type
Float

Default value
-1.0

Description
Abort stint when the error grows too much, compared to the error at the start of the stint. Overrides global ErrorGrowthAbortFactor when set to a value > 0

ExpectedSlope

Type
Float

Default value
-100.0

Description
If the slope of the total SCF is better than this keep on going.

FractionalStepFactor

Type
Float

Default value
-1.0

Description
Multiply the step by this factor. If smaller than zero this is not used.

`MaxInitialError`

Type
Float

Description
Only use the stepper when error is smaller than this.

`MaxIterationNumber`

Type
Integer

Default value
-1

Description
Stepper will only be active for iterations smaller than this number. (Negative value means: Ignore this option)

`MaxStintNumber`

Type
Integer

Default value
-1

Description
Stepper will only be active for stints smaller than this number. (Negative value means: Ignore this option)

`MinInitialError`

Type
Float

Description
Only use the stepper when error is larger than this.

`MinIterationNumber`

Type
Integer

Default value
-1

Description
Stepper will only be active for iterations larger than this number.

`MinStintCyclesForAbort`

Type
Integer

Default value
0

Description
Look at ErrorGrowthAbortFactor only when a number of steps has been completed since the start of the stint. A value of 0 means always. Overrides global value.

`MinStintNumber`

Type
Integer

Default value
-1

Description
Stepper will only be active for stints larger than this number.

`MixAdapter`

Type
Block

Description
Generic mix adapter

`ErrorGrowthPanicFactor`

Type
Float

Default value
10.0

Description
When the error increases more than this factor, this mix is reduced a lot.

`GrowthFactor`

Type
Float

Default value
1.1

Description
When the mix is considered too low it is multiplied by this factor. Otherwise it is divided by it.

`MaxMix`

Type
Float

Default value
0.3

Description
Do not grow the mix above this value.

`MinMix`

Type
Float

Default value
0.1

Description
Do not shrink the mix below this value.

NTrialMixFactors

Type
Integer

Default value
3

Description
Only used with Type=Trial. Must be an odd number.

TrialMode

Type
Multiple Choice

Default value
CurrentMixCentered

Options
[CurrentMixCentered, FullRange]

Description
How are the NTrialMixFactors chosen?

Type

Type
Multiple Choice

Default value
Error

Options
[Error, Energy, UnpredictedStep, Trial]

Description
Adapt the mix factor based on the observed progress (slope).

MixStepper

Type
Block

Description
Simple mixing stepper, only using the previous (in/out) density.

Mix

Type
Float

Default value
0.1

Description
???.

MultiSecantStepper

Type
Block

Description
Multi secant stepper.

MaxCoefficient

Type
Float

Default value
20.0

Description
???.

MaxVectors

Type
Integer

Default value
10

Description
???.

Mix

Type
Float

Default value
0.2

Description
???.

Variant

Type
Multiple Choice

Default value
MSB2

Options
[MSB1, MSB2, MSR1, MSR1s]

Description
There are several version of the Multi secant method.

StintLength

Type
Integer

Description
Override global StintLength.

StintLength

Type
Integer

Default value
10

Description
A stepper is active during a number of SCF cycles, called a stint.

`UsePreviousStintForErrorGrowthAbort`

Type
Bool

Default value
No

Description
The error is normally checked against the first error of the stint. With this option that will be the one from the previous stint, if performed with the same stepper.

4.6.6 DIRIS

In the DIRIS block, which has the same options as the `DIIS` block, you can specify the DIIS options to be used in the Dirac subprogram, for numerical single atom calculations, which constructs the radial tables for the NAOs.

4.7 MBPT scheme

Note: This page describes technical aspects of the MBPT (Many-Body Perturbation Theory) module which is used in double-hybrid and MP2, RPA, GW and GW-BSE calculations. In order to use double-hybrids, MP2 or RPA in your calculation you should request it in the *XC input block* (page 15). In order to perform a GW calculation, you should request it in the *GW input block* (page 137).

BAND implements RPA, GW, and SOS-MP2 (spin-opposite-scaled) using a newly designed algorithm which in all cases scales quadratically with system size[1][3]. Full MP2 is at the moment only implemented using the canonical RI-algorithm which scales to the fifth power with system size. Thus, we strongly discourage using full MP2 or double-hybrids employing full MP2 for system larger than 1000-1500 basis functions. At the moment BAND features a large number of double-hybrids using SOS-MP2 only (For a list of implemented functionals see *XC input block* (page 15)) which are significantly faster than conventional double-hybrids while offering the same level of accuracy[2].

GW, MP2, RPA and double-hybrid functionals can be used icw scalar relativistic effects within the `ZORA`, `X2C`, or `RA-X2C` formalism. In BAND2022 in case of ZORA by default the so called scaled ZORA orbital energies are used in the MBPT expressions.

The Formalism used in the double-hybrid calculation can be changed using the Formalism key. By default, BAND selects the most appropriate algorithm for your system and functional.

The calculation of the independent-particle polarizability or Kohn-Sham density response function in imaginary time is the key step in SOS-MP2, RPA and GW. The equations are solved in the atomic orbital basis exploiting sparsity via advanced density fitting techniques (so-called pair-atomic resolution of the identity or pair-atomic density fitting)[Page 100, 1].

[1] A. Förster, M. Franchini, E. van Lenthe, L. Visscher, *A Quadratic Pair Atomic Resolution of the Identity Based SOS-AO-MP2 Algorithm Using Slater Type Orbitals*, Journal of Chemical Theory and Computation 16 875-891 (2020) (https://doi.org/10.1021/acs.jctc.9b00854)

[3] A. Förster, L. Visscher, *Low-order scaling |G0W0| by pair atomic density fitting*, Journal of Chemical Theory and Computation 16 (12), 7381 (2020) (https://doi.org/10.1021/acs.jctc.0c00693)

[2] A. Förster, L. Visscher, *Double hybrid DFT calculations with Slater type orbitals*, Journal of Computational Chemistry 41 1660-1684 (2020) (https://doi.org/10.1002/jcc.26209)

In case of a SOS-MP2 or RPA calculation, the polarizability is than contracted with the Coulomb potential. For SOS-MP2, the correlation energy is then immediately evaluated in imaginary time while in a RPA calculation the product of Coulomb potential and polarizability is Fourier transformed to the imaginary frequency axis where the correlation energy is evaluated using a matrix logarithm. In a GW calculation, the polarizability is Fourier transformed to the imaginary frequency axis as well where the so-called screened interaction is calculated. The QP states are then evaluated along the real-frequency axis using analytical continuation techniques.

4.7.1 Recommended numerical settings

For all calculations using the MBPT scheme (which includes GW, GW-BSE, RPA, MP2, and double hybrids), we recommend to consider the following points:

Dependency

In BAND a similar method is implemented as in the RIHARTREEFOCK scheme to improve stability of the results with the subkey DEPENDENCY of the key MBPT:

```
MBPT
   Dependency Yes/No
End
```

MBPT

Dependency

Type
Bool

Default value
Yes

Description
If true, to improve numerical stability, almost linearly-dependent combination of basis functions are removed from the Green's function that are used in the MBPT equations. Disabling this key is strongly discouraged. Its value can however be changed. The key to adjust this value is RiHartreeFock%DependencyThreshold

In addition one may remove linear dependencies in the basis set, by using the `Dependency` key. The following is recommended

```
Dependency
   AllowBasisDependency True
   Basis 1e-4
End
```

in case a large auxiliary fit set is used

Starting from AMS2022 BAND will use a `Dependency bas=5e-3` and `RIHartreeFock DependencyThreshold=5e-3` in case of (any variant of) GW and GW-BSE. One can override these values in the input.

Numerical Quality

Augmented basis sets: Numerical quality should be `VeryGood`
Non-augmented basis sets, TZ: Numerical quality should range from `Normal` to `VeryGood`
Non-augmented basis sets, QZ: Numerical quality should range from `Good` to `VeryGood`

There is also the option to choose a different numerical quality for the MBPT and the preceding DFT calculation:

```
MBPT
   NumericalQuality Good
END
```

sets the numerical quality to Good only for the MBPT calculation.

The sizes of the imaginary time and imaginary frequency grids can be controlled with the `nTime` and `nFrequency` keys. For example:

```
MBPT
   nTime 32
   nFrequency 32
END
```

In case of SOS-MP2, `nFrequency` is ignored. The numerical quality automatically sets the number of grid points for imaginary time and frequency integration in case of a GW or RPA calculation:

Numerical quality	Number of points
VERYBASIC	8
BASIC	12
NORMAL	16
GOOD	20
VERYGOOD	24
EXCELLENT	32

For a MP2 or double hybrid calculation, see default is always 9 points, independently of the numerical quality. Note, that the requirements for this parameter are in general lower than for a GW or RPA calculation. Starting from AMS2022, the number of points is automatically adjusted in SOS-MP2 as well.

Note the following: The number of points actually used in a calculation can differ. At runtime, the MBPT algorithm decides what is the maximum number of integration points which is reasonable to use. So the actual number of points which has been used will be equal or smaller. In case the number of points is set by hand, some info is printed in the output. In case of a GW calculation, the numbers of points can also be found in band.rkf file in the `GW` section under the names `nTime` and `nFrequency`.

Changing the defaults can be necessary in case your system contains 4th row elements or heavier and/or your basis set is very large and/or your system has a very small HOMO-LUMO gap. For a GW calculation, 24 points should be sufficient for 5th row elements. 32 points might be required for 6th row elements. The maximum number of points which can be used is 42. For a MP2 calculation, 16 points will usually be sufficient if your systems contains 4th row elements, and 20 points will usually suffice in case of 5th row elements. Note, that also a very small HOMO-LUMO.

```
MBPT
   Formalism [Auto | RI | LT | All]
   FrequencyGridType [LeastSquare | GaussLegendre]
   nTime integer
   nFrequency integer
   nLambda integer
   FitSetQuality [Auto | VeryBasic | Basic | Normal | Good | VeryGood]
   IntegrationQuality [VeryBasic | Basic | Normal | Good | VeryGood]
   ThresholdQuality [VeryBasic | Basic | Normal | Good | VeryGood | Excellent]
   Dependency Yes/No
   UseScaledZORA Yes/No
```

(continues on next page)

(continued from previous page)

```
   SigmaFunctionalParametrization [W1 | W2 | S1 | S2 | S1re]
End
```

MBPT

Type
Block

Description
Technical aspects of the MP2 algorithm.

Formalism

Type
Multiple Choice

Default value
Auto

Options
[Auto, RI, LT, All]

Description
Specifies the formalism for the calculation of the MP2 correlation energy.

'LT' means Laplace Transformed MP2 (also referred to as AO-PARI-MP2),

'RI' means that a conventional RI-MP2 is carried out.

If 'Auto', LT will be used in case of DOD double hybrids and SOS MP2, and RI will be used in all other cases.

'All' means that both RI and LT formalisms are used in the calculation.

For a RPA or GW calculation, the formalism is always LT, irrespective of the formalism specified with this key.

FrequencyGridType

Type
Multiple Choice

Default value
LeastSquare

Options
[LeastSquare, GaussLegendre]

Description
Use Gauss-legendre grid for imaginary frequency integration in RPA and GW calculations instead of the usually used Least-Square optimized ones. Has the advantage that it can be systematically converged and an arbitrary number of grid points can be used. Typically more grid points will be needed to get the same level of accuracy. However, the convergence of the results with the size of the grid can be more systematic. These grids can only be used when Formalism is set to RI.

nTime

Type
Integer

GUI name
Number of time points

Description
Number of imaginary time points (only relevant in case the Laplace Transformed (LT) formalism is used).

In the many-body-perturbation theory module in ADF, the polarizability (or Kohn-Sham density response function) is evaluated in imaginary time to exploit sparsity in the AO basis. For MP2, this is often referred to as a Laplace transform. For MP2, 9 points are the default. This is a safe choice, guaranteeing accuracies higher than 1 Kj/mol for most systems (For many simple organic systems, 6 points are sufficient for good accuracy).

Only for systems with a very small HOMO-LUMO gap or low-lying core states (heavy elements starting from the 4th row of the periodic table) more points might be necessary.

In principle, the same considerations apply for RPA and GW as well, however, the accuracy requirements are somewhat higher and 12 point are the default for RPA. In a GW calculation, the number of points is adjusted according to the numerical quality. Using less than 9 points is strongly discouraged except for the simplest molecules.

In ADF2019, it can happen that the algorithm determining the imaginary time grid does not converge. In this case, the usual reason is that the number of points is too small and more points need to be specified. Starting from AMS2020, this does not happen any more. In case the imaginary time grid does not converge, the number of points is automatically adjusted until it does.

The computation time of AO-PARI-MP2, RPA, and GW scales linearly with the number of imaginary time points.

`nFrequency`

Type
Integer

Default value
12

GUI name
N freq points

Description
Number of imaginary frequency points. This key is only relevant for RPA and GW and will be ignored if used in an AO-PARI-MP2 calculation. 12 Points is the default for a RPA calculation. It is technically possible to use a different number of imaginary frequency points than for imaginary time. The maximum number of points which can be requested for imaginary frequency integration is 42. Important note: The computation time and memory requirements roughly scale linearly with the number of imaginary frequency points. However, memory can be an issue for RPA and GW when the number of imaginary frequency points is high. In case a job crashes, it is advised to increase the number of nodes since the necessary memory distributes over all nodes.

`nLambda`

Type
Integer

Default value
1

GUI name
Number of lambda points

Description
Size of coupling constant integration grid for SOSEX variants in RPA. Default is 4 points

`FitSetQuality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, VeryBasic, Basic, Normal, Good, VeryGood]

Description
Specifies the fit set to be used in the MBPT calculation.

'Normal' quality is generally sufficient for basis sets up to and including TZ2P.

For larger basis sets (or for benchmarking purposes) a 'VeryGood' fit set is recommended. Note that the FitSetQuality heavily influences the computational cost of the calculation.

If not specified or 'Auto', the RIHartreeFock%FitSetQuality is used.

`IntegrationQuality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood]

Description
Specifies the integration quality to be used in the MBPT calculation. If not specified, the RIHartreeFock%IntegrationQuality is used.

`ThresholdQuality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
Controls the distances between atomic centers for which the product of two basis functions is not fitted any more. Especially for spatially extended, large systems, 'VERYBASIC' and 'BASIC' can lead to large computational savings, but the fit is also more approximate. If not specified, the RIHartreeFock%ThresholdQuality is used.

`Dependency`

Type
Bool

Default value
Yes

Description
If true, to improve numerical stability, almost linearly-dependent combination of basis functions are removed from the Green's function that are used in the MBPT equations. Disabling this key is strongly discouraged. Its value can however be changed. The key to adjust this value is RiHartreeFock%DependencyThreshold

UseScaledZORA

Type
Bool

Default value
Yes

Description
If true, use the scaled ZORA orbital energies instead of the ZORA orbital energies in the MBPT equations.

SigmaFunctionalParametrization

Type
Multiple Choice

Default value
S1re

Options
[W1, W2, S1, S2, S1re]

Description
Only relevant if a sigma-functional calculation is performed. Possible choices for the parametrization of the sigma-functional. Not all options are supported for all functionals.

4.8 More Technical Settings

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

4.8.1 Linear Scaling

```
Tails
   Bas float
End
```

Tails

Type
Block

Description
Ignore function tails.

Bas

Type
Float

Default value
1e-06

GUI name
Basis functions

Description
Cut off the basis functions when smaller than the specified threshold.

4.8.2 Dependency

```
Dependency
   AllowBasisDependency Yes/No
   Basis float
   Core float
   CoreValence float
   Fit float
End
```

Dependency

Type
Block

Description
Criteria for linear dependency of the basis and fit set

AllowBasisDependency

Type
Bool

Default value
Yes

Description
Project out the dependent part of the basis set (associated with small eigenvalues of the overlap matrix).

Basis

Type
Float

Default value
1e-08

GUI name
Dependency criterion

Description
Criteria for linear dependency of the basis: smallest eigenvalue of the overlap matrix of normalized Bloch functions.

Core

Type
Float

Default value
0.8

Description
The program verifies that the frozen core approximation is reasonable, by checking the smallest eigen value of the overlap matrix of the core (Bloch) orbitals (which should ideally be one) is bigger than this criterion.

CoreValence

Type
Float

Default value
1e-05

Description
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-corevalence

`Fit`

Type
Float

Default value
5e-06

Description
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.

4.8.3 Screening

Band performs many lattice summations which are in practice truncated. The two prime examples are the construction of the Bloch basis and the calculation of the solvation potential. The precision of the lattice summations is controlled by the `Screening` key

```
Screening
   CutOff float
   DMadel float
   NoDirectionalScreening Yes/No
   RCelx float
   RMadel float
End
```

`Screening`

Type
Block

Description
For the periodic solvation potential and for the old (not default anymore) fitting method, BAND performs lattice summations which are in practice truncated. The precision of the lattice summations is controlled by the options in this block.

`CutOff`

Type
Float

Description
Criterion for negligibility of tails in the construction of Bloch sums. Default depends on Accuracy.

`DMadel`

Type
Float

Description
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

NoDirectionalScreening

Type
Bool

Description
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. Screening is only applied in the periodicity directions. This key restores the original situation: screening in all directions

RCelx

Type
Float

Description
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

Type
Float

Description
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.

4.8.4 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 bool
End
```

4.8.5 Fermi energy search

```
Fermi
   Delta float
   Eps float
   MaxTry integer
   RefinePostSCFFactor integer
End
```

Fermi

Type
Block

Description
Technical parameter used in determining the Fermi energy, which is carried out at each cycle of the SCF procedure.

`Delta`

Type
Float

Default value
0.0001

Description
Convergence criterion: upper and lower bounds for the Fermi energy and the corresponding integrated charge volumes must be equal within delta.

`Eps`

Type
Float

Default value
1e-10

Description
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.

`MaxTry`

Type
Integer

Default value
15

Description
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.

`RefinePostSCFFactor`

Type
Integer

Description
Use a finer k-grid after the scf to calculate a refined fermi level. Makes only sense for metals. Works like DoubleCount. Use 1,2,3

4.8.6 Block size

```
CPVector integer
```

CPVector

Type
Integer

Default value
128

GUI name
Vectorlength (blocksize)

Description
The code is vectorized and this key can be used to set the vector length

```
KGrpX integer
```

KGrpX

Type
Integer

Default value
5

GUI name
Number of K-points at once

Description
Absolute upper bound on the number of k-points processed together. This only affects the computational performance.

CHAPTER

FIVE

SPECTROSCOPY AND PROPERTIES

5.1 Frequencies and Phonons

Frequencies and Phonons and can be computed via numerical differentiation by the AMS diver. See the Normal Modes section or the Phonon section of the AMS manual.

Several thermodynamic properties, such as Zero-point Energy, Internal Energy, Entropy, Free Energy and Specific Heat are computed by default when calculating Phonons.

5.2 Elastic Tensor

The elastic tensor (and related elastic properties such as Bulk modulus, Shear modulus and Young modulus) can be computed via numerical differentiation by AMS. See the Elastic Tensor section of the AMS manual.

When calculating the elastic tensors using Band one should disable *Symmetry* (page 189).

5.3 Optical Properties: 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 115) or in *OldResponse* (page 121).

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

- Tutorial: Silicon (OldResponse)
- Tutorial: MoS2 Monolayer (NewResponse)
- *Example: TD-CDFT for bulk diamond (OldResponse)* (page 284)

5.3.1 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:

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

5.3.2 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 (https://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 (https://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**.

5.3.3 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 $(1/3\mathrm{Tr}(\varepsilon_M(\omega))$). 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[1][2][3][4].

[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).

[1] 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). (https://doi.org/10.1063/1.481315)

[2] 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) (https://doi.org/10.1103/PhysRevB.71.155108).

[3] 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) (https://doi.org/10.1103/PhysRevB.71.155104).

[4] 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) (https://doi.org/10.1063/1.2780146).

5.3.4 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 121) key block. The new code section, relying on more modern algorithms of BAND, is accessible via the *NewResponse* (page 115), *NewResponseSCF* (page 117) and *NewResponseKSpace* (page 120) key blocks. The differences between the two flavors 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	(yes)
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 flavors 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 reciprocal 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** *K-Space* (page 66) **options!!!**

NewResponse

The dielectric function is computed when the key block *NewResponse* (page 115) is present in the input. Several important settings can be defined in this key block.

Additional details can be specified via the *NewResponseKSpace* (page 120) and *NewResponseSCF* (page 117) blocks.

```
NewResponse
   NFreq integer
   FreqLow float
   FreqHigh float
   EShift float
   ActiveESpace float
   DensityCutOff float
   ActiveXYZ string
End
```

NewResponse

Type
Block

Description
The TD-CDFT calculation to obtain the dielectric function is computed when this block is present in the input. Several important settings can be defined here.

NFreq

Type
Integer

Default value
5

Description
Number of frequencies for which a linear response TD-CDFT calculation is performed.

FreqLow

Type
Float

Default value
1.0

Unit
eV

Description
Lower limit of the frequency range for which response properties are calculated. (omega_{low})

FreqHigh

Type
Float

Default value
3.0

Unit
eV

Description
Upper limit of the frequency range for which response properties are calculated (omega_{high}).

EShift

Type
Float

Default value
0.0

Unit
eV

GUI name
Shift

Description
Energy shift of the virtual crystal orbitals.

ActiveESpace

Type
Float

Default value
5.0

Unit
eV

GUI name
Active energy space

Description
Modifies the energy threshold (DeltaE^{max}_{thresh} = omega_{high} + ActiveESpace) for which single orbital transitions (DeltaEpsilon_{ia} = Epsilon_{a}^{virtual} - Epsilon_{i}^{occupied}) are taken into account.

`DensityCutOff`

Type
Float

Default value
0.001

GUI name
Volume cutoff

Description
For 1D and 2D systems the unit cell volume is undefined. Here, the volume is calculated as the volume bordered by the isosurface for the value DensityCutoff of the total density.

`ActiveXYZ`

Type
String

Default value
t

Description
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
   Bootstrap integer
   COApproach Yes/No
   COApproachBoost Yes/No
   Criterion float
   DIIS
      Enabled Yes/No
      MaxSamples integer
      MaximumCoefficient float
      MinSamples integer
      MixingFactor float
   End
   LowFreqAlgo Yes/No
   NCycle integer
   XC integer
End
```

`NewResponseSCF`

Type
Block

Description
Details for the linear-response self-consistent optimization cycle. Only influencing the NewResponse code.

`Bootstrap`

Type
Integer

Default value
0

Description
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!

`COApproach`

Type
Bool

Default value
Yes

Description
The program automatically decides to calculate the integrals and induced densities via the Bloch expanded atomic orbitals (AO approach) or via the cyrstal orbitals (CO approach). The option COApproach overrules this decision.

`COApproachBoost`

Type
Bool

Default value
No

GUI name
CO Approach Boost

Description
Keeps the grid data of the Crystal Orbitals in memory.

Requires significantly more memory for a speedup of the calculation. One might have to use multiple computing nodes to not run into memory problems.

`Criterion`

Type
Float

Default value
0.001

Description
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.

`DIIS`

Type
Block

Description
Parameters influencing the DIIS self-consistency method

Enabled

Type
Bool

Default value
Yes

Description
If not enabled simple mixing without DIIS acceleration will be used.

MaxSamples

Type
Integer

Default value
10

Description
Specifies the maximum number of samples considered during the direct inversion of iteration of subspace (DIIS) extrapolation of the atomic charges during the SCC iterations. A smaller number of samples potentially leads to a more aggressive convergence acceleration, while a larger number often guarantees a more stable iteration. Due to often occurring linear dependencies within the set of sample vectors, the maximum number of samples is reached only in very rare cases.

MaximumCoefficient

Type
Float

Default value
10.0

Description
When the diis expansion coefficients exceed this threshold, the solution is rejected. The vector space is too crowded. The oldest vector is discarded, and the expansion is re-evaluated.

MinSamples

Type
Integer

Default value
-1

Description
When bigger than one, this affects the shrinking of the DIIS space on linear dependence. It will not reduce to a smaller space than MinSamples unless there is extreme dependency.

MixingFactor

Type
Float

Default value
0.2

Description
The parameter used to mix the DIIS linear combination of previously sampled atomic charge vectors with an analogous linear combination of charge vectors resulting from population analysis combination. It can assume real values between 0 and 1.

LowFreqAlgo

Type
Bool

Default value
Yes

GUI name
Low Frequency Algorithm

Description
Numerically more stable results for frequencies lower than 1.0 eV. Note: for a graphene monolayer the conical intersection results in a very small band gap (zero band gap semi-conductor). This leads ta a failing low frequency algorithm. One can then chose to use the algorithm as originally proposed by Kootstra by setting the input value to *false*. But, this can result in unreliable results for frequencies lower than 1.0 eV!

NCycle

Type
Integer

Default value
20

GUI name
Cycles

Description
Number of SCF cycles for each frequency to be evaluated.

XC

Type
Integer

Default value
1

Description
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).

```
NewResponseKSpace
   Eta float
   SubSimp integer
End
```

NewResponseKSpace

Type
Block

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

Eta

Type
Float

Default value
1e-05

Description
Defines the small, finite imaginary number i*eta which is necessary in the context of integration weights for single-particle transitions in reciprocal space.

SubSimp

Type
Integer

Default value
3

Description
determines into how many sub-integrals 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. Note: the computing time for the weights is linear for 1D, quadratic for 2D and cubic for 3D!

OldResponse

```
OldResponse
   Berger2015 Yes/No
   CNT Yes/No
   CNVI float
   CNVJ float
   Ebndtl float
   Enabled Yes/No
   Endfr float
   Isz integer
   Iyxc integer
   NewVK Yes/No
   Nfreq integer
   QV Yes/No
   Shift float
   Static Yes/No
   Strtfr float
End
```

OldResponse

Type
Block

Description
Options for the old TD-CDFT implementation.

Berger2015

Type
Bool

Default value
No

Description
Use the parameter-free polarization functional by A. Berger (Phys. Rev. Lett. 115, 137402). This is possible for 3D insulators and metals. Note: The evaluation of response properties using the Berger2015 is recommend for 3D systems only!

CNT

Type
Bool

Description
Use the 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.

CNVI

Type
Float

Default value
0.001

Description
The first convergence criterion for the change in the fit coefficients for the fit functions, when fitting the density.

CNVJ

Type
Float

Default value
0.001

Description
the second convergence criterion for the change in the fit coefficients for the fit functions, when fitting the density.

Ebndtl

Type
Float

Default value
0.001

Unit
Hartree

Description
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.

Enabled

Type
Bool

Default value
No

Description
If true, the response function will be calculated using the old TD-CDFT implementation

Endfr

Type
Float

Default value
3.0

Unit
eV

Description
The upper bound frequency of the frequency range over which the dielectric function is calculated

Isz

Type
Integer

Default value
0

Description
Integer indicating whether or not scalar zeroth order relativistic effects are included in the TD-CDFT calculation. 0 = relativistic effects are not included, 1 = relativistic effects are included. The current implementation does NOT work with the option XC%SpinOrbitMagnetization equal NonCollinear

Iyxc

Type
Integer

Default value
0

Description
integer for printing yxc-tensor (see http://aip.scitation.org/doi/10.1063/1.1385370). 0 = not printed, 1 = printed.

NewVK

Type
Bool

Description
Use the slightly modified version of the VK kernel (see https://aip.scitation.org/doi/10.1063/1.1385370). When using this option one uses effectively the static option, even for metals, so one should check carefully the convergence with the KSPACE parameter.

Nfreq

Type
Integer

Default value
5

Description
the number of frequencies for which a linear response TD-CDFT calculation is performed.

`QV`

Type
Bool

Description
Use the QV parametrization for the longitudinal and transverse kernels of the XC kernel of the homogeneous electron gas. Use this in conjunction with the NewVK option. (see reference).

`Shift`

Type
Float

Default value
0.0

Unit
eV

Description
energy shift for the virtual crystal orbitals.

`Static`

Type
Bool

Description
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.

`Strtfr`

Type
Float

Default value
1.0

Unit
eV

Description
is the lower bound frequency of the frequency range over which the dielectric function is calculated.

5.4 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[1] and[2].

[1] 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) (https://doi.org/10.1021/jp800494m).

[2] 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) (https://doi.org/10.1021/jp805466c).

Hyperfine A-tensor

The A-tensor is implemented within the non-relativistic and scalar relativistic, spin-polarized Kohn-Sham scheme.

```
ATensor
   Enabled Yes/No
End
```

ATensor

Type
Block

Description
Hyperfine A-tensor.

Enabled

Type
Bool

Default value
No

GUI name
:A-tensor

Description
Compute the hyperfine A-tensor.

Note: Unrestricted calculation is required.

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 within the ESR block:

```
ESR
   Enabled Yes/No
End
```

ESR

Type
Block

Description
Zeeman g-tensor. 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 following options are necessary for ESR: 'Relativistic zora spin' and 'Kspace 1'

Enabled

Type
Bool

Default value
No

GUI name
ESR: g-tensor

Description
Compute Zeeman g-tensor.

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 following options are necessary for ESR: 'Relativistic zora spin' and '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.

5.5 Nuclear Quadrupole Interaction (EFG)

```
EFG
   Enabled Yes/No
End
```

EFG

Type
Block

Description
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.

Enabled

Type
Bool

Default value
No

GUI name
EFG (electric field gradient): Calculate

Description
Compute the EFG tensor (for nuclear quadrupole interaction).

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

5.6 NMR

Warning: The calculations of NMR shielding with BAND has not been thoroughly tested and the results might be unreliable. One should be extra careful when running NMR calculation, and validate the results by using different super-cells and different technical parameters.

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

I) The super cell method is according to the implementation by Skachkov *et al.*[1] The symmetry will be automatically disabled. The unit cell should not be chosen too small.

II) The other method is the single-dipole method. In principle one can now use the primitive cell[2]. 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
   Enabled Yes/No
   SuperCell Yes/No
End
```

NMR

Type
Block

Description
Options for the calculations of the NMR shielding tensor.

Enabled

Type
Bool

Default value
No

Description
Compute NMR shielding.

SuperCell

[1] 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) (https://doi.org/10.1021/ct100046a).

[2] 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) (https://doi.org/10.1139/V11-050).

Type
Bool

Default value
Yes

Description
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.

5.7 Effective Mass

The physics of the effective mass tensor is explained in this tutorial

A rather important aspect is the determination of the top of the valence band (TOVB or HOMO) and the bottom of the conduction band (BOCB or LUMO). There are two ways to do this

- 1) IPOL: from the interpolated bands
- 2) PATH: from the calculated bands along the high symmetry path, as used by amsbands.

In case that the HOMO and LUMO are located on the high symmetry path the PATH method is more accurate. A good example are 2D systems with a triangular lattice. Often the HOMO or LUMO are at the point called "K", which may not be present in the standard k-grid, but is always present on the high symmetry path. Also, along the path a dense k-grid is used. It remains possible that the extrema or not on the path and then the IPOL method should be used (possibly using a denser k-grid than usual). When chooising the PATH method some feedback is printed about whether IPOL and PATH produce the same HOMO and LUMO energy and k coordinates.

5.7.1 Input key block

```
EffectiveMass
   Enabled Yes/No
   KPointCoord float_list
   NumAbove integer
   NumBelow integer
   StepSize float
   UseBandStructureInfoFromPath Yes/No
End
```

`EffectiveMass`

Type
Block

Description
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. The easiest way to use this key is to enabled it without specifying any extra options.

`Enabled`

Type
Bool

Default value
No

GUI name
Effective mass

Description
Compute the EffectiveMass.

`KPointCoord`

Type
Float List

Unit
1/Bohr

Recurring
True

GUI name
At K-point

Description
Coordinate of the k-points for which you would like to compute the effective mass.

`NumAbove`

Type
Integer

Default value
1

GUI name
Include N bands above

Description
Number of bands to take into account above the Fermi level.

`NumBelow`

Type
Integer

Default value
1

GUI name
Include N bands below

Description
Number of bands to take into account below the Fermi level.

`StepSize`

Type
Float

Default value
0.001

Description
Size of the step taken in reciprocal space to perform the numerical differentiation

`UseBandStructureInfoFromPath`

Type
Bool

Default value
Yes

Description
The (automatic) location of the HOMO and LUMO can be determined via band interpolation, or from the path as used by the BandStructure feature. The latter works better when they are located on the path. See also comments in the BandStructure block. To reproduce results from before ams2025 set to no.

5.8 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.

5.9 X-Ray Form Factors

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

Form factors options:

```
FormFactors integer
```

`FormFactors`

Type
Integer

Default value
2

Description
Number of stars of K-vectors for which the form factors are computed

5.10 Dipole moment and Berry Phase

By default, Band computes (and prints to the output) only the components of the dipole moment orthogonal to the periodic direction(s).

Since the position operator is ill-defined in periodic systems, computing the longitudinal components of the dipole moment (i.e. the components along periodic directions) is not completely trivial. To obtain the longitudinal components of the dipole moment, you can perform a Berry Phase calculation:

```
BerryPhase Yes/No
```

`BerryPhase`

Type
Bool

Default value
No

Description
Boolean that determines whether the dipole as determined through the Berry phase approach should be calculated.

In the graphical user interface (AMSinput) you can find the Berry Phase checkbox in the **Details → Expert BAND** panel.

In a Berry phase calculation, the dipole moment of a unit cell is calculated with the help of the geometric phase within the unit cell. The theoretical framework of the calculations originates from the *'Modern theory of polarization'* that was come up with in the early 1990s by King-Smith, Vanderbilt and Resta.[1][2] The implementation in BAND is a generalization of the one-dimensional Berry phase approach for quantum chemistry codes with local basis sets devised by Kudin and Car.[3]

Warning: The Berry phase implementation in BAND has been thoroughly for 1D systems. Nonetheless, the implementation seems to break down for 2D and 3D systems, thus requiring careful testing and validation of the calculations for such systems.

Warning:

- Berry phase calculations require orthorhombic unit cells, differently shaped unit cells are not currently supported.
- For Berry Phase calculations you should use an all *electron basis set* (page 54) (i.e. Set the `core` to `none`)
- Using a good k-space sampling is recommended for Berry Phase calculations

An option that can be useful when validating the Berry Phase calculation is `ShiftCoordinates` in the AMS System block:

```
System header
   ShiftCoordinates float_list
End
```

[1] R. King-Smith, D. Vanderbilt, *Theory of polarization of crystalline solids.* Physical Review B 47, 1651 (1993) (https://doi.org/10.1103/PhysRevB.47.1651).

[2] R. Resta, *Macroscopic polarization in crystalline dielectrics: the geometric phase approach.* Reviews of Modern Physics 66, 899 (1994) (https://doi.org/10.1103/RevModPhys.66.899).

[3] K. Kuding, R. Car, *Berry phase approach to longitudinal dipole moments of infinite chains in electronic-structure methods with local basis sets* Journal of Chemical Physics 126, 234101 (2007) (https://doi.org/10.1063/1.2743018).

System

ShiftCoordinates

Type
Float List

Unit
Bohr

Description
Translate the atoms by the specified shift (three numbers).

5.11 GW

5.11.1 General

This page describes the basic procedure, usage and scope of a GW calculation. Technical details of the algorithm can be tweaked in the *MBPT input block* (page 102).

See also:

Examples GW (page 291)

The GW method is a relatively accurate method to obtain information about so-called charged excitations, or single-particle excitations. We refer to them as Quasiparticle energies. These are especially important to interpret and predict the outcome of direct and inverse photo-emission spectroscopy and can be used to obtain e.g. very accurate ionization potentials and electron affinities which gives access to the so-called fundamental gap (not to be confused with the optical gap).

GW can be used icw scalar relativistic effects within the `ZORA` formalism.

In practice, fully self-consistent GW is rarely used for molecular systems. Instead, perturbative approximations, so called quasiparticle GW methods are used since they are cheaper and also more accurate than fully self-consistent GW. The most popular approach is G_0W_0[1], in which quasiparticle energies are obtained as a one-shot perturbative correction to KS eigenvalues. A downside of this approach is the rather pronounced starting-point dependence. This can be overcome to a large extent in eigenvalue-only self-consistent GW (evGW), in which the quasiparticle energies (but not the density) are iteratively updated until self-consistency is reached. In quasiparticle self-consistent GW, also the density is updated in each iteration, so that the results become completely starting point independent. The partially self-consistent GW variants usually converge within 6-8 iterations which make these approaches a factor of 6 to 8 more expensive than G_0W_0.

A GW calculation as implemented in BAND proceeds in five steps as has been described in detail in a series of papers[Page 132, 156]. The technique is also known as the as GW space-time method. Our implementation is closely related to the scheme outlined in[4].

- A DFT single-point calculation is performed. This can be an LDA, GGA or hybrid calculation. At the moment, BAND does not support the use of XCfun, but libXC is supported. Valid choices could be LDA, PBE, BLYP, PBE0, BHandHLYP and many more. The default is LDA. As usual, the functional to be used during the SCF is requested in the *XC input block* (page 15) block. The following requests a G_0W_0 calculation with default settings using a PBE reference:

[1] Arno Förster, Lucas Visscher, *Low-order scaling G0W0 by pair atomic density fitting*, Journal of Chemical Theory and Computation 16 (12), 7381–7399 (2020) (https://doi.org/10.1021/acs.jctc.0c00693)

[5] Arno Förster, Lucas Visscher, *GW100: A Slater-Type Orbital Perspective*, Journal of Chemical Theory and Computation 17 (8), 5080–5097 (2021) (https://doi.org/10.1021/acs.jctc.1c00308)

[6] Arno Förster, Lucas Visscher, *Low-Order Scaling Quasiparticle Self-Consistent GW for Molecules*, frontiers in Chemistry 9, 736591 (2021) (https://doi.org/10.3389/fchem.2021.736591)

[4] J. Wilhelm, D. Golze, L. Talirz, J. Hutter, C.A. Pignedoli, *Toward GW Calculations on Thousands of Atoms*, Journal of Physical Chemistry Letters 9 (2), 306-312 (2012) (https://doi.org/10.1021/acs.jpclett.7b02740)

```
XC
   GGA PBE
END

GW
End
```

Hybrid starting points can be requested with the `LibXC` key.

- From the KS orbitals and orbital energies, a Green's function (G) is evaluated and from the Green's function the so-called independent-particle polarizability is calculated. This is done in imaginary time.
- The polarizability is Fourier transformed to the imaginary frequency axis where the screened Coulomb potential (W) is evaluated using the Coulomb potential and the polarizability.
- The screened Coulomb potential is Fourier transformed to imaginary time. Here, the self-energy is calculated using G and W (that why it is called GW) which gives access to spectroscopic properties.
- The self-energy is transformed to the molecular orbital basis from where it is Fourier transformed to the imaginary frequency axis from where it is analytically continued to the complex plane. Along the real frequency axis, the quasiparticle energies are evaluated.
- In case of evGW (evGW$_0$), the input KS eigenvalues are replaced by the quasiparticle energies from the previous iteration and the scheme is iterated until self-consistency in the quasiparticle energies is reached.
- In case of qsGW (qsGW$_0$), A non-local, hermitian, and static exchange-correlation potential is constructed from the self-energy. This exchange-correlation potential replaces the KS exchange-correlation potential. Diagonalization gives a new set of single-particle orbitals and quasiparticle energies. The procedure is repeated until self-consistency is reached.
- By default, the DIIS algorithm is used to accelerate and stabilize convergence of the self-consistent GW schemes. A linear-mixing scheme can also be used.

The GW space-time method has the distinct advantage that it can be very fast, while a full frequency, conventional GW calculation scales to the 6th power of the system size and is prohibitive for systems larger than a few tens of atoms. BAND used advanced density fitting options to accelerate the space-time method further and in practice nearly quadratic scaling can be reached. This enables the routine application of the method to systems of several hundreds of atoms. A G_0W_0 calculation (without the preceding SCF) is usually not much slower than a hybrid calculation. The downside of the approach is that the analytical continuation technique produces large errors (up to several eV) for core states which are in example important in X-ray spectroscopy. Thus, the GW implementation in BAND should only be used to predict quasiparticle energies for states in the valence-region. In fact, we have only tested it for HOMO and LUMO states (which are arguably most important)

The states of interest can be requested in the GW block

```
GW
  nStates 5
End
```

is the default and calculates 5 occupied and five unoccupied states.

```
GW
  nStates 1
End
```

calculates the HOMO and LUMO quasiparticle energy only.

5.11.2 Levels of self-consistency

Eigenvalue-only self-consistent GW

An evGW calculation is requested by

```
GW
  SelfConsistency evGW
End
```

One can also only iterate G by keep W fixed which reduces the cost of each iteration by roughly 50 %. This is requested by

```
GW
  SelfConsistency evGW0
End
```

On the other hand, much of the starting point dependence of the G_0W_0 method remains in the evGW$_0$ method and it is generally not recommended.

quasiparticle self-consistent GW

An qsGW calculation is requested by

```
GW
  SelfConsistency qsGW
End
```

One can also only iterate G by keep W fixed which reduces the cost of each iteration by roughly 50 %. This is requested by

```
GW
  SelfConsistency qsGW0
End
```

On the other hand, much of the starting point dependence of the G_0W_0 method remains in the qsGW$_0$ method and it is generally not recommended.

More options:

QPHamiltonian
: In quasiparticle self-consistent GW, the frequency-dependent self-energy is mapped to a static exchange-correlation in each iteration. The mapping is not unique and different schemes have been suggested. They differ in the way the frequency dependence of the self-energy is treated. In BAND, three variants can be used. KSF1 and KSF2 are from the paper by Kotani et al.[7]. KSF1 refers to eq. 10 therein and is most commonly implemented, KSF2 refers to eq. 11 therein. KSF2 is the default in BAND since it is numerically more stable. The variant denoted as Kutepov has been suggested bu Kutepov et al.[8] and uses a first-order expansion around the chemical potential.

FixedGrids
: Per default, the imaginary frequency and imaginary time grids used in the GW calculation are updated in each iteration in a qsGW calculation since this is necessary to make the results strictly starting point dependent. This can also be turned off, and fixed grids are used throughout. This might be helpful in case of convergence problems.

[7] A.L. Kutepov, V.S. Oudovenko, G. Kotliar, *Linearized self-consistent quasiparticle GW method: Application to semiconductors and simple metals*, Computer Physics Communications 407-414 (2017) (http://dx.doi.org/10.1016/j.cpc.2017.06.012)

[8] Takao Kotani, Mark Van Schilfgaarde, Sergey V. Faleev, *Quasiparticle self-consistent GW method: A basis for the independent-particle approximation*, Physical Review B 76 (16) 1-24 (2007) (https://doi.org/10.1103/PhysRevB.76.165106)

Convergence

The self-consistency can be controlled by a few parameters: For example

```
GW
  Converge HOMO=5e-3
End
```

requests that the evGW(0) calculation is considered converged if the difference between the HOMO quasiparticle energies of 2 consecutive iterations does not change by more than 5 meV. The default is 1 meV which is in practice usually a little too tight. We recommend to adjust this parameter according to your requirements, for example the experimental resolution you would like to match.

For qsGW (qsGW$_0$), the change in the norm of the density matrix is used as an additional criterion to control convergence. In evGW (evGW$_0$) it is ignored.

```
GW
  Converge Density=1e-07
End
```

is the default. For very large systems and when QZ basis sets are used, it is recommended to decrease that value, for example to

```
GW
  Converge Density=1e-08
End
```

```
GW
  linearmixing 0.2
End
```

turns of DIIS and request to use linear mixing instead with a mixing parameter of 0.2. This can be useful if for some reason, convergence using DIIS cannot be achieved. However, it is usually better to adjust the number of vectors in the DIIS expansion. This is achieved by (for example)

```
GW
  DIIS 5
End
```

The default are 10 expansion vectors. In case of difficulties converging

5.11.3 Second-order self-energy

Starting from ADF2BAND, it is possible to go beyond the GW approximation for the self-energy (expansion in screened interaction to first order) and also take into account the next term (expansion in the screened interaction to second order). The algorithm has been described in this paper. ..[10]

- This can be used with all variants of self-consistency.
- The second-order self-energy is always applied as a perturbative correction of the GW quasiparticle energies, using a statically screened interaction W. More precisely,

$$\Sigma^{GW} = G(\omega) * W(\omega)$$

$$\Sigma^{GW+G3W2} = G(\omega) * W(\omega) + G(\omega) * W(\omega = 0) * G(\omega) * G(\omega) * W(\omega = 0)$$

[10] A. Förster, L. Visscher, *EGW100: A Slater-Type Orbital Perspective*, Journal of Chemical Theory and Computation 17(8), 5080-5097 (2021) (https://doi.org/10.1021/acs.jctc.1c00308)

- The second-order self-energy is activated by typing

```
GW
  selfenergy G3W2
End
```

- It has been shown that the second-order correction is especially accurate for electron affinities and HOMO-LUMO gaps.[Page 135, 10] Since it effectively scales as the fourth power of the system size, it should not be used for systems much larger than 50 atoms.

5.11.4 Recommendations

Basis sets

The recommended numerical settings depend strongly on the basis set. The recommended basis set depends on system size and the property of interest. The following are recommendations which should be seen as guidelines and not as definite. We recommend to always verify the basis set convergence of for the property of interest. Details about basis set convergence can be found in this paper[Page 135, 10]

Basis sets

All-electron: All-electron calculations are always recommended.
Basis set size: Larger basis sets are needed to achieve the same accuracy as in a DFT calculation
Augmented basis sets: Should only be used if absolutely necessary. They are however often necessary for an accurate description of the electron affinity for molecules with LUMO above the vacuum level.

Ionization Potentials: TZ2P or larger
Electron affinities, bound LUMO: Corr/TZ3P or larger
Electron affinities, unbound LUMO: AUG/ATZ2P or larger
Fundamental Gaps: Triple-zeta quality basis sets are typically sufficient.

The GW-BSE excitation energies and the screened Coulomb interaction do only depend on the QP gaps and not on their absolute values. Therefore, triple-zeta quality basis sets are typically sufficient.

For **highly accurate results** we recommend to perform an extrapolation to the complete basis set limit: For this, perform 2 calculations using the Corr/TZ3P and Corr/QZ6P. The basis set limit is then calculated according to

$$\epsilon_n^{CBS} = \epsilon_n^{QZ} - \frac{1}{N_{bas}^{QZ}} \frac{\epsilon_n^{QZ} - \epsilon_n^{TZ}}{\frac{1}{N_{bas}^{QZ}} - \frac{1}{N_{bas}^{TZ}}}$$

The values for N_{bas}^{TZ} and N_{bas}^{QZ} can be found in the BAND.rkf file in the `GW` section under the entry `nBas`. Additional explanations can be found in the references[3] and[Page 135, 10]

[3] M.J. Van Setten, F. Caruso S. Sharifzadeh X. Ren M. Scheffler F. Liu J. Lischner L. Lin J. Deslippe S.G. Louie C. Yang F. Weigend J.B. Neaton F. Evers P. Rinke, *GW100: Benchmarking G0W0 for Molecular Systems*, Journal of Chemical Theory and Computation 11 (12), 5665-5687 (2015) (https://doi.org/10.1021/acs.jctc.5b00453)

Numerical aspects

- According to the choice of basis set, the recommended numerical settings can differ. For a discussion, see the *MBPT page* (page 102).
- qsGW has the highest requirements on the numerical parameters than evGW and G_0W_0. Usually, it is necessary to use a larger value in the `Dependency` key, see the *MBPT page* (page 102). Note that starting from AMS2022 BAND will use a `Dependency bas=5e-3` and `RIHartreeFock DependencyThreshold=5e-3` in case of (any variant of) GW. One can override these values in the input.
- The implemented GW algorithm is very sensitive to numerical noise and depending on the numerical settings and/or the underlying exchange-correlation functional, (occupied) quasiparticle energies from partially self-consistent GW calculations performed on different hardware can differ by a few meV. The discrepancies are generally more pronounced for core states for which the analytical continuation technique is rather inaccurate.
- In our experience, these differences only occur with large basis sets (QZ4P or larger) and when Minnesota functionals (we tested M06, M06-2X and M06-HF) are used to calculate the KS reference. When the DIIS algorithm is used to converge the quasiparticle energies, the number of iterations needed for convergence can then differ as well.
- For optimal accuracy, full cores should be used. Note, that frozen cores do not require in large computational savings for GW calculations.

Choosing the KS reference

- As for ground-state properties, it is far from trivial to recommend a universal functional. However, for G_0W_0 calculations, a few general guidelines can be offered. A good discussion is found in[2]. It should be noted, that GGA functional should not be used, even though a PBE starting point is a popular choice. We rather recommend to use a hybrid functional. The relevant parameter in the choice of the hybrid is usually the fraction of exact exchange. In our experience, PBE0 with 40-50 % exact exchange is a good choice. You might also use a range-separated hybrid via LibXC, for example LRC-wPBEH.
- If in doubt, one of the partially self-consistent schemes should be used. evGW is almost starting point independent and qsGW is completely starting point independent. Therefore, for qsGW, it is in principle irrelevant what starting point is chosen. Convergence properties can in principle be affected, although the number of iterations until convergence is reached is also more or less independent of the starting point[Page 132, 6]. Also for evGW, starting from a hybrid functional or a range-separated functional usually results in the highest accuracy.

5.11.5 GW key

```
GW
   nStates integer
   SelfEnergy [HF | GW | G3W2 | SOSEX | GWGamma | G3W2dynamic | GWGammaInf |␣
↪GWGammaInfDyn]
   SelfConsistency [G0W0 | EVGW0 | EVGW | QSGW0 | QSGW]
   QPHamiltonian [KSF1 | KSF2 | SRG | LQSGW]
   nIterations integer_list
   Converge
      Density float_list
      HOMO float
   End
```

(continues on next page)

[2] F. Bruneval, M.A.L., Marques, *Benchmarking the starting points of the GW approximation for molecules*, Journal of Chemical Theory and Computation 9 (1), 324-329 (2013) (https://dx.doi.org/10.1021/ct300835h)

(continued from previous page)

```
   LinearMixing float_list
   AdaptiveMixing float_list
   DIIS integer
   LinearizeQPequations Yes/No
   ScissorShift Yes/No
End
```

GW

Type
Block

Description
Perform a GW calculation. G0W0 is the default for GW%SelfConsistency.

nStates

Type
Integer

Default value
5

GUI name
N states

Description
Number of Quasiparticle States to be printed to output.

The default is 5 states which in this case means that min(5, Number of particle states) occupied and min(5, Number of hole states) hole states are printed. The whole list of states can be printed by setting this parameter to -1'

SelfEnergy

Type
Multiple Choice

Default value
GW

Options
[HF, GW, G3W2, SOSEX, GWGamma, G3W2dynamic, GWGammaInf, GWGammaInfDyn]

Description
Controls the form of the self-energy.

GW is the default and corresponds to the standard GW calculation.

G3W2 is a GW calculation plus a perturbative second-order statically screened exchange correction (second order expansion in the self-energy). Note, that there the self-energy is always static.

SelfConsistency

Type
Multiple Choice

Default value
G0W0

Options
[G0W0, EVGW0, EVGW, QSGW0, QSGW]

Description
Sets the level of self-consistency in a GW calculation.

G0W0 calculates a one-shot, perturbative correction to the KS eigenvalues.

In evGW and evGW0, the quasi-particle energies are updated until self-consistency is reached.

evGW0 requests that the Green's function is evaluated self-consistently but not the screened interaction.

In qsGW, the density is updated as well, however, the self-energy is mapped to a static effective potential and the Dyson equation is solved by diagonalization instead of inversion. The results of a qsGW are independent of the choice of the underlying exchange-correlation functional and are usually the most accurate ones.

The same is done in qsGW0, but the screened interaction is not updated.

`QPHamiltonian`

Type
Multiple Choice

Default value
KSF2

Options
[KSF1, KSF2, SRG, LQSGW]

Description
The quasi-particle Hamiltonian can be constructed in different ways.

KSF1 refers to the original construction by Kotani, Van Schilfgaarde anf Faleev (KSF) which is also implemented in TURBOMOLE.

KSF2 refers to an alternative construction by KSF.

KSF1 is not recommended since it is numerically less stable than KSF2 in case analytical continuation is used (the default).

In case the analytical integration algorithm is used, only KSF1 is implemented. Therefore, make sure that KSF1 is specified. The results are typically very similar.

The QP energies at which the matrix elements are evaluated can be tweaked further, see the two subsequent keys: However, KSF2 is recommended since it typically leads to QP energies with the best agreement with experiment.

Ignored when not a quasi-particle self-consistent GW calculation is performed

`nIterations`

Type
Integer List

Default value
[10]

GUI name
Number of iterations

Description
The maximum number of iterations within the (partially or fully) self-consistent GW calculation has to converge.

Ignored when Formalism is set to G0W0

Converge

Type
Block

Description
Sets convergence criteria for the GW calculation in self-consistent case

Density

Type
Float List

Default value
[1e-08, 1e-05]

Description
First Criterion for self-consistency procedure to terminate.

Criterion is the trace of the density matrix. Ignored in non-selfconsistent Calculation and in eigenvalue self-consistent GW

It is possible to run a qsGW calculation with an inner SCF loop which updates the static part of the elf-energy only. This can be useful to accelerate the convergence in case linear mixing is used. It is not recommended to use linear mixing, so it is also not recommended to use that inner loop as well. The second number in this list specifies the convergence criterion for the inner SCF loop.

HOMO

Type
Float

Default value
0.003

Unit
eV

GUI name
HOMO energy convergence

Description
Criterion for self-consistency procedure to terminate.

The self-consistent GW calculation terminates, when the difference between the HOMO QP energies between 2 consecutive iterations is below this number.

The LUMO energy converged faster than the HOMO energy so when the HOMO energy is converged according to this criterion, the LUMO energy will be converged as well.

In non-selfconsistent Calculation, this criterion is ignored.

LinearMixing

Type
Float List

Description
Requests to use linear mixing instead of DIIS and sets the mixing parameter for linear mixing of Green's function in case of self-consistency.

It is ignored in non-selfconsistent calculation and overwritten by DIIS when DIIS is also present.

AdaptiveMixing

Type
Float List

Description
Requests to use adaptive mixing instead of DIIS and sets the staring mixing parameter for mixing of Green's function in case of self-consistency.

Adapative mixing is recommended in case a qsGW calculation does not converge with DIIS.

It is ignored in non-selfconsistent calculation and overwritten by DIIS when DIIS is also present.

`DIIS`

Type
Integer

Default value
10

Description
Requests to use DIIS. This is the Default. Number of expansion coefficients can be requested as well. Ignored in non-selfconsistent calculation

`LinearizeQPequations`

Type
Bool

Default value
No

Description
Instead of solving the non-linear QP equations in a G0W0 (or evGW calculation) by bisection exacly, linearize them by first-order Taylor expansion.

This is not recommended since it does not save computational time when used together with analytical continuation (as implemented in AMS). It might however be useful for benchmarking or for validating results.

If the results os the linearization differ by a lot (for instance, more than 0.1 eV in frontier QP energies) from the non-linearized results, this might indicate that the GW calculation is not reliable.

`ScissorShift`

Type
Bool

Default value
No

Description
Only calculate the HOMO and LUMO QP energies and shift the remaining QP energies by the same amount.

This is a rather crude approximation and not recommended.

It might again be useful for benchmarking purposes.

CHAPTER

SIX

ANALYSIS

6.1 Density of States (DOS)

```
DOS
   CalcDOS Yes/No
   CalcPDOS Yes/No
   CalcPopulationAnalysis Yes/No
   CompensateDeltaE Yes/No
   DeltaE float
   Energies integer
   File string
   IntegrateDeltaE Yes/No
   Max float
   Min float
   StoreCoopPerBasPair Yes/No
End
```

DOS

Type
Block

Description
Density-Of-States (DOS) options

CalcDOS

Type
Bool

Default value
Yes

GUI name
Calculate DOS

Description
Whether or not to calculate the density of states.

CalcPDOS

Type
Bool

Default value
No

GUI name
Calculate PDOS

Description
Whether or not to calculate the partial DOS (projections on basis functions). This can be significantly more expensive than calculating the total DOS

`CalcPopulationAnalysis`

Type
Bool

Default value
Yes

GUI name
Calculate Mulliken charges

Description
Whether or not to calculate the population analysis. Population analysis can become very expensive when there are many symmetry operators, such as in a super cell.

`CompensateDeltaE`

Type
Bool

Default value
Yes

Description
Only relevant when IntegrateDeltaE=yes. If set to true then after integrating each interval over DeltaE the result is divided by DeltaE, so that the unit is DOS.

`DeltaE`

Type
Float

Default value
0.005

Unit
Hartree

GUI name
Delta E

Description
Energy step for the DOS grid. Using a smaller value (e.g. half the default value) will result in a finer sampling of the DOS.

`Energies`

Type
Integer

Description
Number of equidistant energy-values for the DOS grid. This keyword is superseded by the 'DeltaE' keyword.

`File`

Type
String

Description
Write the DOS (plain text format) to the specified file instead of writing it to the standard output.

`IntegrateDeltaE`

Type
Bool

Default value
Yes

Description
This subkey handles which algorithm is used to calculate the data-points in the plotted DOS. If 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)]. If 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.

`Max`

Type
Float

Unit
Hartree

Description
User defined upper bound energy (with respect to the Fermi energy)

`Min`

Type
Float

Unit
Hartree

Description
User defined lower bound energy (with respect to the Fermi energy)

`StoreCoopPerBasPair`

Type
Bool

Default value
No

GUI name
Calculate COOP

Description
Calculate the COOP (crystal orbital overlap population).

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 146) and *OverlapPopulations* (page 147)).

A common problem is that of missing DOS: an energy interval with bands but no DOS. This is caused by an insufficient k-space sampling. Try to *Restart the DOS* (page 189) with a better k-grid.

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)).

Tip: The tutorial Calculation of Band Structure and COOP of CsPbBr3 with BAND contains some advanced usage of the **DOS** and **BAND Structure GUI modules**.

6.1.1 Gross populations

```
GrossPopulations # Non-standard block. See details.
   ...
End
```

GrossPopulations

Type
Non-standard block

Description
Partial DOS (pDOS) are generated for the gross populations listed under this key. See example.

Syntax:

```
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
```

6.1.2 Overlap populations

```
OverlapPopulations # Non-standard block. See details.
   ...
End
```

OverlapPopulations

Type
Non-standard block

Description
Overlap population weighted DOS (OPWDOS), also known as the crystal orbital overlap population (COOP).

Overlap population weighted DOS 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 146)), 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
```

(continues on next page)

(continued from previous page)

```
      FragFun 1 1
   RIGHT
      2 1
      FragFun 3 5
End
```

6.2 Band Structure

BAND can calculate the band structure for the standard k-path in the Brillouin zone[1] and saves the corresponding data to the binary file RUNKF.

The band structure is best examined with the GUI module **BandStructure** see:

- Advanced BAND tutorial: Calculation of Band Structure and COOP of CsPbBr3 with BAND

```
BandStructure
   Enabled Yes/No
   Automatic Yes/No
   DeltaK float
   FatBands Yes/No
   UseSymmetry Yes/No
   EnergyAboveFermi float
   EnergyBelowFermi float
End
```

`BandStructure`

Type
Block

Description
Options for the calculation of the band structure.

`Enabled`

Type
Bool

Default value
No

GUI name
Calculate band structure

Description
If True, Band will calculate the band structure and save it to file for visualization.

`Automatic`

Type
Bool

Default value
Yes

[1] W. Setyawan and S. Curtarolo, *High-throughput electronic band structure calculations: Challenges and tools*, Computational Materials Science 49 (2010) 299–312 (https://doi.org/10.1016/j.commatsci.2010.05.010).

GUI name
Automatic generate path

Description
If True, BAND will automatically generate the standard path through the Brillouin zone.

If False BAND will use the user-defined path in BZPath.

`DeltaK`

Type
Float

Default value
0.1

Unit
1/Bohr

GUI name
Interpolation delta-K

Description
Step (in reciprocal space) 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.

`FatBands`

Type
Bool

Default value
Yes

GUI name
Calculate fatbands

Description
If True, BAND will compute the fat bands (only if BandStructure%Enabled is True).

The Fat Bands are the periodic equivalent of the Mulliken population analysis.

`UseSymmetry`

Type
Bool

Default value
Yes

GUI name
Use symmetry

Description
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 key does not influence the symmetry of the band structure sampling. Only available for Setyawan and Curtarolo convention (see `KPathFinderConvention`).

`EnergyAboveFermi`

Type
Float

Default value
0.75

Unit
Hartree

GUI name
Energy above Fermi level

Description
Bands with minimum energy larger then FermiEnergy + EnergyAboveFermi are not saved to file. Increasing the value of EnergyAboveFermi will result in more unoccupied bands to be saved to file for visualization.

`EnergyBelowFermi`

Type
Float

Default value
10.0

Unit
Hartree

GUI name
Energy below Fermi level

Description
Bands with maximum energy smaller then FermiEnergy - EnergyBelowFermi are not saved to file. Increasing the value of EnergyBelowFermi will result in more occupied core bands to be saved to file for visualization. Note: EnergyBelowFermi should be a positive number!

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.

6.2.1 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.

```
BZPath
   path # Non-standard block. See details.
      ...
   End
End
```

`BZPath`

Type
Block

Description
Definition of the user-defined path in the Brillouin zone for band structure plotting.

`path`

Type
Non-standard block

Recurring
True

Description
Definition of the k-points in a path. The vertices of your path should be defined in fractional coordinates (wrt the reciprocal lattice vectors)

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
   End
   Path
      0.625   0.250   0.625
      0.500   0.000   0.500
   End
End
```

6.2.2 Definition of the Fat Bands

The *fat bands* (page 148) $F_{i,n,\sigma,\vec{k}}$ 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.

6.2.3 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
```

6.2.4 Calculation of the Fermi Surface

If the system has no band gap it is a metal, and that means that the Fermi surface is a complex shape in the Brillouin zone. As multiple bands may be crossing the fermi energy there may be multiple surfaces. In case of a spin unrestricted calculation both spins have a Fermi surface of their own. For 2D systems the "surface" consists of one or more lines. The result can be viewed with the GUI module AMSbands.

- Advanced BAND tutorial: Bands, dos and Fermi surface with BAND

```
FermiSurface
   Enabled Yes/No
   KIntegForSymmetricKGrid integer
   NMesh integer
End
```

FermiSurface

Type
Block

Description
Calculation of the Fermi surface for metals

Enabled

Type
Bool

Default value
No

GUI name
Calculate Fermi surface

Description
Calculate the Fermi surface if the system has no band gap (i.e. is a metal). The result can be visualized with amsbands.

KIntegForSymmetricKGrid

Type
Integer

Default value
-1

Description
If the (default) regular k-grid is used, a symmetric one is created to determine the Fermi surface. If this key is not specified an automatic value of kInteg is used. Odd values trigger quadratic interpolation.

NMesh

Type
Integer

Default value
7

Description
Improves the matching of the interpolated quadratic surface. For better results it makes more sense to increases KIntegForSymmetricKGrid.

6.3 Charges

6.3.1 Default Atomic Charge Analysis

By default BAND computes the following atomic charge analyses:

- **Hirshfeld Charges**[1][2]
- **Voronoi Deformation Charges** (VDD, Voronoi Deformation Density)
- **Mulliken Charges** (note: not calculated for *Spin-Orbit* (page 33) calculations)
- **CM5** (Charge Model 5)[3][4]

These atomic charges are printed to the output file and can be visualized using the AMSview GUI module.

A more detailed output of the atomic charges can be printed by specifying following print option (note: in Band 2017 and previous versions this detailed output was printed by default):

```
Print AtomicChargesDetails
```

6.3.2 Bader Analysis (AIM)

The QTAIM (Quantum Theory of Atoms in Molecules), also known as Bader Analysis can be enabled in the GridBasedAIM input block:

```
GridBasedAIM
   Enabled Yes/No
   Iterations integer
   SmallDensity float
   UseStartDensity Yes/No
End
```

GridBasedAIM

Type
Block

Description
Invoke the ultra fast grid based Bader analysis.

Enabled

Type
Bool

Default value
No

GUI name
Bader (AIM): Atomic properties

[1] F.L. Hirshfeld, *Bonded-atom fragments for describing molecular charge densities*, Theoretica Chimica Acta 44, 129 (1977) (https://doi.org/10.1007/BF00549096)

[2] K.B. Wiberg and P.R. Rablen, *Comparison of atomic charges derived via different procedures*, Journal of Computational Chemistry 14, 1504 (1993) (https://doi.org/10.1002/jcc.540141213)

[3] A.V. Marenich, S.V. Jerome, C.J. Cramer, D.G. Truhlar, *Charge Model 5: An Extension of Hirshfeld Population Analysis for the Accurate Description of Molecular Interactions in Gaseous and Condensed Phases*, Journal of Chemical Theory and Computation 8, 527 (2012) (https://doi.org/10.1021/ct200866d)

[4] C.A. Peeples and G. Schreckenbach, *Implementation of the SM12 Solvation Model into ADF and Comparison with COSMO*, Journal of Chemical Theory and Computation 12, 4033 (2016) (https://doi.org/10.1021/acs.jctc.6b00410)

Description
Invoke the ultra fast grid based Bader analysis.

`Iterations`

Type
Integer

Default value
40

Description
The maximum number of steps that may be taken to find the nuclear attractor for a grid point.

`SmallDensity`

Type
Float

Default value
1e-06

Description
Value below which the density is ignored. This should not be chosen too small because it may lead to unassignable grid points.

`UseStartDensity`

Type
Bool

Default value
No

Description
Whether the analysis is performed on the startup density (True) or on the final density (False).

```
AIMCriticalPoints
   Enabled Yes/No
   EqvPointsTol float
   GridPadding float
   GridSpacing float
End
```

`AIMCriticalPoints`

Type
Block

Description
Compute the critical points of the density (Atoms In Molecules). The algorithm starts from a regular mesh of points, and from each of these it walks towards its corresponding critical point.

`Enabled`

Type
Bool

Default value
No

GUI name
: Critical points and bond paths

Description
Compute the critical points of the density (Atoms In Molecules). The algorithm starts from a regular mesh of points, and from each of these it walks towards its corresponding critical point.

`EqvPointsTol`

Type
Float

Default value
0.27

Unit
Bohr

Description
If the distance between two critical points is smaller than this value, the two critical points are considered to be the same point.

`GridPadding`

Type
Float

Default value
0.7

Unit
Bohr

Description
How much extra space is added to the starting guess domain in the search for the critical points

`GridSpacing`

Type
Float

Default value
0.5

Unit
Bohr

Description
The distance between the initial trial points.

Note: The Bader (AIM) analysis is performed on the fitted density (see *ZlmFit* (page 75)). We advise to use a Good (or better) ZlmFit quality.

6.4 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 ("rkf"), to something like "frag.rkf", see the example script discussed below or the *example* (page 292) covering this topic.

```
Fragment
   AtomMapping # Non-standard block. See details.
      ...
   End
   Filename string
   Labels # Non-standard block. See details.
      ...
   End
End
```

`Fragment`

Type
Block

Recurring
True

Description
Defines a fragment. You can define several fragments for a calculation.

`AtomMapping`

Type
Non-standard block

Description
Format '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.

`Filename`

Type
String

Description
Filename of the fragment. Absolute path or path relative to the executing directory.

`Labels`

Type
Non-standard block

Description
This gives the possibility to introduce labels for the fragment orbitals. See examples.

Example:

```
Fragment
  filename test.rkf
  AtomMapping
    1 3 !  atom 1 of this fragment is assigned to third atom
    2 4 !  atom 2 of this fragment is assigned to fourth atom
  End
  Labels
    Sigma
    Sigma*
    Pi_x
    Pi_y
    Pi_x*
    Pi_y*
  End
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.

6.5 Energy Decomposition Analysis

In BAND there are two fragment-based energy decomposition methods available: the periodic energy decomposition analysis (PEDA)[1] and the periodic energy decomposition analysis combined with the natural orbitals of chemical valency method (PEDA-NOCV)[Page 157, 1].

6.5.1 Periodic Energy Decomposition Analysis (PEDA)

```
PEDA Yes/No
```

PEDA

Type
Bool

Default value
No

Description
If present in combination with the fragment block, the decomposition of the interaction energy between fragments is invoked.

[1] 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) (https://doi.org/10.1063/1.4919943).

If used 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 306) 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.

6.5.2 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 186).

See also:

- *example* (page 310)
- tutorial
- advanced tutorial

```
PEDANOCV
   EigvalThresh float
   Enabled Yes/No
End
```

PEDANOCV

Type
Block

Description
Options for the decomposition of the orbital relaxation (pEDA).

EigvalThresh

Type
Float

Default value
0.001

GUI name
Use NOCVs with ev larger than

Description
The threshold controls that for all NOCV deformation densities with NOCV eigenvalues larger than EigvalThresh the energy contribution will be calculated and the respective pEDA-NOCV results will be printed in the output

Enabled

Type
Bool

Default value
No

GUI name
Perform PEDA-NOCV analysis

Description
If true in combination with the fragment blocks and the pEDA key, the decomposition of the orbital relaxation term is performed.

6.6 Local Density of States (STM)

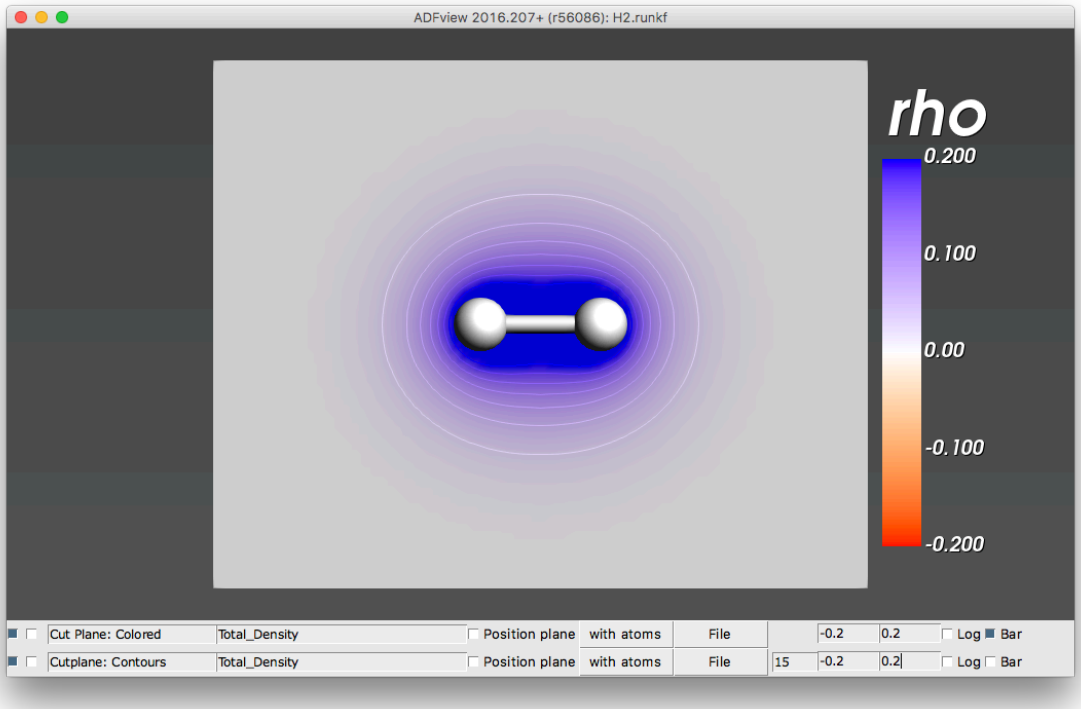
In the **VIEW GUI module**, you can visualize the local density of states, see also the *LDOS (STM)* (page 187) key.

6.7 3D field visualization with BAND

With AMSview 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).



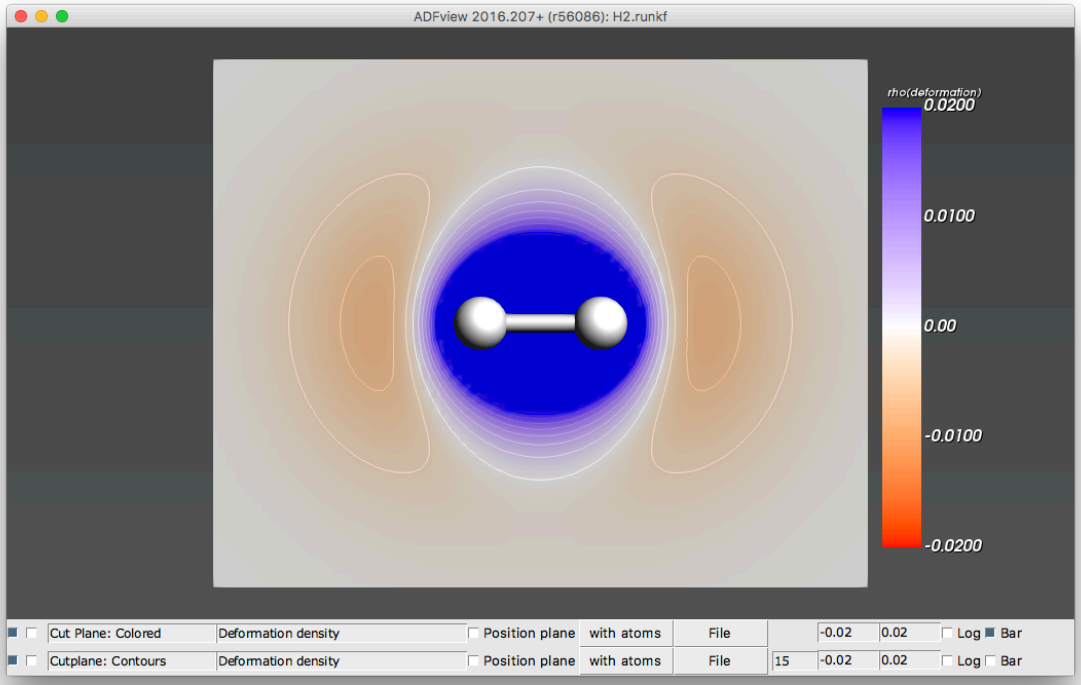
Deformation density (rho(deformation))

The deformation density is the difference between total density $\rho(r)$ and reference density $\rho_{\text{reference}}(r)$

$$\rho_{\text{deformation}}(r) = \rho(r) - \rho_{\text{reference}}(r)$$

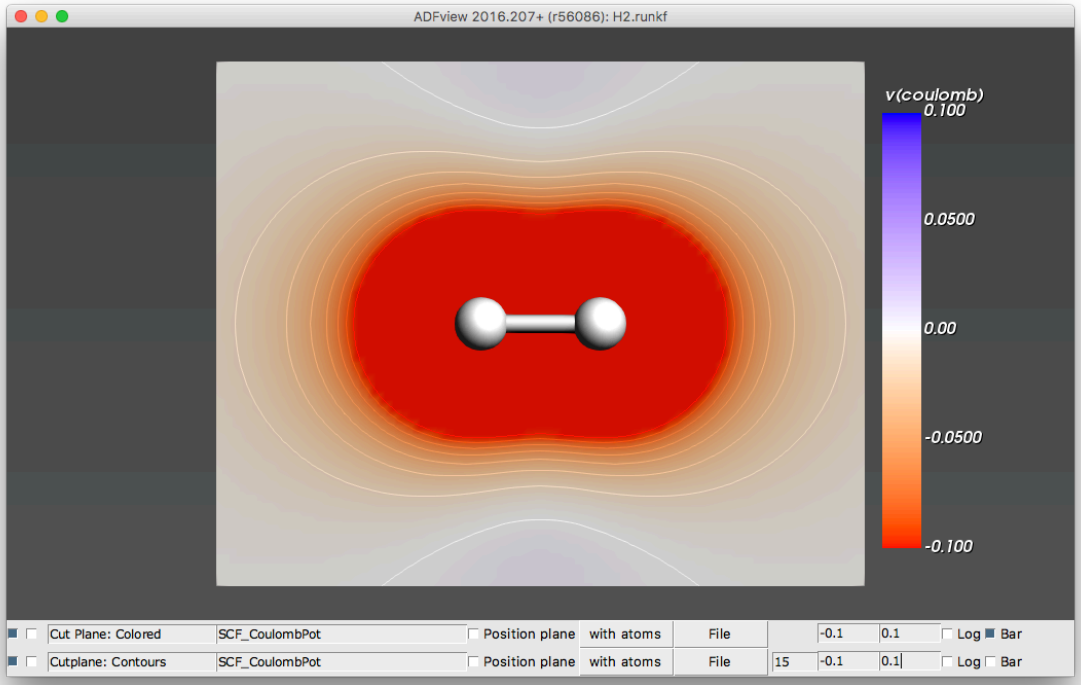
The reference density $\rho_{\text{reference}}(r)$ is defined as the sum of densities of spherical spin-unrestricted isolated atoms.

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.



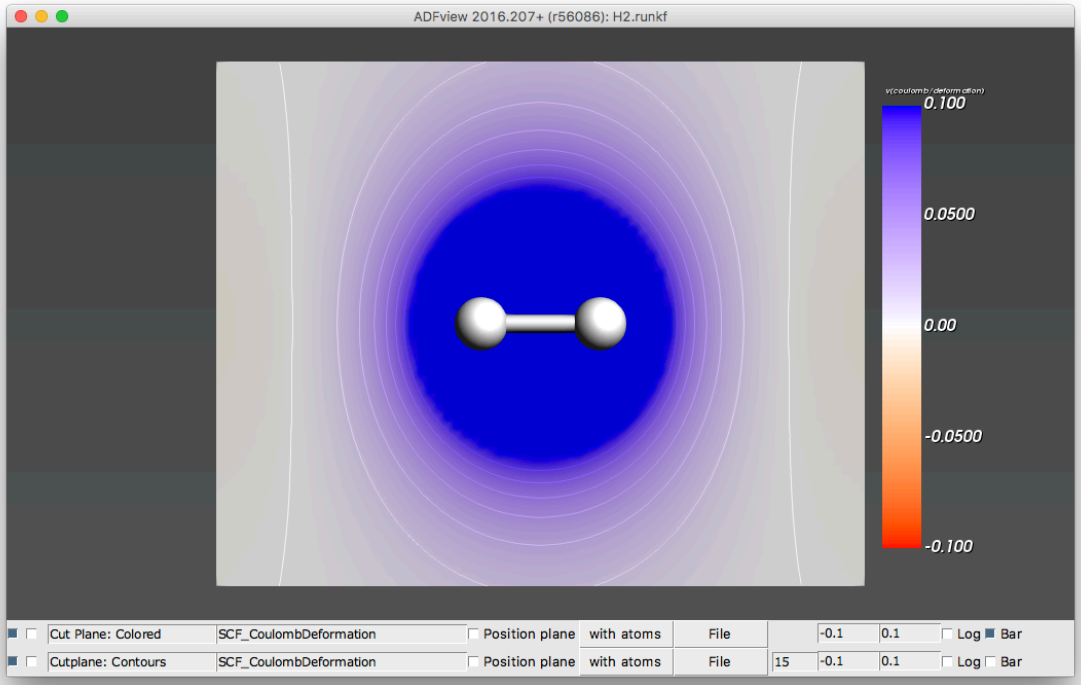
SCF_CoulombPot (v(coulomb))

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.



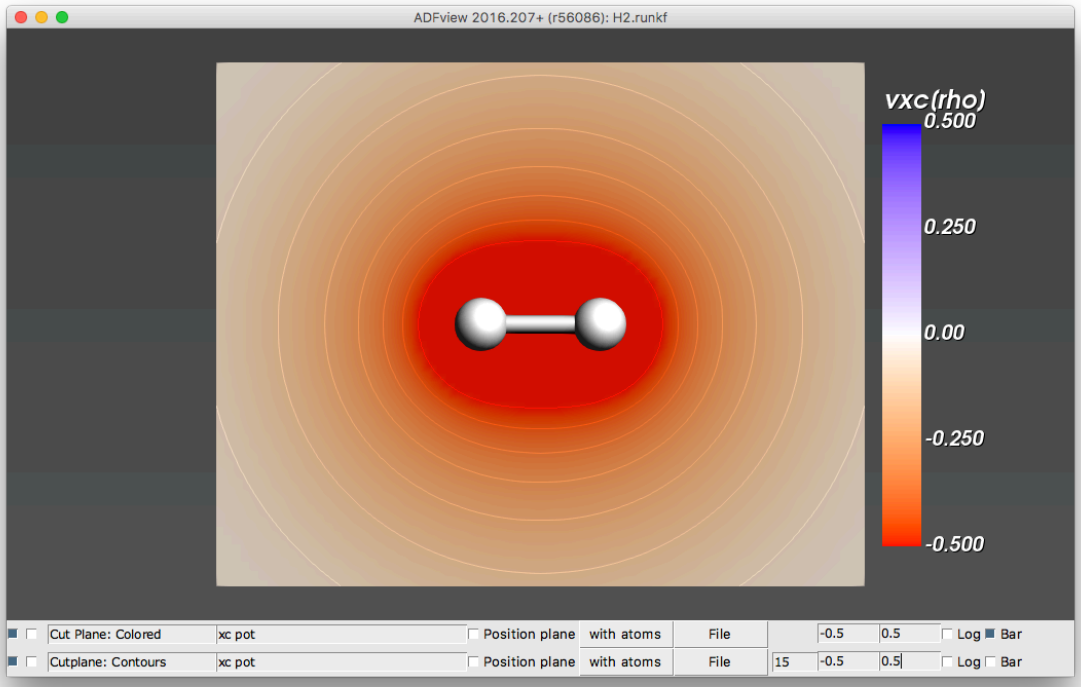
Note: The sign convention for potentials in **BAND** is the opposite to the **ADF** sign convention.

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)

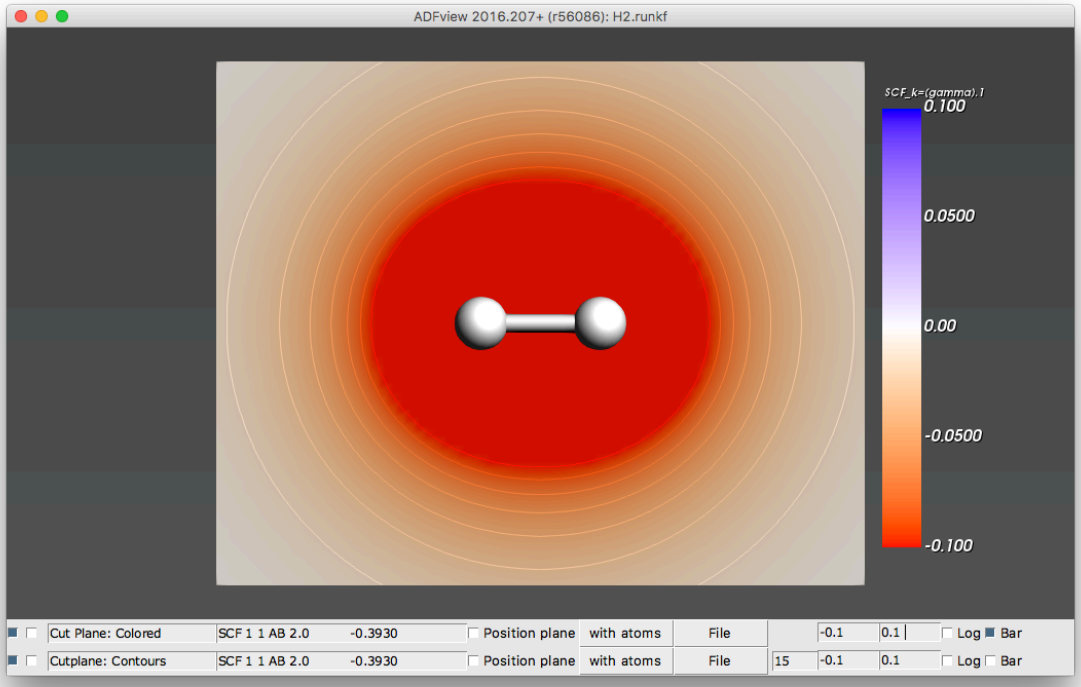


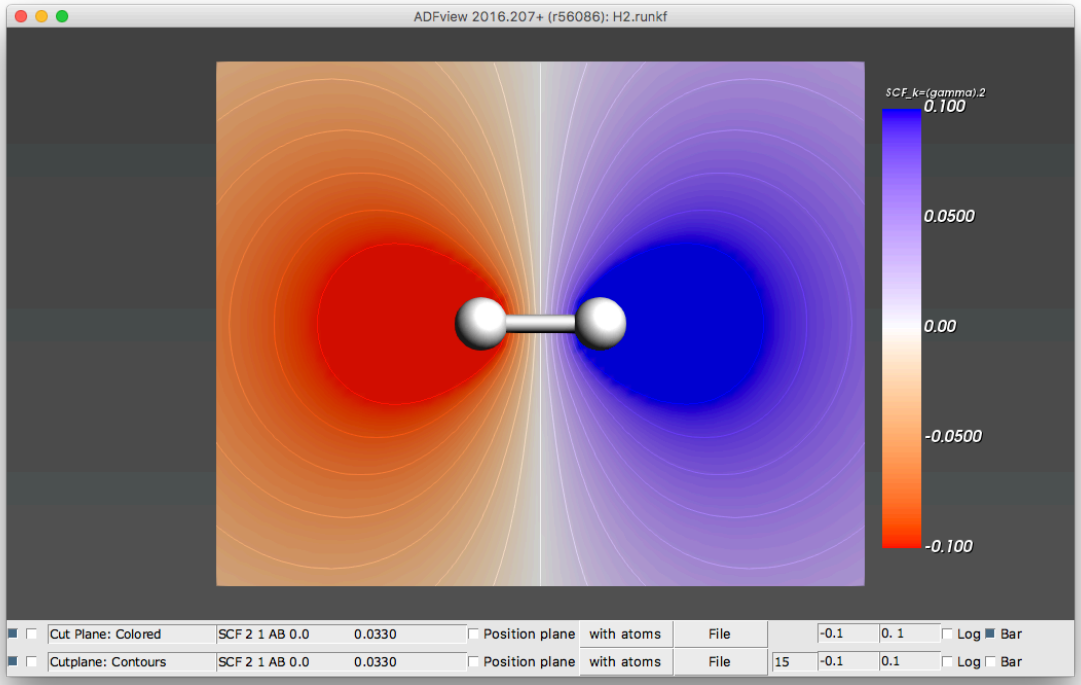
Orbitals (occupied/virtual)
The Kohn-Sham orbitals.

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.




ADFview 2016.207+ (r56086): H2.runkf
SCF_k=(gamma),2
0.100
0.0500
0.00
-0.0500
-0.100
Cut Plane: Colored
SCF 2 1 AB 0.0
0.0330
Position plane
with atoms
File
-0.1
0. 1
Log
Bar
Cutplane: Contours
SCF 2 1 AB 0.0
0.0330
Position plane
with atoms
File
15
-0.1
0.1
Log
Bar

CHAPTER

SEVEN

ELECTRONIC TRANSPORT (NEGF)

See also:

BAND-NEGF GUI tutorial

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

Example: Main NEGF flavors (page 253)
Example: NEGF with bias (page 260)

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

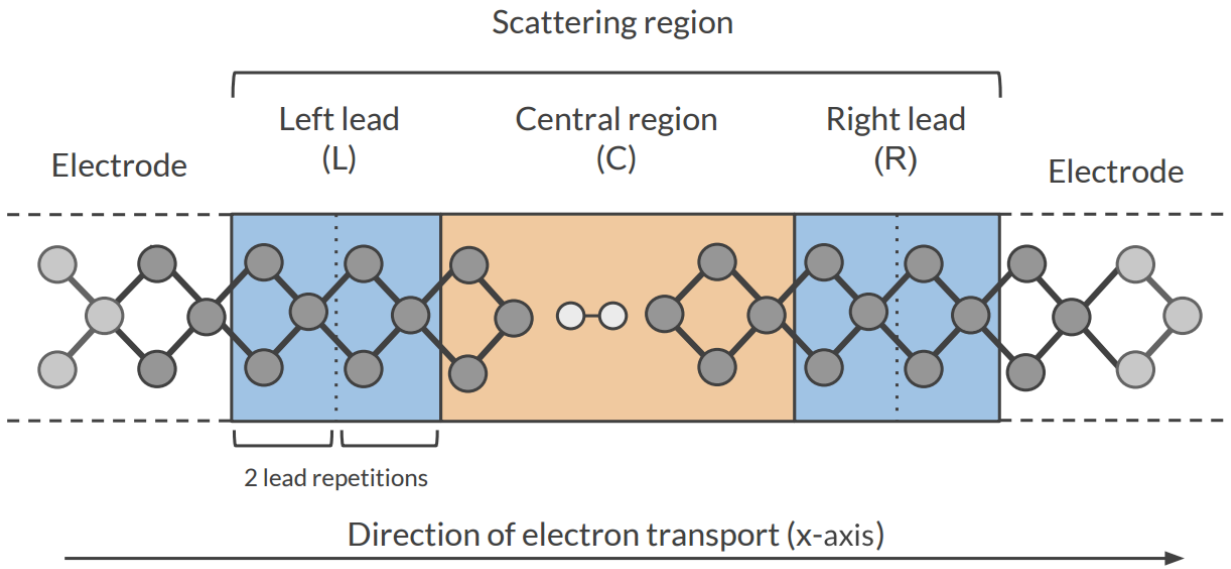
Self consistent
This is the internal BAND-NEGF implementation, which is described in this page.

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).

7.1 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:



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

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

7.1.1 Self consistency

The density matrix is determined self consistently[1]:

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

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.

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

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

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

7.1.2 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.

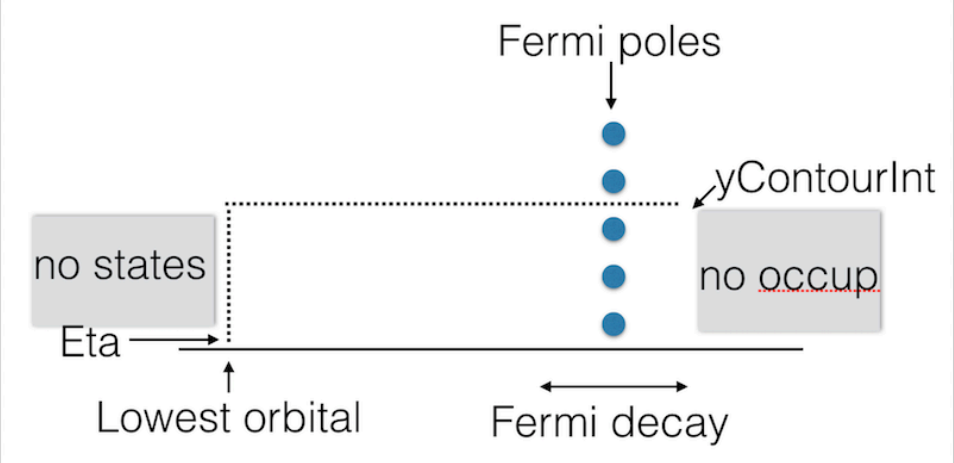


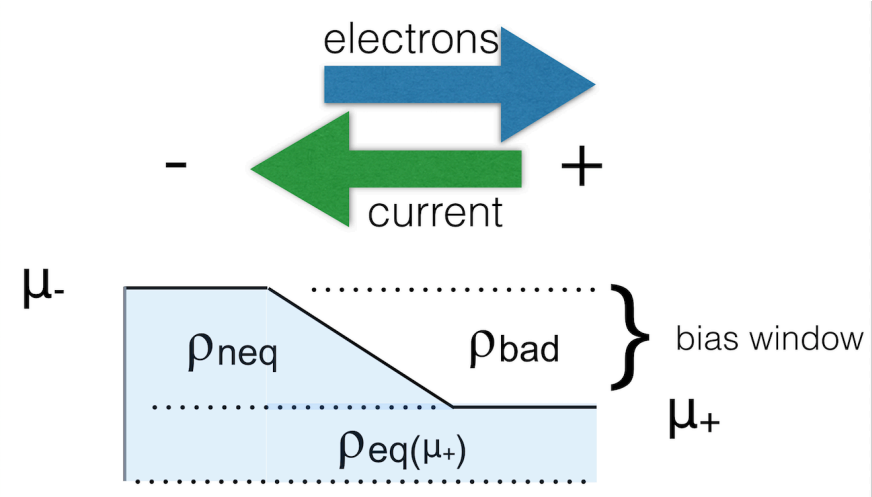
Fig. 7.1: Figure: BAND uses a rectangular contour in the complex energy plane to integrate the (integrand of the) density matrix. The integrand also needs to be evaluated in the enclosed FD poles (three in this picture).

[1] 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 (https://doi.org/10.1021/jp3044225).

7.1.3 Gate potential

There is no direct key for the gate potential. You can model this with the *FuzzyPotential* (page 51) key. Setting up the gate potential for NEGF is most conveniently done with the GUI.

7.1.4 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 51) key. The GUI can be helpful here.

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

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

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:

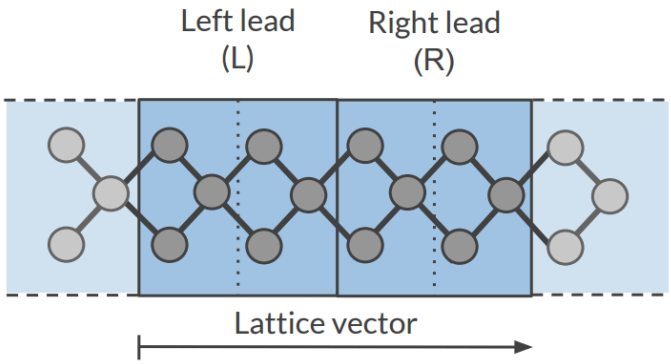
PhD Thesis (https://www.scm.com/wp-content/uploads/Verzijl2012.pdf) of C. Verzijl (BAND-NEGF developer)

7.2 Workflow

The computation of the transmission function $T(E)$ within the BAND-NEGF[Page 167, 1] 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 177)):



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 62) 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.

7.3 Input options

7.3.1 SGF Input options

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

```
$AMSBIN/sgf   << eor
TITLE Test for NEGF inputs
SAVE SIGMA
SURFACEGF
   SCMCode True
   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.

7.3.2 NEGF Input options (no bias)

The NEGF functionality is controlled by the NEGF block key.

```
NEGF
   LeadFile string
   SGFFile string
   ContourQuality [basic | normal | good | verygood]
   EMin float
   EMax float
   NE integer
End
```

NEGF

Type
Block

Description
Options for the NEGF (non-equilibrium green function) transport calculation.

LeadFile

Type
String

Default value
Description
File containing the tight binding representation of the lead.

SGFFile

Type
String

Default value
Description
The result from the SGF program. Contains the Fermi energy of the lead.

ContourQuality

Type
Multiple Choice

Default value
good

Options
[basic, normal, good, verygood]

Description
The density matrix is calculated numerically via a contour integral. Changing the quality influences the number of points. This influences a lot the performance.

EMin

Type
Float

Default value
-5.0

Unit
eV

Description
The minimum energy for the transmission grid (with respect to the Fermi level of the lead)

EMax

Type
Float

Default value
5.0

Unit
eV

Description
The maximum energy for the transmission grid (with respect to the Fermi level of the lead)

NE

Type
Integer

Default value
100

Description
The number of energies for the transmission energy grid.

The following are expert / technical options:

```
NEGF
   CheckOverlapTol float
   Eta float
   ApplyShift1 Yes/No
   ApplyShift2 Yes/No
   YContourInt float
   DEContourInt float
End
```

NEGF

Type
Block

Description
Options for the NEGF (non-equilibrium green function) transport calculation.

CheckOverlapTol

Type
Float

Default value
0.01

Description
BAND checks how well the TB overlap matrix 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.

Eta

Type
Float

Default value
1e-05

Description
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.

ApplyShift1

Type
Bool

Default value
Yes

Description
Apply the main shift, obtained from comparing matrix elements in the leads with those from the tight-binding run. Strongly recommended.

ApplyShift2

Type
Bool

Default value
Yes

Description
Apply the smaller alignment shift. This requires an extra alignment run. Usually this shift is smaller.

YContourInt

Type
Float

Default value
0.3

Description
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

Type
Float

Default value
-1.0

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

7.3.3 NEGF Input options (with bias)

With a bias potential there are some extra keys.

```
NEGF
   BiasPotential float
   NonEqDensityMethod integer
   BoundOccupationMethod integer
   YRealaxisInt float
   DERealAxisInt float
End
```

NEGF

Type
Block

Description
Options for the NEGF (non-equilibrium green function) transport calculation.

BiasPotential

Type
Float

Default value
0.0

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

NonEqDensityMethod

Type
Integer

Default value
1

Description
See text.

BoundOccupationMethod

Type
Integer

Default value
1

Description
See text. Only relevant with NonEqDensityMethod equal 2 or 3.

YRealaxisInt

Type
Float

Default value
1e-05

Description
The non-Equilibrium density is calculated near the real axis.

DERealAxisInt

Type
Float

Default value
-1.0

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

NonEqDensityMethod
Let us introduce some terms[2]. 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 fromPage 174, 2).

BoundOccupationMethod
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[2]. 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.

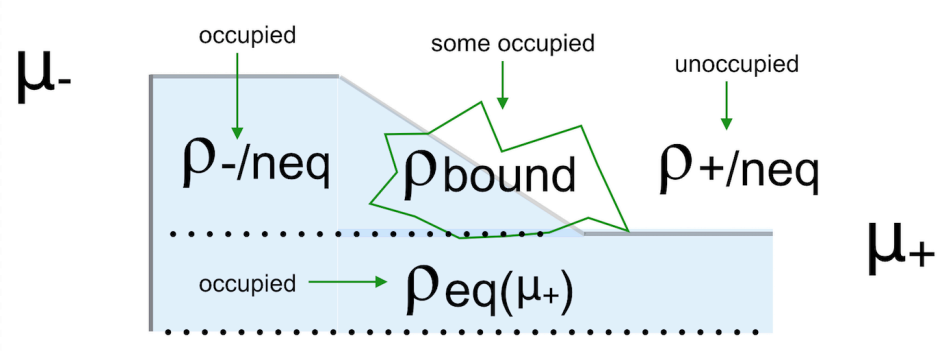


Fig. 7.2: Figure: The non-equilibrium density consists of three parts: the left and right parts (ρ-/neq and ρ+/neq) and the bound states (ρbound). We want to know the occupied part.

Setting the method BoundOccupationMethod to 1, leads to

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

[2] 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 (https://doi.org/10.1016/j.chemphys.2007.06.011).

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[2]

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

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

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

```
$AMSBIN/amsreport RUNKF 'NEGF%current'
```

7.3.4 NEGF Input options (alignment)

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

```
NEGF
   DoAlignment Yes/No
   Alpha float
   AlignChargeTol float
   CDIIS Yes/No
End
```

NEGF

Type
Block

Description
Options for the NEGF (non-equilibrium green function) transport calculation.

DoAlignment

Type
Bool

Default value
No

Description
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

Type
Float

Default value
1e-05

Description
A charge error needs to be translated in a potential shift. DeltaV = alpha * DeltaQ

AlignChargeTol

Type
Float

Default value
0.1

Description
In an alignment run you want to get the number of electrons in the center right. This number specifies the criterion for that.

CDIIS

Type
Bool

Default value
No

Description
Make the normal DIIS procedure aware of the align charge error

7.4 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 189) DensityMatrix" and the *Restart* (page 179) 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.

7.5 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.

7.5.1 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 Yes/No
```

StoreHamiltonian2

Type
Bool

Default value
No

Description
determine the tight-binding representation of the overlap an fock matrix. Used for (at least) NEGF.

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).

CHAPTER

EIGHT

EXPERT OPTIONS

8.1 Restarts

The main results of a BAND calculation are stored in the rkf 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 143) or the *band structure* (page 148)). Similarly, a geometry optimization can be restarted with the subkey `GeometryOptimization` You can use the geometry of a previous calculation.

Usually the input for a restarted job is the same as for the original calculation, with some extra options, described in this section.

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

Example: Restart SCF for properties calculation (page 245)
Example: Restart the SCF (page 239)
Example: Properties on a grid (page 246)
Example: DOS and BandStructure from a previous calculation (page 249)

8.1.1 Restart key

```
Restart
   File string
   SCF Yes/No
   DensityPlot Yes/No
   OrbitalPlot Yes/No
   NOCVdRhoPlot Yes/No
   NOCVOrbitalPlot Yes/No
   UseDensityMatrix Yes/No
   BandStructure Yes/No
   DOS Yes/No
End
```

Restart

Type
Block

Description
Tells the program that it should restart with the restart file, and what to restart.

`File`

Type
String

Default value
GUI name
Restart using

Description
Name of the restart file. The file should be a band.rkf file from a previous run.

`SCF`

Type
Bool

Default value
No

GUI name
Restart: SCF

Description
Continue the SCF procedure using the orbital coefficients and occupations from the restart file.

`DensityPlot`

Type
Bool

Default value
No

Description
Goes together with the DensityPlot block and Grid blocks

`OrbitalPlot`

Type
Bool

Default value
No

Description
Goes together with the OrbitalPlot and Grid

`NOCVdRhoPlot`

Type
Bool

Default value
No

Description
Goes together with the NOCVdRhoPlot and Grid blocks.

`NOCVOrbitalPlot`

Type
Bool

Default value
No

Description
Goes together with the NOCVOrbitalPlot and Grid blocks.

`UseDensityMatrix`

Type
Bool

Default value
No

Description
If set to True: For restarting the SCF the density matrix will be used. Requires you to set 'Save DensityMatrix' in the previous run.

`BandStructure`

Type
Bool

Default value
No

Description
Calculate the band structure from a previous calculation. Does not work with model potentials and Hubbard.

`DOS`

Type
Bool

Default value
No

Description
Calculate the DOS from a previous calculation. Does not work with model potentials and Hubbard.

8.1.2 Grid

The Grid block is 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 automatically generates a grid around the atoms in the unit cell:

```
Grid
   Type [coarse | medium | fine]
End
```

`Grid`

Type
Block

Description
Options for the regular grid used for plotting (e.g. density plot). Used ICW the restart option.

`Type`

Type
Multiple Choice

Default value
coarse

Options
[coarse, medium, fine]

Description
The default regular grids.

One alternative is to specify everything by hand via the 'UserDefined' sub-block.

```
Grid
   UserDefined header # Non-standard block. See details.
      ...
   End
End
```

`Grid`

Type
Block

Description
Options for the regular grid used for plotting (e.g. density plot). Used ICW the restart option.

`UserDefined`

Type
Non-standard block

Description
One can define the regular grid specification in this block. See example. Default unit is Bohr

The following input would create a cube from (-1,-1,-1) to (1,1,1) bohr:

```
Grid
  UserDefined
     -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
End
```

Note: The grid is specified in bohr

One can also specify a text file from which the grid is imported:

```
Grid
   FileName string
End
```

Grid

Type
Block

Description
Options for the regular grid used for plotting (e.g. density plot). Used ICW the restart option.

FileName

Type
String

Default value

Description
Read in the grid from a file. The file format of the grid is: three numbers per line (defining the x, y and z coordinates of the points).

8.1.3 Plots of the density, potential, and many more properties

```
DensityPlot # Non-standard block. See details.
   ...
End
```

DensityPlot

Type
Non-standard block

Description
Plots of the density. Goes together with the Restart%DensityPlot and Grid keys.

The DensityPlot block goes together with the `Restart%DensityPlot` and `Grid` keys. Example input:

```
...
Restart
   File my_file.rkf
   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 AMSview.

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 187))
- `elf[rho]` The electron localization function
- `X` The *Electron energy density function* (page 188) from Ref[1][2] . Equivalently `X(fit)` may be used as an approximation, employing the density fit.

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 292) example which shows how this can be used in combination with the `Fragment` key.

8.1.4 Orbital plots

```
OrbitalPlot # Non-standard block. See details.
   ...
End
```

`OrbitalPlot`

Type
Non-standard block

Description
Goes together with the Restart%OrbitalPlot and Grid keys. See Example.

The OrbitalPlot block goes together with the `Restart%OrbitalPlot` and `Grid` keys. Example input:

```
...
Restart
   File my_file.rkf
   OrbitalPlot
End

Grid
   Type Coarse
End

OrbitalPlot
```

(continues on next page)

[1] Stefano Racioppi, Martin Rahm. In-Situ Electronegativity and the Bridging of Chemical Bonding Concepts. Chemistry – A European Journal 72 (2021): 18156-18167 (https://doi.org/10.1002/chem.202103477)

[2] Stefano Racioppi, Per Hyldgaar, Martin Rahm. Quantifying Atomic Volume, Partial Charge, and Electronegativity in Condensed Phases. The Journal of Physical Chemistry C 128.9 (2024): 4009 (https://doi.org/10.1021/acs.jpcc.3c07677)

(continued from previous page)

```
   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 AMSview.

8.1.5 Induced Density Plots of Response Calculations

```
ResponseInducedDensityPlot # Non-standard block. See details.
   ...
End
```

`ResponseInducedDensityPlot`

Type
Non-standard block

Description
Goes together with Restart%ResponseInducedDensityPlot and Grid.

`ResponseInducedDensityPlot (block-type)` The ResponseInducedDensityPlot block goes together with the `Restart%ResponseInducedDensityPlot` and `Grid` keys. In the BAND example directory there is the *TD-CDFT for MoS2 Monolayer* (page 278) example that shows how this can be used. Example input:

```
...
Restart
   File my_file.rkf
   ResponseInducedDensityPlot
End

Grid
   Type Coarse
End

ResponseInducedDensityPlot
   XCOMPONENT 5 8  ! plot x component of induced densities
                   ! for frequencies number 5 to 8
   YCOMPONENT 6    ! plot y component of induced densities
                   ! for frequency number 6
   ZCOMPONENT 1    ! plot z component of induced densities
                   ! for frequency number 1
End
...
```

After such a run you get a TAPE41 file that you should rename to my.t41, and view with AMSview.

Attention: The plotting capability works only with response calculation RUNKF files based on the *NewResponse* (page 115) method!

8.1.6 NOCV Orbital Plots

```
NOCVOrbitalPlot # Non-standard block. See details.
   ...
End
```

NOCVOrbitalPlot

Type
Non-standard block

Description
Goes together with the Restart%NOCVOrbitalPlot and Grid keys. See example.

The NOCVOrbitalPlot blockg oes together with the `Restart%NOCVOrbitalPlot` and `Grid` keys. See example *PEDANOCV_MgO+CO* (page 310). Example input:

```
...
Restart
   File my_file.rkf
   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 AMSview.

8.1.7 NOCV Deformation Density Plots

```
NOCVdRhoPlot # Non-standard block. See details.
   ...
End
```

NOCVdRhoPlot

Type
Non-standard block

Description
Goes together with the Restart%NOCVdRhoPlot and Grid keys. See example.

The NOCVdRhoPlot blockg oes together with the `Restart%NOCVdRhoPlot` and `Grid` keys. See example *PEDANOCV_MgO+CO* (page 310). Example input:

```
...
Restart
   File my_file.rkf
   NOCVdRhoPlot
End
```

(continues on next page)

(continued from previous page)

```
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 AMSview.

8.1.8 LDOS (STM)

The local density of states (LDOS) represents a partial density, (see wikipedia (https://en.wikipedia.org/wiki/Density_of_states#Local_density_of_states)): it is the density arising from states within an energy window.

```
LDOS
   DeltaNeg float
   DeltaPos float
   Shift float
End
```

LDOS

Type
Block

Description
Local Density-Of-States information. This can be used to generate STM images in the Tersoff-Hamann approximation (see https://doi.org/10.1103/PhysRevB.31.805)

DeltaNeg

Type
Float

Default value
0.0001

Unit
Hartree

Description
Lower bound energy (Shift-DeltaNeg)

DeltaPos

Type
Float

Default value
0.0001

Unit
Hartree

Description
Upper bound energy (Shift+DeltaPos)

Shift

Type
Float

Default value
0.0

Unit
Hartree

Description
The energy bias with respect to the Fermi level.

Integrating from minus infinity (DeltaNeg=1e6) to the fermi level (DeltaPos=0) produces the total (valence) density.

The local density of states is integrated over the resulting interval. Example of an LDOS restart:

```
Restart
   File my_file.rkf
   DensityPlot
End

Grid
   Type Coarse
End

DensityPlot
   LDOS
End

LDOS
   Shift    0.1
   DeltaNeg 0.001
   DeltaPos 0.0
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 AMSview. (Rename the file to my.t41)

See also *Restart* (page 179), and *DensityPlot* (page 183).

8.1.9 Electron Energy Density

The electron energy density is defined as[Page 184, 1]

$X(r) = -\left\{\frac{1}{2}\sum_i^{\text{occ}} \nabla\psi_i \cdot \nabla\psi_i - \frac{1}{4}\nabla^2\rho - V_{\text{effective}}\rho\right\}$

It can be obtained by requesting `X` or `X(fit)` in a restart, see also *Restart* (page 179), and *DensityPlot* (page 183).

8.1.10 Save

```
Save string
```

Save

Type
String

Recurring
True

Description
Save scratch files or extra data that would be otherwise deleted at the end of the calculation. e.g. 'TAPE10' (containing the integration grid) or 'DensityMatrix'

8.1.11 Restarting the DOS and/or BandStructure

Perhaps you did a calculation, optimizing the geometry, and now want to see the band structure and partial DOS. This can be achieved by using `Restart%DOS` and `Restart%BandStructure`. This way you can easily refine your plots, or solve a missing DOS problem, without having to repeat the whole SCF.

With the restarting of the DOS there is the special possibility to use a better k-grid than was used during the SCF. Whereas the Band structure is not very k-grid sensitive, the DOS depends strongly on the k-grid. A common problem is that of missing DOS: where there are bands there is no DOS, being an artifact of insufficient k-sampling. Using only a better k-grid for the DOS calculation may produce a DOS that is almost as good as if a full calculation (SCF and DOS) was done with the better k-grid. Notice that the effective potential in such a restart with a better k-grid corresponds to the k-grid as was used during the SCF. Therefore the band structure is not affected by using a better k-grid for the restart. See also the *RestartDosAndBandStructure* (page 249) example.

This figure shows that the DOS obtained from restarting with a better k-grid is very close to the one obtained with a full calculation with the better k-grid.

Below we show how a missing DOS issue can be solved by either using a better k-grid, or more efficiently by only using a better k-grid for the DOS (using the `DOS%Restart` option)

While restarts for plotting should be done with the `Grid` key, the restarting of the DOS/BandStructure should not.

8.2 References

8.3 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.

Whether or not symmetry should be used can be controlled vial the `UseSymmetry` key

```
UseSymmetry Yes/No
```

UseSymmetry

Type
Bool

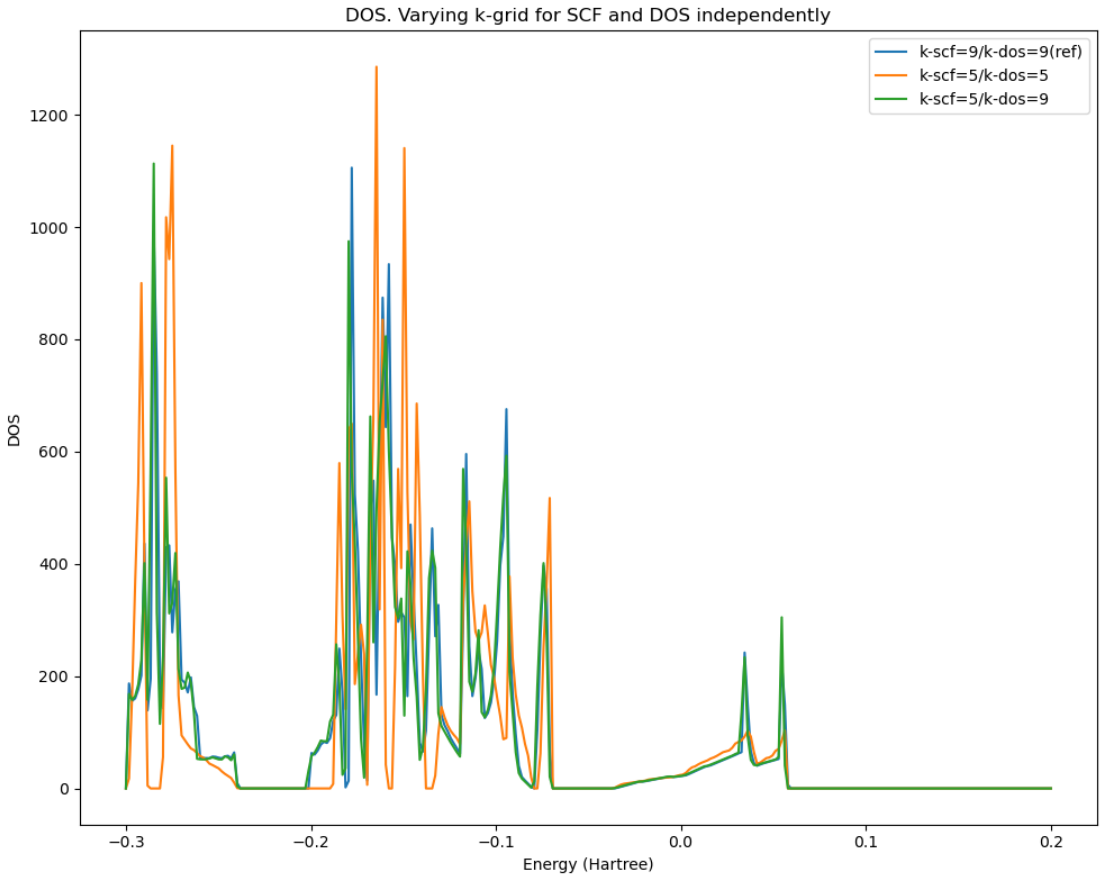


Fig. 8.1: DOS of a Mo3WSeS7 slab. The best result is when using a 9x9 k-grid for both the SCF and the DOS calculation (blue curve). Using a worse 5x5 grid for both the SCF and the DOS produces a quite different DOS (amber). Doing the SCF with the coarser 5x5 grid and restarting the DOS with the finer 9x9 grid gives the green DOS, matching closely, and mostly hides, the best DOS (blue).

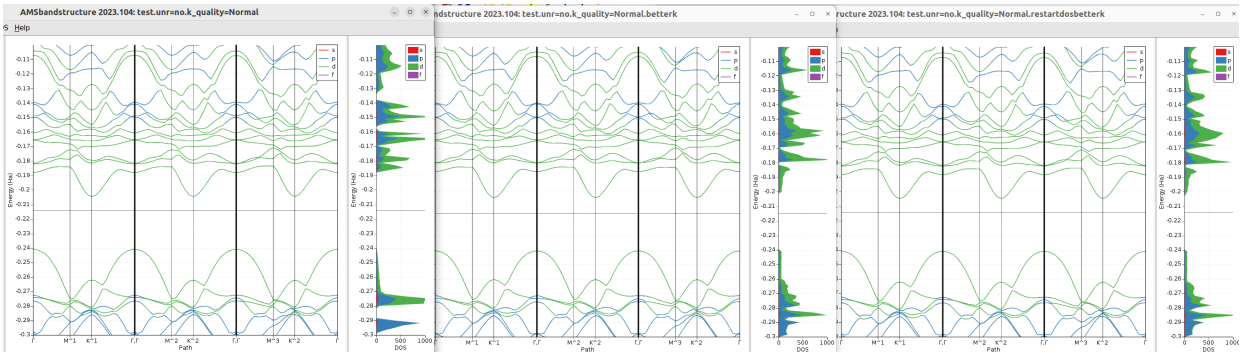


Fig. 8.2: Illustration of the missing DOS problem for a Mo3WSeS7 slab. Left panel: calculation with a normal k-grid. We clearly see DOS missing in the energy intervals -0.20 to -0.19, and -0.29 to -0.28. Middle panel: using a good k-grid, now the missing DOS appears. The band structure is not much affected, only the lowest band at Gamma being slightly higher with a good k-grid. Right panel: restart from the normal k-grid calculation, but using a good k-grid for the restart. The restarted DOS is very close to the one obtained with the good k-grid for both the SCF and the DOS.

Default value
Yes

Description
Whether or not to exploit symmetry during the calculation.

One can also select a sub set of symmetry operators:

```
SubSymmetry integer_list
```

SubSymmetry

Type
Integer List

Description
The indices of the symmetry operators to maintain.

To get the indices of the symmetry operators, you should first run the calculation with the following options 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:

```
SubSymmetry 1 7 21 31
```

8.3.1 Symmetry breaking for SCF

```
PotentialNoise float
```

PotentialNoise

Type
Float

Default value
0.0001

Description
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.

8.4 Advanced Occupation Options

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 alpha and beta electrons with the `Occupations` (Γ-only) and alternatively the `EnforcedSpinPolarization` (for an arbitrary number of k-points) key.

```
Occupations # Non-standard block. See details.
   ...
End
```

Occupations

Type
Non-standard block

Description
Allows one to input specific occupations numbers. Applies only for calculations that use only one k-point (i.e. pseudo-molecule calculations). See example.

Example:

```
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.

```
ElectronHole
   BandIndex integer
   SpinIndex integer
End
```

ElectronHole

Type
Block

Description
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.

`BandIndex`

Type
Integer

Description
Which occupied band shall be depopulated.

`SpinIndex`

Type
Integer

Description
Defines the spin of the shifted electron (1 or 2).

See the example *Si_ElectronHole* (page 322)

Finally, also the `Convergence%StartWithMaxSpin` influences (indirectly) the final state. Running for instance the H2 molecule with a PBE functional preserves the triplet state through the SCF (not being the ground state).

CHAPTER

NINE

TROUBLESHOOTING

9.1 Recommendations

9.1.1 Model Hamiltonian

Relativistic model

By default we do not use relativistic effects. The best approximation is to use spin-orbit coupling, however that is computationally 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* (page 15) available (eg. TB-mBJ and GLLC-SC). The *Unrestricted* (page 32) 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* (page 19). Unfortunately there is no clear-cut answer to this problem, and one has to try in practice what works best.

9.1.2 Technical Precision

See also:

- *Which basis set should I use?* (page 55)
- *Recommendations for k-space* (page 69)

The easiest way to control the technical precision is via the *NumericalQuality* (page 53) key. One can also independently tweak the precision of specific technical aspects, e.g.:

```
BeckeGrid
  Quality Good ! tweak the grid
End
KSpace
  Quality Good  ! tweak the k-space grid
End
ZlmFit
```

(continues on next page)

(continued from previous page)

```
  Quality Normal   ! tweak the density fit
End
SoftConfinement
  Quality Basic   ! 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 an inaccurate grid.
- **KSpace**: Increase quality for metals
- **ZlmFit**: Increase quality if the SCF does not converge.
- **SoftConfinement**: Increase quality for weakly bonded systems, such as layered materials

9.1.3 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 195).

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.

Performance on machines with many cores

When running a not so big system (1000 basis functions) on a single machine with many cores you may observe a large discrepancy between the cpu and the elapsed time.

In the logfile you see for instance

```
cyc=  1 err=1.99E+00 meth=m nvec= 1 mix=0.0750 cpu=   3s ela=  27s fit=3.75E-02
```

indicating that the cpu time is 3 seconds, but the ela(psed) time was 27 seconds, much longer.

The (likely) reason is the use of shared arrays when calculating the matrix elements, which requires locking within a node.

A way to avoid this problem is to emulate as if you are using multiple nodes. Say you have a 128 core machine and add to your script

```
export SCM_SHAR_NCORES=8

$AMSBINB/ams ...
```

and tell ams that a node is made of 8 cores. When you run the job (on a single node 128 core machine) you should see in the logfile

```
AMS 2020.203  RunTime: Jan16-2021 18:05:27  Nodes: 16  Procs: 128
```

and now it is as if you are running on 16 nodes with 8 cores each. This requires more memory but this only becomes an issue with large system (10000 basis functions). In this example we need 16 times as much memory. You can use other values for the SCM_SHAR_NCORES variable, and it also works when using more than one (physical) node.

Since ams2021 a physical node may already be split automatically when ams recognizes that it uses multiple slots.

Is memory an issue for your calculation? The number of basis functions is printed in the output

```
                                    ********************************
                                    *  S C F   P R O C E D U R E   *
                                    ********************************


Nr. of basis functions                1424
Nr. of core functions                  592
Nr. of fit functions                     0
Nr. of symmetry unique K-points          1
Valence charge                      640.00
```

You take this number squared and multiply this with 16 (complex number) and you have the size of a single matrix. Then you multiply with 10 or 30 to get an estimate of the memory needed (per “node”). You should compare is to the memory per core times the nr, of cores per node.If it does not fit then the performance drop will be even more dramatic, as the system needs to swap.

Systems with many basis functions are either systems with many atoms (1000) or systems with heavy elements and a small core (gold slab).

9.1.4 Band gap calculations

The simplest method is to compare the difference between the HOMO and the LUMO to the experimental band gap. It is quite well known that regular GGAs, such as the PBE functional, underestimate the band gap this way. However, there are some functionals designed to make the HOMO LUMO gap resemble the experimental gap. These are either MetaGGAs or model potentials (for which there is no energy expression). When using MetaGGAs it is recommended to use a small frozen core, as the core orbitals will be calculated at the LDA level. The KSpace%Quality should be set to Good. The TZP basis set gives already good gaps.

We have calculated with these settings for some functionals the band gap on the Crowley (https://doi.org/10.1021/acs.jpclett.5b02870) test set (leaving out transition metal oxides and 2D slabs). The spin orbit change to the band gap was calculated only at the PBE level and used for all other functionals.

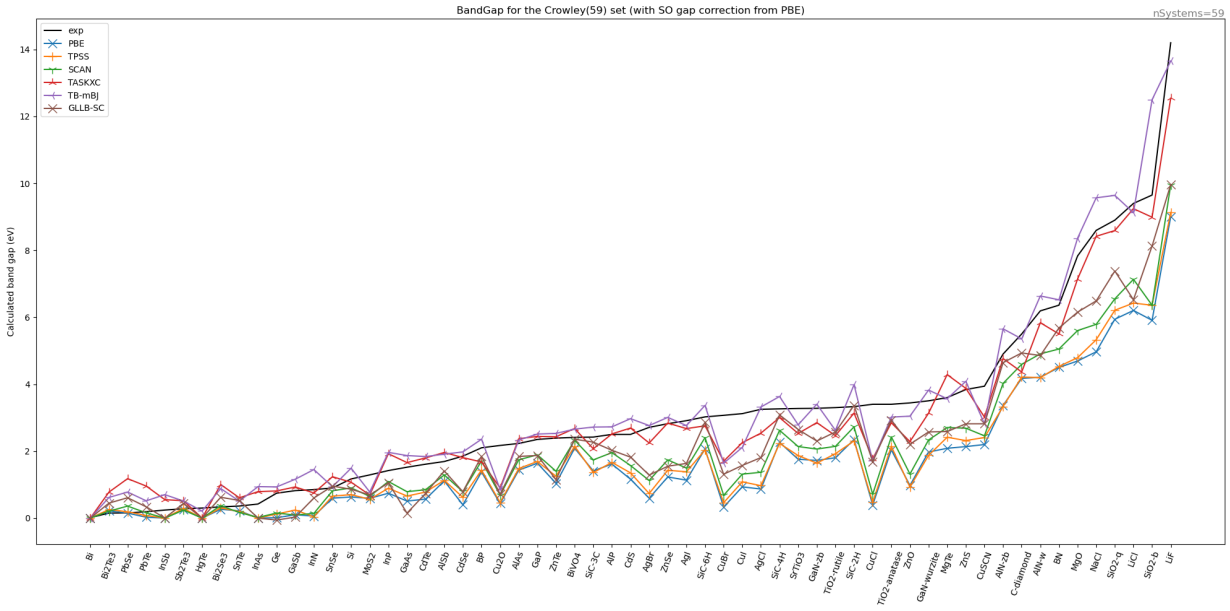


Table 9.1: Error statistics for calculated band gaps

XC	MAE	MARE
PBE	1.374	0.520
TPSS	1.288	0.489
SCAN	1.042	0.425
TASKXC	0.469	0.510
TB-mBJ	0.462	0.414
GLLB-SC	0.850	0.459

We can conclude that the TB-mBK functional appears to be best one, but the TASKXC is also doing quite well.

Here is the (Band engine) input used for the TASKXC functional

```
Basis Type=TZP Core=Small    # for metaGGA a small core is recommended
KSpace Quality=Good
XC
   metaGGA TASKXC
End
BandStructure Enabled=yes     # this gives you also the gap estimated along the␣
↪standard path through the BZ zone (in the text output)
```

The gap is either obtained from the kf file from the Variable BandStructure%BandGap, or from the text output.

in a plams script both can be obtained this way

```
try:
   gap = job_result.readrkf("BandStructure","BandGap", "band")

   gap=Units.convert(gap, 'hartree', 'eV')
except:
   gap=0.0

try:
   grep_string=job_result.grep_output('band gap from band-structure')
   assert len(grep_string) == 1
   words=grep_string[0].split(':')
   gap_from_bs=float(words[-1].strip())
except:
   gap_from_bs=0.0
```

Usually the difference is very small between the two methods.

9.2 Troubleshooting

9.2.1 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 179) 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**.

Next thing to try is the MultiSecant method. This one comes at no extra cost per SCF cycle compared to the DIIS method.

```
SCF
  Method MultiSecant
End
```

(continues on next page)

(continued from previous page)

```
MultiSecantConfig
   ! put here optional keywords to tweak the MultiSecant method
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 87).

```
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.

Finite temperature during geometry optimization

Often systems are more easy to converge when applying a finite electronic temperature. By doing so your energy will deviate from the ground state. If you are optimizing, say, a fancy molecule over a Ni slab, then you do not care too much about this when the geometry is not nearly converged yet (when the gradients are still big).

Using so-called automations, it is possible to instruct band to use a higher electronic temperature in the beginning of a geometry optimization, and a lower one at the end. Similarly you can allow for more loose SCF convergence at the start of the geometry optimization. You specify such "automations" inside the ams-level GeometryOptimization block, for example

```
GeometryOptimization

  EngineAutomations
     Gradient variable=Convergence%ElectronicTemperature InitialValue=0.01␣
↪FinalValue=0.001 HighGradient=0.1 LowGradient=1.0e-3
     Iteration variable=Convergence%Criterion InitialValue=1.0e-3 FinalValue=1.0e-6␣
↪FirstIteration=0 LastIteration=10
     Iteration variable=SCF%Iterations InitialValue=30 FinalValue=300␣
↪FirstIteration=0 LastIteration=10
  End
End
```

Let us concentrate on the first "automation". What this will do is the following.

- At the first step Convergence%ElectronicTemperature (kT) will be set to InitialValue, i.e. 0.01. (Temperatures are entered as kT in Hartree.)
- If at any step the gradient is bigger than HighGradient the temperature will be InitialValue
- If at any step the gradient is smaller than LowGradient the temperature will be FinalValue, i.e. 0.001
- If the gradient is in between HighGradient and LowGradient, a linear interpolation is done (on a logarithmic scale)
- At the last calculation FinalValue will be used, even if the geometry did not converge

Another trigger for automation is the number of geometry steps taken, shown in the two automations with "Iteration".

Let us look at the second automation.

- At the first geometry the Convergence%Criterion is relaxed to 1.0e-3.
- After the tenth geometry step this will be 1.0e-6
- In between an interpolated value will be used

The third automation shows that you can also automate SCF%Iterations. Currently only three band keywords can be automated this way.

9.2.2 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
```

9.2.3 Lattice optimization does not converge for gga

What can help is use to use analytical stress rather than numerical stress. There are three things to change for this to work.

```
SoftConfinement Radius=10.0

StrainDerivatives Analytical=yes

XC
  libxc PBE     # may also be another GGA, but not a metaGGA
End
```

Here follows an explanation why. The SoftConfinement is set to a fixed value, rather than one depending on the lattice vectors, because that is not covered by the analytical stress code. The normal value for this radius is 10, except for 3D bulk systems with a small lattice where it is set to something like the size of the lattice vector.

The other two bits are needed to trigger the use of analytical stress (which are strain derivatives of the energy). The use of the libxc library is needed, as it affords derivatives required for the stress.

9.2.4 I see two band gaps, which one is best?

The band gap is the difference between the bottom of the conduction band (BOCB) and the top of the valence band (TOVB).

It is now important to distinguish two methods to arrive at the information of TOVB, BOCB, and hence the gap. The first is the one coming from the analytical k-space integration scheme determining the fermi level and occupations, let us call this the interpolation method. The second one is a strictly post SCF method that cannot be used to determine the fermi energy and occupations. This is about calculating the bands along a path through the BZ assuming a certain fixed density/potential. Let us call this second method the from band structure method.

The advantage of the "band structure" method is that is you can use a very dense sampling of k-points along the chosen path (DeltaK). Normally this is the better way to get the gap, but you assume that both the TOVB and BOCB are on the specified path, which is not a mathematical necessity (although in practice very often true). The advantage of the "interpolation" method that is really follows the (quadratically interpolated) bands over the whole BZ. Typically the spacing between k-points is much larger than DeltaK from the plot (as the number of points for the first grows cubic and the latter only linear).

Finally the gap printed in the kf file is the one from the "interpolation method".

9.2.5 Band structure does not match the DOS

This problem may be related to the previous topic. The DOS is derived from true k-space integration called the "interpolation method" in the previous topic: it samples through interpolation the whole BZ. For the band structure plot a single line is used, and typically a much denser grid can be afforded. So make sure that the DOS is converged with respect to the KSpace%Quality paramter: try a better (or worse) value.

Ultimately a converged DOS may still not match a band structure as it is possible that the chosen line misses some features (it does not cover the whole BZ).

Finally the energy grid for the DOS may be too coarse, it can be made smaller with DOS%DeltaE.

9.2.6 Missing core bands or DOS peaks in amsbands

If you do a calculation on, say, an Al chain you may expect a core like band and corresponding DOS peak at -1500 eV. To achieve this you first need to set the frozen core to None. Even then it still does not show up, because by default BandStructure%EnergyBelowFermi is 10 Hartree (~300 eV). If you set it to something large (10000) you will now see the 1s band showing up near -1500 eV, however the corresponding DOS peak appears invisible… that is, until you zoom in the y-axis appropriately. If the DeltaE used for the dos is smaller than the height of a pixel the peak will be invisible, or very faint.

9.2.7 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**, or the **step size** used in the Phonon run is too large. Also **general accuracy** issues may be the cause, such as numerical integration, k-space integration and fit error.

9.2.8 Too much scratch disk space is used

For systems having either many basis functions or many k-points the required disk space demand can grow and te program can crash. The needed scratch space is overwhelmingly determined by (temporary) matrices written to disk. How this is done is determined by an input key. In case of this problem set

```
Programmer Kmiostoragemode=1   ! 1 means: fully distributed, 2 is the default meaning␣
↪distributed only within (ShM) nodes
```

You can increase the available scratch disk space by using more nodes. The number of nodes (according to the AMS definition) is printed in the output and logfile

```
AMS 2021.105  RunTime: Feb08-2022 19:27:17  ShM Nodes: 1  Procs: 24
```

and you can search for "ShM Nodes".

9.2.9 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.

9.2.10 Frozen core too large

BAND calculates the overlap matrix of the core functions, and this should approximate the unit matrix. To validate that this is the case two numbers are checked: the smallest eigenvalue of the overlap matrix (which should be larger than `Dependency%Core`) and the maximum absolute deviation of diagonal elements from one, which should be smaller than 1-`Dependency%Core`. When the deviation is larger then the frozen-core overlap criterion the program stops. The default (`Dependency%Core` = 0.80) is fairly tolerant. The safest solution is to choose a smaller *Frozen core* (page 57), this can also be controller per atom type (element) or per region, see *More Basis input options* (page 60). In the text output the name of the element is printed that seeems to be most responsible for the frozen core too large error. For performance reasons, however, this may not be the preferred option. In practice you might still get reliable results by setting the criterion to 0.5, see the *Dependency* (page 107) block. For the *5d* transition metals, for instance, you can often freeze the 4f orbital (using the more tolerant setting), thus reducing the basis set considerably. If you loosen the criterion 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.

9.2.11 requested kgrid appears to break the symmetry

Band creates by default a regular k-grid. In the case of 3D the size is determined by k1,k2, and k3. Normally the values are derived from KSpace%Quality and the length of the trhee reciprocal lattice vectors. If a lattice vector is long enough the corresponding k may be set to 1, which is a special value. For instance if you get for your system a 3x5x1 grid it means that effectively we have a 2 dimensional problem.

This can be problematic when there is symmetry. Perhaps there are symmetry relations between the “remaining” two vectors and the “removed” lattice vector. If such is the case the error “kgrid appears to break the symmetry” is raised.

You can either disable the symmetry

```
UseSymmetry No # inside the BandEngine block
```

or you can figure out what the automatic values for k1,k2, and k3 are.

In case of the error you see in the output

```
grid method with params           3           3           1
ERROR: requested kgrid appears to break the symmetry
       there are two solutions
       1) Disable symmetry for the engine
       2) Avoid a automatic grid like 3x3x1. Try a better KSpace%Quality.
```

So that in this case a 3x3x1 grid was chosen. In this example you probably want to increase k3

```
KSpace
    Regular
       NumberOfPoints 3 3 3 # specify k1,k2,k3 "by hand"
    End
end
```

9.3 Various issues

9.3.1 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.

```
<Oct16-2019> <11:44:37>  AMS 2019  RunTime: Oct16-2019 11:44:37  Nodes: 1  Procs: 4
<Oct16-2019> <11:44:37>  BAND 2019  RunTime: Oct16-2019 11:44:37  Nodes: 1  Procs: 4
<Oct16-2019> <11:44:37>  All basis functions smoothly confined at radius: 10.0
<Oct16-2019> <11:44:37>  >>>> RADIAL
<Oct16-2019> <11:44:38>  >>>> POINTS
<Oct16-2019> <11:44:38>  >>>> KPNT
<Oct16-2019> <11:44:39>  >>>> CELLS
<Oct16-2019> <11:44:39>  >>>> NUMGRD
<Oct16-2019> <11:44:39>  >>>> ELSTAT
<Oct16-2019> <11:44:39>  >>>> ATMFNC
<Oct16-2019> <11:44:39>  CalcAtomicProperties
<Oct16-2019> <11:44:39>  >>>> PREPAREBAS
<Oct16-2019> <11:44:39>  ------ K ..   1
<Oct16-2019> <11:44:39>  >>>> PREPAREHAM
<Oct16-2019> <11:44:39>  ------ K ..   1
<Oct16-2019> <11:44:39>  >>>> PREPAREFIT
<Oct16-2019> <11:44:39>  calling scf
<Oct16-2019> <11:44:39>  start of SCF loop
<Oct16-2019> <11:44:39>  initial density from psi
<Oct16-2019> <11:44:40>  cyc=  0 err=0.00E+00 cpu=   0s ela=   0s
<Oct16-2019> <11:44:40>  cyc=  1 err=5.88E-01 meth=m nvec= 1 mix=0.0750 cpu=   0s␣
↪ela=   0s fit=9.96E-03
<Oct16-2019> <11:44:40>  cyc=  2 err=5.35E-01 meth=d nvec= 2 mix=0.2000 cpu=   0s␣
↪ela=   0s fit=6.79E-03
<Oct16-2019> <11:44:41>  cyc=  3 err=8.63E-02 meth=d nvec= 3 mix=0.2000 cpu=   0s␣
↪ela=   0s fit=8.05E-03
<Oct16-2019> <11:44:41>  cyc=  4 err=2.10E-02 meth=d nvec= 3 mix=0.2200 cpu=   0s␣
↪ela=   0s fit=8.19E-03
<Oct16-2019> <11:44:42>  cyc=  5 err=1.46E-02 meth=d nvec= 3 mix=0.2420 cpu=   0s␣
↪ela=   0s fit=8.29E-03
```

(continues on next page)

(continued from previous page)

```
<Oct16-2019> <11:44:42>  cyc=  6 err=9.90E-03 meth=d nvec= 4 mix=0.2420 cpu=   0s␣
↪ela=   0s fit=8.28E-03
<Oct16-2019> <11:44:42>  HALFWAY
<Oct16-2019> <11:44:42>  cyc=  7 err=5.85E-04 meth=d nvec= 4 mix=0.2662 cpu=   0s␣
↪ela=   0s fit=8.28E-03
<Oct16-2019> <11:44:43>  cyc=  8 err=3.76E-04 meth=d nvec= 5 mix=0.2662 cpu=   0s␣
↪ela=   0s fit=8.29E-03
<Oct16-2019> <11:44:43>  cyc=  9 err=7.20E-05 meth=d nvec= 3 mix=0.2928 cpu=   0s␣
↪ela=   0s fit=8.29E-03
<Oct16-2019> <11:44:43>  cyc= 10 err=2.80E-05 meth=d nvec= 4 mix=0.2928 cpu=   0s␣
↪ela=   0s fit=8.29E-03
<Oct16-2019> <11:44:44>  cyc= 11 err=9.03E-06 meth=d nvec= 5 mix=0.2928 cpu=   0s␣
↪ela=   0s fit=8.29E-03
<Oct16-2019> <11:44:44>  SCF CONVERGENCE
<Oct16-2019> <11:44:44>  cyc= 12 err=1.59E-06 meth=d nvec= 5 mix=0.3221 cpu=   0s␣
↪ela=   0s fit=8.29E-03
<Oct16-2019> <11:44:44>  cyc= 13 err=1.59E-06 meth=d nvec= 1 mix=1.0000 cpu=   0s␣
↪ela=   0s fit=8.29E-03
<Oct16-2019> <11:44:44>  ENERGY OF FORMATION:   -1.1620 A.U.
<Oct16-2019> <11:44:44>                        -31.6196 E.V.
<Oct16-2019> <11:44:44>                       -729.1660 KCAL/MOL
<Oct16-2019> <11:44:44>  FERMI ENERGY:          -0.2051 A.U.
<Oct16-2019> <11:44:44>                         -5.5801 E.V
<Oct16-2019> <11:44:44>  Band gap:               0.2204 A.U.
<Oct16-2019> <11:44:44>                          5.9986 E.V
<Oct16-2019> <11:44:44>  >>>> CHARGE
<Oct16-2019> <11:44:44>  >>>> HIRSH
<Oct16-2019> <11:44:44>  >>>> CM5CHARGES
<Oct16-2019> <11:44:44>  >>>> DOS
<Oct16-2019> <11:44:44>  Storing all partial DOS
<Oct16-2019> <11:44:44>  Integrate over delta E
<Oct16-2019> <11:44:44>  partial dos
<Oct16-2019> <11:44:44>  copy T(V/VOC)
<Oct16-2019> <11:44:44>  copy eigensystem
<Oct16-2019> <11:44:45>  NORMAL TERMINATION
```

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.

The preparation phase is the part up to "start of SCF loop". The first entries are usually not very costly. The section PREPAREBAS is about the overlap matrix, core orthogonalization, and the transformation to the orthogonal basis. In PREPAREHAM the fixed part of the Hamiltonian is calculated (mostly kinetic energy).

Let us take a closer look at a line during the SCF.

```
<Jul10-2018> <18:24:59>  cyc=  3 err=4.35E-02 meth=d nvec= 2 mix=0.2200 cpu=   1s␣
↪ela=   1s fit=1.60E-02
```

The meaning of cyc is the iteration number, so it is the third iteration. The self consistent error (err) is 4.35E-02. The method (meth) to guess the density for the next cycle is d, meaning DIIS, being a linear combination (nvec) of two 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 1 seconds of real time. Finally the error of the density fitting was 1.60E-02

9.3.2 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 *UseSymmetry key* (page 189), 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
System
    Atoms
      Fe.a    0.0    0.0    0.0
      Fe.b  -1.435 -1.435  1.435
    End
End

Lattice
   -1.435  1.435  1.435
    1.435 -1.435  1.435
    2.87   2.87  -2.87
End

...
...

Band Engine

  ...
  CONVERGENCE
      CRITERION 1.0e-4
      Degenerate default
      SpinFlip 2  ! Flip (startup) spin density at second atom
  END
  ...
EndEngine
```

Another solution is to use the expert SYMMETRY keyword.

9.3.3 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`.

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
```

(continues on next page)

(continued from previous page)

```
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`.

9.3.4 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.

9.3.5 Numerical Atoms and Basis 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 $AMSHOME/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.

9.4 Warnings

9.4.1 Warnings specific to periodic codes (BAND, DFTB)

```
WARNING: GUESS: INPUT LATTICE MAY BE INACCURATE
WARNING: KSPACE ONLY GAMMA POINT.
WARNING: LINEAR INTEGRATION IN K-SPACE
WARNING: Problem with Brillouin zone k-path generation (see output file)
WARNING: problematic mapping to central cell
WARNING: QUAD2: electronic temperature problem
```

CHAPTER

TEN

EXAMPLES

10.1 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 $AMSHOME/examples/, where $AMSHOME is the main directory of the ADF package. The main subdirectory for the BAND examples is $AMSHOME/examples/band. Each sample run has its own directory. For instance, $AMSHOME/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 AMSBIN and AMSRESOURCES. 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 AMSBIN and AMSRESOURCES in your environment. If a parallel 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 $AMSBIN directory, however, you may find that the program executables have names 'adf.exe', 'band.exe'. There are also files in $AMSBIN 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.

10.2 Model Hamiltonians

10.2.1 Example: Spin polarization: antiferromagnetic iron

Download BetaIron.run

```
#! /bin/sh

# By setting 'Unrestricted Yes' we do a spin polarized calculation. Normally
# this would converge to the ferromagnetic solution.

# With the SpinFlipRegion 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.

$AMSBIN/ams <<eor

Task SinglePoint

System
  Atoms
    Fe   0.0    0.0    0.0
    Fe  -1.435 -1.435  1.435 region=flip
  end

  Lattice
     -1.435  1.435  1.435
      1.435 -1.435  1.435
      2.87   2.87  -2.87
  End
End

Engine Band
  Title Beta iron

  Convergence
     Criterion 1.0e-4
     SpinFlipRegion flip # Flip (startup) spin density at second atom
  End

  Unrestricted Yes

  Print AtomicChargesDetails
EndEngine
eor
```

10.2.2 Example: Applying a Magnetic Field (S . B)

Download BFieldB.run

```
#!/bin/bash

myreport () {
  mX=`$AMSBIN/amsreport $AMS_JOBNAME.results -k "Properties%Value(2)#6.3f" | awk '
↪{print $1}'`
  mY=`$AMSBIN/amsreport $AMS_JOBNAME.results -k "Properties%Value(2)#6.3f" | awk '
↪{print $2}'`
  mZ=`$AMSBIN/amsreport $AMS_JOBNAME.results -k "Properties%Value(2)#6.3f" | awk '
↪{print $3}'`
  energy=`$AMSBIN/amsreport $AMS_JOBNAME.results -k "AMSResults%Energy#10.6f"`
  nCycles=`$AMSBIN/amsreport $AMS_JOBNAME.results -k "SCCLogger%nEntries"`
  printf "%20s %10s %10.3f %10.3f %10.3f %10s\n"  $BFieldComp $nCycles $mX $mY $mZ
↪$energy
}

LC_NUMERIC=en_US.UTF-8

export NSCM=1

BField=1.0e-1

BFieldComp=bY

report=report.txt

echo "" > $report

echo "We run a H atom with three different directions of a finite B field" >> $report
echo "with the NonCollinear model." >> $report
echo "This way the magnetization vector m should align with the B field vector." >>
↪$report

printf "\n%20s %10s %10s %10s %10s %10s\n" "BField" "nCycles" "mX" "mY" "mZ" "energy"␣
↪>> $report

for BFieldComp in bX bY bZ
do

export AMS_JOBNAME=H.$BFieldComp

$AMSBIN/ams --delete-old-results <<eor

Task SinglePoint

System
  Atoms
    H 0 0 0
  End
End

Engine Band

  Relativity
    Level Spin-Orbit
```

(continues on next page)

(continued from previous page)

```
  End

  xc
    SpinOrbitMagnetization NonCollinear
  End

  BField
    Unit a.u.
    $BFieldComp $BField
  End

  Print AtomicChargesDetails
EndEngine
eor

myreport >> $report

done

echo "begin report"
cat $report
echo "end report"
```

10.2.3 Example: Applying a Magnetic Field (L . B)

Download BFieldLdotB.run

```
#!/bin/sh

export AMS_JOBNAME=run1

$AMSBIN/ams <<EOF
Task SinglePoint

System
   GeometryFile $AMSHOME/atomicdata/Molecules/TestMols/Methane.xyz
End

Engine Band

   BField
    Unit A.U.
    BZ 0.001
    Method NR_LDOTB
   end

   Basis
     Type QZ4P
     Core None
   End

EndEngine

EOF
```

(continues on next page)

(continued from previous page)

```
echo "Begin of shielding row for all atoms, unit-=ppm"
$AMSBIN/amsreport $AMS_JOBNAME'.results/band.rkf' -k 'Magnetic properties
↪%ShieldingRowAtNuclei(ppm)#12.5f##3'
echo "End of shielding row"


export AMS_JOBNAME=run2

$AMSBIN/ams <<EOF
Task SinglePoint

System
   GeometryFile $AMSHOME/atomicdata/Molecules/TestMols/Methane.xyz
End

Engine Band

   BField
    Unit A.U.
    BZ 0.001
    Method NR_LDOTB
    Dipole true
    DipoleAtom 1
   end

   Basis
     Type QZ4P
     Core None
   End

EndEngine

EOF

echo "Begin of shielding row for all atoms, unit-=ppm"
$AMSBIN/amsreport $AMS_JOBNAME'.results/band.rkf' -k 'Magnetic properties
↪%ShieldingRowAtNuclei(ppm)#12.5f##3'
echo "End of shielding row"
```

10.2.4 Example: Graphene sheet with dispersion correction

Download Graphene_Dispersion.run

```
#! /bin/sh

# 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 ==
```

(continues on next page)

(continued from previous page)

```
AMS_JOBNAME=default $AMSBIN/ams <<eor

Task SinglePoint

System
   GeometryFile $AMSHOME/examples/band/Graphene_Dispersion/Graphene_double_layer.xyz
End

Engine Band
   XC
      gga scf bp86
      dispersion default
   End

   NumericalQuality Basic

   Basis
      Type TZP
      Core Large
   End
EndEngine
eor

# == Second run: dispersion Grimme3 ==

AMS_JOBNAME=grimme3 $AMSBIN/ams <<eor

Task SinglePoint

System
   GeometryFile $AMSHOME/examples/band/Graphene_Dispersion/Graphene_double_layer.xyz
End

Properties
  Gradients True
End

Engine Band
   Output
      Print Section=Properties Level=Detail
   End

   XC
      gga scf blyp
      dispersion Grimme3
   end

   NumericalQuality Basic

   Basis
      Type TZP
      Core Large
   End
EndEngine
eor

# == Third run: dispersion Grimme3 bjdamp ==
```

(continues on next page)

(continued from previous page)

```
AMS_JOBNAME=grimme3_bjdamp $AMSBIN/ams <<eor

Task SinglePoint

System
   GeometryFile $AMSHOME/examples/band/Graphene_Dispersion/Graphene_double_layer.xyz
End

Properties
  Gradients True
End

Engine Band
   Output
      Print Section=Properties Level=Detail
   End

   XC
      gga scf blyp
      dispersion Grimme3 bjdamp
   end

   NumericalQuality Basic

   Basis
      Type TZP
      Core Large
   End
EndEngine
eor
```

10.2.5 Example: H on perovskite with the COSMO solvation model

Download HonPerovskite_Solvation.run

```
#! /bin/sh

# 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.

$AMSBIN/ams <<eor

Task SinglePoint

System
   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
```

(continues on next page)

(continued from previous page)

```
      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
End

Properties
   Gradients True
End

Engine Band
   TITLE Hydrogen on Perovskite wit solvation

   Solvation
      Enabled True
      Surf Delley
      charge method=inver
      Solvent
         Eps 78.4
         Rad 1.4
      End
   End

   PeriodicSolvation
      nstar 3
      SymmetrizeSurfacePoints true
      RemovePointsWithNegativeZ true
   End

   Screening
      rmadel 30 ! to speed up the calculation
   End

   Convergence
      Criterion 1.0e-4
   End

   Basis
      Type SZ
      Core Large
   End
EndEngine
eor
```

10.2.6 Example: Applying a homogeneous electric field

Download EField.run

```
#!/bin/sh

# With the EFIELD keyword you can specify a static electric field in the
# z-direction.

# == first run: field of 1.5427 Volt/Angstrom ==

$AMSBIN/ams <<eor

Task SinglePoint

System
   lattice [Bohr]
      15.0 0.0  0.0
      0.0  15.0 0.0
   End
   Atoms [Bohr]
      H   0.0 0.0 0.0
      Li  0.0 1.0 3.0
   End
   ElectrostaticEmbedding
      ElectricField 0.0 0.0 1.5427
   End
End

Properties
  Gradients True
End

Engine Band
   Output
      Print Section=Properties Level=Detail
   End
   KSpace
      Quality GammaOnly
   End

   Basis
      Type TZP
      Core Large
   End
EndEngine
eor

rm -r ams.results

# == second run: field of -1 Volt/Angstrom ==

$AMSBIN/ams <<eor

Task SinglePoint

System
   lattice [Bohr]
```

(continues on next page)

(continued from previous page)

```
      15.0 0.0  0.0
      0.0  15.0 0.0
   End
   Atoms [Bohr]
      H   0.0 0.0 0.0
      Li  0.0 1.0 3.0
   End
   ElectrostaticEmbedding
      ElectricField 0.0 0.0 -1.0
   End
End

Properties
  Gradients True
End

Engine Band

Output
   Print Section=Properties Level=Detail
End


   KSpace
      Quality GammaOnly
   End

   Basis
      Type TZP
      Core Large
   End
EndEngine

eor
```

10.2.7 Example: Finite nucleus

Download FiniteNucleus.run

```
#! /bin/sh

# 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, and the
# A-tensor. For such calculations one needs to crank up the precision and also
# use a relativistic Hamiltonian.

# == First run: NuclarModel PointCharge ==

AMS_JOBNAME=PointCharge $AMSBIN/ams <<eor

Task SinglePoint

System
  lattice
     30.0 0.0 0.0
```

(continues on next page)

(continued from previous page)

```
  End

  Atoms
     Au  0.000000      0.000000      0.000000
  End
End

Engine Band
  NuclearModel PointCharge

  Efg
    Enabled True
  End

  Atensor
    Enabled True
  End

  Unrestricted
  Relativity
    Level Scalar
  End

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

  RadialDefaults
     nr 10000
  End

  NumericalQuality Good

  Basis
     Type TZ2P
     Core None
  End

  XC
     gga PBE
  END
EndEngine
eor

# == Second run: NuclearModel Gaussian ==

AMS_JOBNAME=Gaussian $AMSBIN/ams <<eor

Task SinglePoint

System
  lattice
     30.0 0.0 0.0
```

(continues on next page)

(continued from previous page)

```
  End

  Atoms
     Au  0.000000      0.000000      0.000000
  End
End

Engine Band
  NuclearModel Gaussian

  Efg
    Enabled True
  End

  Atensor
    Enabled True
  End

  Unrestricted
  Relativity
    Level Scalar
  End

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

  RadialDefaults
     nr 10000
  End

  NumericalQuality Good

  Basis
     Type TZ2P
     Core None
  End

  XC
     gga PBE
  END
EndEngine

eor
```

10.2.8 Example: Fixing the Band gap of NiO with GGA+U

Download NiO_Hubbard.run

```
#! /bin/sh

# 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.


$AMSBIN/ams <<eor

Task SinglePoint

System
   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
End

Engine Band
   Title NiO GGA+U (Hubbard)

   Unrestricted Yes

   HubbardU
      Enabled True
      PrintOccupations True
      Atom Element=Ni uValue=0.3 lValue=d
   End

   KSpace
      Symmetric KInteg=3
      Type Symmetric
   End

   Basis
      Type TZP
      Core Large
   End

   XC
      GGA Becke Perdew
   End

   Print AtomicChargesDetails
EndEngine
```

(continues on next page)

(continued from previous page)

```
eor
```

10.2.9 Example: Hubbard combined with spin flip

The Hubbard model can be combined with spin flip. The toy system consists of two Cr atoms and two H atoms, the spin is flipped on one of the Cr atoms.

In this example it is shown how to achieve the same using three different input methods, of which the first (atom) is almost always the most convenient.

Download HubbardInputOptions.run

```
#!/bin/bash

report=report.txt

print_header() {
   echo "* This is a fairly advanced application of the Hubbard model"
   echo "* The toy model is a Cr dimer surrounded by two H atoms"
   echo "* It is a spin polarized calculation with the spin is flipped on one of the␣
↪Cr atoms"
   echo "* The d orbitals of the two Cr atoms get a Hubbard U correction"
   echo "* Three equivalent input setups are shown to achieve the same"
   echo "* In the output feedback is given after the header: H U B B A R D    S E T T␣
↪I N G S"

   printf "\n%10s %10s %10s\n" "input" "Energy" "E(hubbard)"
}

myreport () {
   energy=`$AMSBIN/amsreport $AMS_JOBNAME.results -k "AMSResults%Energy#10.6f"`
   hubbard_energy=`$AMSBIN/amsreport $AMS_JOBNAME.results/band.rkf -k "Bond energy␣
↪terms%Hubbard Energy#10.6f"`

   printf "%10s %10s %10s\n" $1 $energy $hubbard_energy
}


print_header > $report


# Method 1, set hubbard for all Cr atoms

export AMS_JOBNAME=inp=atom

$AMSBIN/ams --delete-old-results <<EOF

Task SinglePoint

System
   Atoms
      Cr     0.0000     0.0000     0.0000
      H      10.0       0.0        0.0
      Cr     0.0000     0.0000     1.6790  region=SpinFlip
      H      -10.0       0.0       0.0
```

(continues on next page)

(continued from previous page)

```
   End
End


Engine band
   Unrestricted yes

   Convergence  SpinFlipRegion=SpinFlip

   HubbardU
      Enabled yes
      PrintOccupations No
      Atom Element=Cr UValue=0.02 lValue=d
   End
EndEngine


EOF

myreport "atoms" >> $report

# Method 2, set hubbard for atoms in a region

export AMS_JOBNAME=inp=region

$AMSBIN/ams --delete-old-results <<EOF

Task SinglePoint

System
   Atoms
      Cr     0.0000     0.0000     0.0000  region=AllCr
      H      10.0       0.0        0.0
      Cr     0.0000     0.0000     1.6790  region=SpinFlip,AllCr
      H      -10.0       0.0       0.0
   End
End


Engine band
   Unrestricted yes

   Convergence  SpinFlipRegion=SpinFlip

   HubbardU
      Enabled yes
      PrintOccupations No
      Region Name=AllCr UValue=0.02 lValue=d
   End
EndEngine


EOF

myreport "regions" >> $report

# Method 3: the old and inconvenient way, relying on knowledge of Band's internal␣
```

(continues on next page)

(continued from previous page)

```
↪atom type system.

export AMS_JOBNAME=inp=obsolete

$AMSBIN/ams --delete-old-results <<EOF

Task SinglePoint

System
   Atoms
      Cr     0.0000     0.0000     0.0000
      H      10.0       0.0        0.0
      Cr     0.0000     0.0000     1.6790  region=SpinFlip
      H      -10.0       0.0       0.0
   End
End


Engine band
   Unrestricted yes

   Convergence  SpinFlipRegion=SpinFlip

   HubbardU
      Enabled yes  # band sees three types, 1 and 3 are the Cr atoms
      PrintOccupations No
      UValue 0.02    0.0   0.02
      lValue 2      -1     2
   End
EndEngine


EOF

myreport "obsolete" >> $report

echo "begin report"
cat $report
echo "end report"
```

10.2.10 Example: Fixing the band gap of ZnS with the TB-mBJ model potential

Download ZnS_ModelPotential.run

```
#! /bin/sh

# With the XC subkey model we invoke the so-called TB-mBJ model potential, which
# increases band gaps for solids.



AMS_JOBNAME=TB-mBJ $AMSBIN/ams <<eor

Task SinglePoint
```

(continues on next page)

(continued from previous page)

```
System
   ATOMS
      Zn  0.0000  0.0000  0.0000
      S   1.3525  1.3525  1.3525
   END

   Lattice
      0.000  2.705  2.705
      2.705  0.000  2.705
      2.705  2.705  0.000
   End
End

Engine Band
   TITLE ZnS pot=TB-mBJ

   scf method=DIIS

   XC
      model TB-mBJ
   END

   Basis
      Type DZ
      Core Large
   End
EndEngine
eor


AMS_JOBNAME=GLLB-SC $AMSBIN/ams <<eor

Task SinglePoint

System
   ATOMS
      Zn  0.0000  0.0000  0.0000
      S   1.3525  1.3525  1.3525
   END
   Lattice
      0.000  2.705  2.705
      2.705  0.000  2.705
      2.705  2.705  0.000
   End
End

Engine Band
   TITLE ZnS pot=GLLB-SC

   scf method=DIIS

   XC
      model GLLB-SC
   END

   Basis
      Type DZ
```

(continues on next page)

(continued from previous page)

```
      Core Large
   End
EndEngine
eor


AMS_JOBNAME=lb94 $AMSBIN/ams <<eor

Task SinglePoint

System
   Atoms
   H       0.000000000     0.000000000    -0.370500000
   H       0.000000000     0.000000000     0.370500000
   End
End

Engine Band
   Title H2 pot=lb94

   scf method=DIIS

   XC
       model lb94
   end

   Basis
      Type TZP
      Core Large
   End
EndEngine
eor
```

10.2.11 Example: DFT-1/2 method for Silicon

Download DFTHalf_Si.run

```
#! /bin/sh

"$AMSBIN/ams" <<eor

Task SinglePoint
System
   Atoms
      Si -0.67875 -0.67875 -0.67875
      Si  0.67875  0.67875  0.67875
   End
   Lattice
      0.0 2.715 2.715
      2.715 0.0 2.715
      2.715 2.715 0.0
   End
End

Engine BAND
```

(continues on next page)

(continued from previous page)

```
   # To get better results one should use a larger basis set and
   # a better k-space quality (e.g. 'Basis Type=TZ2P' and 'KSpace Quality=Good')

   Basis Type=DZP

   XC
      LDA SCF VWN
      DFTHalf
         Enabled Yes
         ActiveAtomtype
            AtomType Si
            IonicCharge 0.1
            ScreeningCutOffs 2.0 4.0 6.0
         End
      End
   End
EndEngine
eor
```

10.3 Precision and performance

10.3.1 Example: Convenient way to specify a basis set

Download BasisDefaults.run

```
#! /bin/sh

# This example shows some of the flexibility of the Basis 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 or to directly
# specify basis set files for particular atom types.

$AMSBIN/ams <<eor

Task SinglePoint

System
   Atoms [Bohr]
      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.custom       4.0   1.43  0.0
   End
End

Engine Band
   Title CO + H2: fine tuning the basis defaults

   NumericalQuality Basic

   Basis
      ! Cheap defaults
```

(continues on next page)

(continued from previous page)

```
      Type DZ
      Core Large
      PerAtomType Symbol=C              Core=None    ! This C has no frozen core
      PerAtomType Symbol=O.large_basis  Type=TZ2P    ! This O with a larger basis
      PerAtomType Symbol=H.large_basis  Type=V       ! This one also with a larger␣
↪basis
      PerAtomType Symbol=H.custom File=DZ/H
   End
EndEngine
eor
```

10.3.2 Example: Tuning precision and performance

Download Peptide_NumericalQuality.run

```
#! /bin/sh

# This example shows how to tune the numerical quality of the calculation. This
# will influence both efficiency and accuracy of the calculation.

$AMSBIN/ams <<eor

Task SinglePoint

System
   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
End

Engine Band
   TITLE Quality

   NumericalQuality Normal

   ZlmFit
      Quality Normal
   End
```

(continues on next page)

(continued from previous page)

```
   BeckeGrid
      Quality Basic
   End

   KSpace
      Quality Basic
   End

   SoftConfinement
      Quality VeryGood
   End

   Basis
      Type DZ
      Core Large
   End

   Relativity Level=None

   XC
      GGA PBE
   END
EndEngine
eor
```

10.3.3 Example: Multiresolution

Download Multiresolution_H2O.run

```
#! /bin/sh

# 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). The system can be divided into sub-regions and
# different levels of numerical accuracy can be used for each of these sub-
# regions.

# In this example we will tweak:
# - the basis set (Basis)
# - the numerical integration (BeckeGrid)
# - the density fitting for Coulomb potential (ZlmFit)
# - the fit-set used in the Hartree-Fock Resolution of identity (RIHartreeFock)

# Note: For the regions for which no quality has been explicitly defined
# through a QualityPerRegion keyword, the quality defined in NumericalQuality
# will be used (Normal, in this example).

$AMSBIN/ams <<eor

Task SinglePoint

System
```

(continues on next page)

(continued from previous page)

```
  Atoms
     O    0.00000000       0.00000000       0.00000000    region=Accurate
     H    0.40399229      -0.65578342       0.63241539    region=Accurate
     H    0.81410032       0.55624569      -0.23129097    region=Accurate
     O   -3.02535626      -0.08473104      -0.47678489
     H   -2.56531481       0.62644005       0.07759386
     H   -2.25289644      -0.61700366      -0.80790649
     O    2.95394790      -0.54939973      -0.38206034
     H    3.91427727      -0.21304908      -0.44738291
     H    2.87780992      -1.13241278      -1.20853726
     O   -5.95425742      -0.56764616      -0.02016682    region=Far
     H   -5.26308282      -0.46969096       0.69255963    region=Far
     H   -5.42117992      -0.54361203      -0.86443121    region=Far
     O    6.25171470      -0.62004899      -0.03702467    region=Far
     H    6.16508647      -1.38696453       0.58541903    region=Far
     H    7.09161199      -0.16700550       0.23679419    region=Far
  End
End

Engine Band

   XC
      LibXC B3LYP
   End

   ! ========================================================
   ! Set different basis sets for atoms in different regions:
   ! ========================================================

   Basis
      Type DZ
      Core None
      PerRegion Region=Accurate Type=TZP
      PerRegion Region=Far      Type=SZ
   End

   ! ============================================================
   ! Set the NumerialQuality to be used for the atoms that are not
   ! explicitly defined through a QualityPerRegion keyword
   ! ============================================================

   NumericalQuality Normal

   ! ======================
   ! Numerical integration:
   ! ======================

   BeckeGrid
      QualityPerRegion Region=Accurate Quality=Good
      QualityPerRegion Region=Far      Quality=Basic
   End

   ! ====================================
   ! Density fitting for Coulomb potential:
   ! ====================================

   ZlmFit
```

(continues on next page)

(continued from previous page)

```
      QualityPerRegion Region=Accurate Quality=Good
      QualityPerRegion Region=Far      Quality=Basic
   End

   ! ============================================================
   ! Hartree-Fock Resolution of identity (for hybrid functionals)
   ! ============================================================

   RIHartreeFock
      QualityPerRegion Region=Accurate Quality=Good
      QualityPerRegion Region=Far      Quality=Basic
   End

   Relativity Level=None

EndEngine

eor
```

10.3.4 Example: BSSE correction

Download BSSE.run

```
#!/bin/bash

# This example shows how to calculate the basis set superposition error for the
# interaction of CO with H2. In this shell script we loop over progressively
# better basis sets.

for bas in DZ TZ2P QZ4P
do

$AMSBIN/ams <<eor
Task SinglePoint
System
  Atoms [Bohr]
    C 0 0 0
    O 2.13 0 0
    H 0 0 6
    H 1.5 0 6
  End
End

Engine Band
  Basis
    Type $bas
  End
EndEngine
eor

ECOH2=`$AMSBIN/amsreport  ams.results/band.rkf -r 'Bond energies%final bond energy'`
rm -r ams.results

$AMSBIN/ams <<eor
Task SinglePoint
```

(continues on next page)

(continued from previous page)

```
System
  Atoms [Bohr]
    H 0 0 6
    H 1.5 0 6
  End
End

Engine Band
  Basis
    Type $bas
  End
EndEngine

eor

EH2=`$AMSBIN/amsreport  ams.results/band.rkf -r 'Bond energies%final bond energy'`
rm -r ams.results


$AMSBIN/ams <<eor
Task SinglePoint
System
  Atoms [Bohr]
    Gh.C 0 0 0
    Gh.O 2.13 0 0
    H 0 0 6
    H 1.5 0 6
  End
End

Engine Band
  Basis
    Type $bas
  End
EndEngine

eor

EH2_GHOST_CO=`$AMSBIN/amsreport  ams.results/band.rkf -r 'Bond energies%final bond␣
↪energy'`
rm -r ams.results


$AMSBIN/ams <<eor
Task SinglePoint
System
  Atoms [Bohr]
    C 0 0 0
    O 2.13 0 0
  End
End

Engine Band
  Basis
    Type $bas
  End
EndEngine
```

(continues on next page)

(continued from previous page)

```
eor

ECO=`$AMSBIN/amsreport  ams.results/band.rkf -r 'Bond energies%final bond energy'`
rm -r ams.results


$AMSBIN/ams <<eor
Task SinglePoint
System
  Atoms [Bohr]
    C 0 0 0
    O 2.13 0 0
    Gh.H 0 0 6
    Gh.H 1.5 0 6
  End
End

Engine Band
  Basis
    Type $bas
  End
EndEngine

eor

ECO_GHOST_H2=`$AMSBIN/amsreport  ams.results/band.rkf -r 'Bond energies%final bond
↪energy'`
rm -r ams.results


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) : $ECO_GHOST_H2"
BSSEEV=`$AMSBIN/amspython -c "print (( $EH2 - $EH2_GHOST_CO + $ECO - $ECO_GHOST_H2 ) *
↪$EV)"`
echo "BSSE correction: $BSSEEV (eV)"
BOND1EV=`$AMSBIN/amspython -c "print (( $ECOH2 - $EH2 - $ECO ) *$EV)"`
BOND2EV=`$AMSBIN/amspython -c "print ($BOND1EV + $BSSEEV)"`
echo "Bond energy: $BOND1EV (eV)"
echo "Bond energy + BSSE: $BOND2EV (eV)"
echo "End report"

done
```

10.3.5 Example: Speed up SCF during geometry optimization

Generally the SCF converges more quickly when using a finite electronic temperature.

In this example it is shown (for a toy system that does not need the trick) how this can be done.

The report shows how the value of kT varies during a geometry optimization.

Download report BandAutomations.txt

```
We use a gradient dependent KT value (finite electronic temperature)

The value of kT gets progressively lower during the optimization


For two optimizers we do 3 steps and they do not converge. Yet the last single point␣
↪should be done at KTlow=0.001

kT series for optimizer: Quasi-Newton
0.010000
0.007196
0.005094
0.002040
0.001000
0.001000
(the last kT should be 0.001)


scf converge serie for optimizer: Quasi-Newton
1.0E-03
1.0E-06
1.0E-06
1.0E-06
1.0E-06

kT series for optimizer: FIRE
0.010000
0.010000
0.010000
0.009000
0.007105
0.004077
0.001000
(the last kT should be 0.001)


scf converge serie for optimizer: FIRE
1.0E-03
1.0E-06
1.0E-06
1.0E-06
1.0E-06
1.0E-06
```

Download BandAutomations.run

```
#!/bin/bash
```

(continues on next page)

(continued from previous page)

```
# the System is extremely artificial but the calculation points out something useful

# The system has two CO molecules, one of which is compressed.
# We freeze the coordinates of the compressed CO molecules

# We define a gradient dependent electronic temperature (excluding the gradient of␣
↪the constrained atoms)
# When far from convergence a higher value is used to ease SCF convergence (not␣
↪relevant to this system)
# When the gradients become small the temperature is lowered, so that is will have␣
↪negligible influence on the energy

# Here we let on purpose not converge the geometry optimization
# The final calculation should be performed as a normal single point and we␣
↪explicitly set in band the ElectronicTemperature to 0.001

report=report.txt

echo "We use a gradient dependent KT value (finite electronic temperature)" > $report
printf "\nThe value of kT gets progressively lower during the optimization\n\n" >>
↪$report
printf "\nFor two optimizers we do 3 steps and they do not converge. Yet the last␣
↪single point should be done at KTlow=0.001\n\n" >> $report

for optim in Quasi-Newton FIRE
do

export AMS_JOBNAME=test.optim=$optim

rm -rf $AMS_JOBNAME.results

$AMSBIN/ams<<EOF

Log
   Debug AutomationInteractionModule
End

Task GeometryOptimization

GeometryOptimization
   Method $optim
   MaxIterations 5

   EngineAutomations
      Gradient variable=Convergence%ElectronicTemperature InitialValue=0.01␣
↪FinalValue=0.001 HighGradient=0.1 LowGradient=1.0e-3
      Iteration variable=Convergence%Criterion InitialValue=1.0e-3 FinalValue=1.0e-6␣
↪FirstIteration=0 LastIteration=1
      Iteration variable=SCF%Iterations InitialValue=1 FinalValue=300␣
↪FirstIteration=0 LastIteration=1
   End
End

Constraints
   Atom 3
   Atom 4
End
```

(continues on next page)

(continued from previous page)

```
System
   Atoms
      C 0.0 0.0 0.0
      O 1.13 0.0 0.0
      C 0.0 5.0 0.0
      O 1.0 5.0 0.0
   End
End

Engine Band
   Basis Type=DZ
   Convergence
      ElectronicTemperature 0.001
      NumBoltz 100
   End
   NumericalQuality Basic
EndEngine

EOF

echo "kT series for optimizer: $optim" >> $report
grep "temperature kT" $AMS_JOBNAME.results/ams.log | awk '{print $NF}' >> $report
echo "(the last kT should be 0.001)" >> $report
echo """>>$report

echo """>>$report
echo "scf converge series for optimizer: $optim" >> $report
grep "automated value for Convergence%Criterion" $AMS_JOBNAME.results/ams.log | awk '
↪{print $NF}' >> $report
echo """>>$report

tmp='.0'
echo """>>$report
echo "scf max Iterations series for optimizer: $optim" >> $report
echo "  (converted to a real number to be able to catch a diff)" >> $report
grep "automated value for SCF%Iterations" $AMS_JOBNAME.results/ams.log | awk '{print
↪$NF ".0"}' >> $report
echo """>>$report

done

echo "begin report"
cat $report
echo "end report"
```

10.4 GTO

10.4.1 Example: eigenvalue-only self-consistent GW@PBE calculation: Ne with GTO type basis set

Download GW_Ne_GTO.run

```
#! /bin/sh

# GW calculation for Water. By default. The highest 5 occupied and lowest 5
# unoccupied states are calculated.

# We use an all-electron basis set since core-correlation effects are important.
# A large basis set is recommended for relatively well converged QP energies.
# With the 5Z GTO type basis set, we expect the QP energies to be converged
# within 0.1 eV.

# Fot this example we will use the GGA PBE.
# This is NOT a recommended starting point for a G0W0 calculation.
# However, the eigenvalue-only self-consistency removes most of the
# starting point dependence of the functional.
# Thererfore, PBE is a reasonable choice.

# RECOMMENDED: VeryGood numerical quality,
# especially in self-consistent GW calculations.

# We calculate the auxiliary fit set from products of primary basis functions.
# When the primary basis contains functions with angular momenta of l=5, like
# the cc-pV5Z one, we need functions in the fit with angular momenta of l=10.
# Building the auxiliary basis from fit functions ensures this.

$AMSBIN/ams << eor
Symmetry
  SymmetrizeTolerance 0.001
End

System
  Atoms
    Ne 0.0 0.0 0.0
  End
  Symmetrize Yes
End

task SinglePoint

Engine band
  Basis
    Core None
    Type GTO/CC-PVTZ
  end

  NumericalQuality VeryGood

  Dependency
     AllowBasisDependency True
     basis 1e-4
```

(continues on next page)

(continued from previous page)

```
  End

  RIHartreeFock
    DependencyCoreRange 0.0
    FitGenerationDetails
      Method FromBasisProducts
      OneCenterDependencyThreshold 1e-08
    End
    FitSetQuality FromBasisProducts
  End

  SoftConfinement
     Quality Excellent
  End

  relativity
     level None
  End

  usesymmetry False

  XC
    gga PBE
  end

  GW
     nStates 3
     selfconsistency evGW
  END
EndEngine
eor
```

10.5 Restarts

10.5.1 Example: Restart the SCF

Download RestartSCF.run

```
#! /bin/sh

# 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 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 --------------------------

AMS_JOBNAME=BChain $AMSBIN/ams <<eor

Task SinglePoint
```

(continues on next page)

(continued from previous page)

```
System
   Lattice [Bohr]
      4.0 0.0 0.0
   End
   Atoms [Bohr]
      B 0.0 0.0 0.0
   End
End

Engine Band
   Title B chain

   NumericalQuality Good

   skip dos

   XC
      GGA Becke Perdew
   END

   UNRESTRICTED
   Relativity Level=None

   DIIS
      NCycleDamp 0
      DiMix 0.5
      Adaptable false ! Otherwise it converges to a spin-restricted solution
   End

   Basis
      Type TZ2P
      Core Large
   End
EndEngine

eor

# --- let us first try a EngineRestart

AMS_JOBNAME=restart_engine $AMSBIN/ams <<eor

Task SinglePoint

EngineRestart BChain.results/band.rkf

System
   Lattice [Bohr]
      4.0 0.0 0.0
   End
   Atoms [Bohr]
      B 0.0 0.0 0.0
   End
End

Engine Band
   Title B chain restart via EngineRestart
```

(continues on next page)

(continued from previous page)

```
   NumericalQuality Good

   XC
      GGA Becke Perdew
   END

   UNRESTRICTED
   Relativity Level=None


   Basis
      Type TZ2P
      Core Large
   End
EndEngine

eor


# ---------------------------- second run -------------------------

AMS_JOBNAME=restart_1 $AMSBIN/ams <<eor

Task SinglePoint

System
   Lattice [Bohr]
      4.0 0.0 0.0
   End
   Atoms [Bohr]
      B 0.0 0.0 0.0
   End
End

Engine Band
   Title B chain restart

   NumericalQuality Good

   XC
      GGA Becke Perdew
   END

   UNRESTRICTED
   Relativity Level=None

   Restart
      File BChain.results/band.rkf
      scf
   end

   Basis
      Type TZ2P
      Core Large
   End
EndEngine
```

(continues on next page)

(continued from previous page)

```
eor

# ---------------------------- third run -------------------------

AMS_JOBNAME=BChain_SZ $AMSBIN/ams <<eor

Task SinglePoint

System
   Lattice [Bohr]
      4.0 0.0 0.0
   End
   Atoms [Bohr]
      B 0.0 0.0 0.0
   End
End

Engine Band
   Title B chain bas_SZ

   NumericalQuality Good

   Save DensityMatrix

   skip dos

   XC
      GGA Becke Perdew
   END

   UNRESTRICTED
   Relativity Level=None

   DIIS
      NCycleDamp 0
      DiMix 0.3
      Adaptable false ! Otherwise it converges to a spin-restricted solution
   End

   Basis
      Type SZ
      Core Large
   End
EndEngine

eor

# ---------------------------- fourth run -------------------------


AMS_JOBNAME=restart_2 $AMSBIN/ams <<eor

Task SinglePoint

System
   Lattice [Bohr]
```

(continues on next page)

(continued from previous page)

```
      4.0 0.0 0.0
   End
   Atoms [Bohr]
      B 0.0 0.0 0.0
   End
End

Engine Band
   Title B chain restart bas_SZ from density matrix

   NumericalQuality Good

   XC
      GGA Becke Perdew
   END

   UNRESTRICTED
   Relativity Level=None

   Restart
      File BChain_SZ.results/band.rkf
      scf
      useDensityMatrix true
   end

   Basis
      Type SZ
      Core Large
   End
EndEngine
eor


# ----------------------------- fifth run --------------------------

AMS_JOBNAME=BChain_TZ2P $AMSBIN/ams <<eor

Task SinglePoint

System
   Lattice [Bohr]
      4.0 0.0 0.0
   End
   Atoms [Bohr]
      B 0.0 0.0 0.0
   End
End

Engine Band
   Title B chain restart bas=TZ2P from orbitals

   NumericalQuality Good

   XC
      GGA Becke Perdew
   END
```

(continues on next page)

(continued from previous page)

```
   UNRESTRICTED
   Relativity Level=None

   Restart
      File BChain_SZ.results/band.rkf
      scf
      useDensityMatrix false
   end

   Basis
      Type TZ2P
      Core Large
   End
EndEngine

eor

# ---------------------------- sixth run -------------------------

$AMSBIN/ams <<eor

Task SinglePoint

System
   Lattice [Bohr]
      4.0 0.0 0.0
   End
   Atoms [Bohr]
      B 0.0 0.0 0.0
   End
End

Engine Band
   Title B chain restart bas=TZ2P from density matrix (bas_SZ)

   NumericalQuality Good

   XC
      GGA Becke Perdew
   END

   UNRESTRICTED
   Relativity Level=None

   Restart
      File BChain_SZ.results/band.rkf
      scf
      useDensityMatrix true
   end

   Basis
      Type TZ2P
      Core Large
   End
EndEngine
eor
```

10.5.2 Example: Restart SCF for properties calculation

Download RestartProperties.run

```
#!/bin/sh

# This example shows how to restart the SCF and compute various properties, like
# a density of states, and a band structure 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)
# ================================================

AMS_JOBNAME=ToBeRestarted $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile polyethylene.xyz
End

Engine Band
  Unrestricted True
EndEngine
eor

# ===================================
# Restart and compute some properties
# ===================================

AMS_JOBNAME=prop $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile polyethylene.xyz
End

Engine Band
  Unrestricted True

  Restart
```

(continues on next page)

(continued from previous page)

```
    SCF
    File ToBeRestarted.results/band.rkf
  End

  DOS
    CalcDOS True
  End

  BandStructure
    Enabled True
    DeltaK 0.3
    EnergyAboveFermi 10.0
  End

  EffectiveMass
    Enabled True
  End
EndEngine
eor


echo 'Extract some properties from the rkf file:'

echo "Density of States:"
$AMSBIN/amsreport prop.results/band.rkf -r 'DOS%Total DOS##1'

echo "Band curve:"
$AMSBIN/amsreport prop.results/band.rkf -r 'band_curves%Edge_1_bands##1'

echo "Fab bands:"
$AMSBIN/amsreport prop.results/band.rkf -r 'band_curves%Edge_1_fatBands##1'

echo "Effective Mass:"
$AMSBIN/amsreport prop.results/band.rkf -r 'EffectiveMass%EffectiveMasses##1'

echo 'Done extracting properties'
```

10.5.3 Example: Properties on a grid

Download BeO_tape41.run

```
#!/bin/sh

# 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 in the user documentation (Restarts).

# 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.kf. 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: ==
```

(continues on next page)

(continued from previous page)

```
AMS_JOBNAME=First $AMSBIN/ams <<eor

Task SinglePoint

System
  FractionalCoords True

  Atoms
   Be  0.              0.              0.
   Be  0.333333333333 0.333333333333 0.5
   O   0.              0.              0.375
   O   0.333333333333 0.333333333333 0.875
  END

  Lattice [Bohr]
    5.10 0              0
    2.55 4.416729559300 0
    0    0              8.328265125462
  End
End

Engine Band
  Title BeO

  NumericalQuality Basic

  xc
    GGA BP86
  end

  Basis
    Type DZ
    Core large
  end
EndEngine
eor

# == Second Job: ==

AMS_JOBNAME=Second $AMSBIN/ams <<eor

Task SinglePoint

System
  FractionalCoords True

  Atoms
   Be  0.              0.              0.
   Be  0.333333333333 0.333333333333 0.5
   O   0.              0.              0.375
   O   0.333333333333 0.333333333333 0.875
  END

  Lattice [Bohr]
    5.10 0              0
    2.55 4.416729559300 0
    0    0              8.328265125462
```

(continues on next page)

(continued from previous page)

```
  End
End

Engine Band
  Title BeO_restart

  Restart
    File First.results/band.rkf
    DensityPlot
  End

  Grid
     Type Coarse
  End

  DensityPlot
     rho(deformation/fit) ! FITDENSITY_deformation_scf
     rho(fit)             ! FITDENSITY_total_scf
     rho(atoms)           ! ATOMIC_density
     v(coulomb/atoms)     ! ATOMIC_coulombPot
     v(coulomb)           ! COULOMBPOTENTIAL_scf
     vxc[rho(fit)]        ! XCPOTENTIAL_scf
     X                    ! Electron Energy Density
     X(fit)               ! Electron Energy Density, using the fit
  End

  NumericalQuality Basic

  xc
    GGA BP86
  end

  Basis
    Type DZ
    Core large
  end
EndEngine
eor


NSCM=1
export NSCM
echo ""
echo "Begin TOC of tape41"

$AMSBIN/dmpkf -n 1 Second.results/TAPE41 --toc

echo "End TOC of tape41"
```

10.5.4 Example: DOS and BandStructure from a previous calculation

It is quite likely that you did a calculation, but forgot to request the DOS or BandStructure. Or maybe you did but want to use different settings.

Fortunately it is possible to calculate these directly from the previous calculation. The traditional way was to restart the SCF, but this is no longer needed.

With the restart of the DOS it has become possible to calculate the DOS with a finer k-space sampling. In this example it is shown that the DOS calculated this way matches well the DOS obtained from a full calculation with a dense k-grid (for both the SCF and DOS).

The example below features some concepts

- use environment variables to control in one place repeated settings
- DOS calculated on a fixed grid. This is not really needed, but it makes the comparison between different DOS results much more stable.
- `KSpace%Regular%DoubleCount` key. Useful when you want to check against a k-grid that is twice as dense.
- A text report is created, useful for automatic testing.
- –delete-old-results. Allow ams to overwrite an existing results folder.

`Download RestartDosAndBandStructure.run`

```
#!/bin/bash

# some env. variables to control settings
unr=no
k_quality=Normal
DOSLINE="DOS CalcPDos=true Energies=300 AbsoluteMaxMin=yes Min=-0.3 Max=0.2"
BANDSTTRUCTURELINE="BandStructure Enabled=true DeltaK=0.03"

cat << eor > convergence.inc
SCF Method=DIIS
Convergence Criterion=1e-8
eor

report=report.txt
echo "" > $report

base=test.unr=$unr.k_quality=$k_quality

export AMS_JOBNAME=$base

# First calculation, where we "forget" to request the PDOS and BandStructure, or did␣
↪not satisfactory settings for those.

$AMSBIN/ams --delete-old-results << eor

Task SinglePoint

System
    Atoms
        Cu     -1.014698231    -0.585836297     0.414248818
        Ni      1.014698231     0.585836297    -0.414248818
    End

    Lattice
```

(continues on next page)

(continued from previous page)

```
        4.058792924     0.000000000     0.000000000
        2.029396462     3.515017781     0.000000000
    End
End

Engine Band

Unrestricted $unr
KSpace Quality=$k_quality

@include convergence.inc

$DOSLINE

$BANDSTTRUCTURELINE

EndEngine

eor


export AMS_JOBNAME=$base.betterk

# We could have done the calculation with a better k-grid

$AMSBIN/ams --delete-old-results << eor

Task SinglePoint

System
    Atoms
        Cu     -1.014698231    -0.585836297     0.414248818
        Ni      1.014698231     0.585836297    -0.414248818
    End

    Lattice
        4.058792924     0.000000000     0.000000000
        2.029396462     3.515017781     0.000000000
    End
End

Engine Band

Unrestricted $unr

KSpace
    Quality $k_quality
    Regular DoubleCount=1  # use a twice as dense k-grid. You could also use a better␣
↪KSpace%Quality, or set a more dense grid otherwise.
End


@include convergence.inc

$DOSLINE

$BANDSTTRUCTURELINE
```

(continues on next page)

(continued from previous page)

```
EndEngine

eor


$AMSBIN/amspython $AMSHOME/scripting/tools/comparekf.py --dot-product  $base.results/
↪band.rkf  $base.betterk.results/band.rkf "DOS%Total DOS" >> $report


# We can request the dos and band structure simply restarting the SCF.

export AMS_JOBNAME=$base.restartscf

$AMSBIN/ams --delete-old-results << eor

Task SinglePoint

LoadSystem
  File $base.results/ams.rkf
End

Engine Band

Unrestricted $unr
KSpace Quality=$k_quality

Restart
    File $base.results/band.rkf
    SCF true
End

@include convergence.inc

$DOSLINE

$BANDSTTRUCTURELINE

EndEngine

eor


# We can also request both the DOS and BandStructure in a restart, avoiding␣
↪restarting the SCF

export AMS_JOBNAME=$base.restartdos

$AMSBIN/ams --delete-old-results << eor

Task SinglePoint

LoadSystem
  File $base.results/ams.rkf
End

Engine Band
```

(continues on next page)

(continued from previous page)

```
Unrestricted $unr
KSpace Quality=$k_quality

Restart
    File $base.restartscf.results/band.rkf      #  can as well be restarted from
↪$base.results/band.rkf.  this restart used for the most precise comparison
    Dos true
    BandStructure true
End

@include convergence.inc

$DOSLINE

$BANDSTTRUCTURELINE

EndEngine

eor

$AMSBIN/amspython $AMSHOME/scripting/tools/comparekf.py --dot-product  $base.results/
↪band.rkf  $base.restartscf.results/band.rkf "DOS%Total DOS" >> $report
$AMSBIN/amspython $AMSHOME/scripting/tools/comparekf.py --dot-product  $base.results/
↪band.rkf  $base.restartdos.results/band.rkf "DOS%Total DOS" >> $report
$AMSBIN/amspython $AMSHOME/scripting/tools/comparekf.py --dot-product  $base.
↪restartscf.results/band.rkf  $base.restartdos.results/band.rkf "DOS%Total DOS" >>
↪$report


# We can even use a better k-space sampling which improves the DOS. BandStructure not␣
↪affected.

export AMS_JOBNAME=$base.restartdosbetterk

$AMSBIN/ams --delete-old-results << eor

Task SinglePoint

LoadSystem
  File $base.results/ams.rkf
End

Engine Band

Unrestricted $unr

KSpace
    Quality $k_quality
    Regular DoubleCount=1  # use a twice as dense k-grid. You could also use a better␣
↪KSpace%Quality, or set a more dense grid otherwise.
End

Restart
    File $base.results/band.rkf
    Dos true
    BandStructure true
```

(continues on next page)

(continued from previous page)

```
End

@include convergence.inc

$DOSLINE

$BANDSTTRUCTURELINE

EndEngine

eor


$AMSBIN/amspython $AMSHOME/scripting/tools/comparekf.py --dot-product  $base.betterk.
↪results/band.rkf  $base.restartdosbetterk.results/band.rkf "DOS%Total DOS" >>
↪$report


echo "begin report"
cat $report
echo "end report"
```

10.6 NEGF

10.6.1 Example: Main NEGF flavors

Download NEGF_Cr_wire.run

```
#! /bin/sh

# This example shows how to use the NEGF functionality.
# Note: Setting up a NEGF calculation is quite hard without the GUI.

# 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.

# ============================================================================
#  Method #1: non-self consistent NEGF (uses the conductance program, like DFTB)
# ============================================================================


# =========================   Method #1. Run #1   ==============================

AMS_JOBNAME=lead_1 $AMSBIN/ams <<eor

Task SinglePoint

System
```

(continues on next page)

(continued from previous page)

```
  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
End

Engine Band
  Title method_1_run_1

  KSpace
      Quality Good
  End

  NumericalQuality Basic

  SCF Method=DIIS

  Basis
     Type DZ
     Core Large
  End

  Unrestricted

  StoreHamiltonian2
EndEngine
eor

# ========================   Method #1. Run #2   ================================

AMS_JOBNAME=scattering_1 $AMSBIN/ams <<eor

Task SinglePoint

System
  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
End

Engine Band
```

(continues on next page)

(continued from previous page)

```
  Title method_1_run_2

  NumericalQuality Basic

  SCF Method=DIIS

  Basis
     Type DZ
     Core Large
  End

  Unrestricted

  StoreHamiltonian2
  StoreHamAsMol
EndEngine

eor

# ==========================   Method #1. Run #3   ============================

$AMSBIN/conductance <<EOF
EnergyGrid min=-5 max=5 num=200
Files
   Leads      lead_1.results/band.rkf
   Scattering scattering_1.results/band.rkf
End

EOF

# Copy the content of the "results" section from ConductanceResults.kf to band.rkf␣
↪and rename the section to NEGF
$AMSBIN/cpkf "ConductanceResults.kf" "scattering_1.results/band.rkf" "results --
↪rename NEGF"


echo "Extract transmisstion from rkf file (Method 1)"
$AMSBIN/amsreport scattering_1.results/band.rkf -r "NEGF%transmission#12.5f##1"

# ============================================================================
#               Method #2: self consistent NEGF without alignment
# ============================================================================


# ==========================   Method #2. Run #1   ============================

AMS_JOBNAME=lead_2 $AMSBIN/ams <<eor

Task SinglePoint

System
  ATOMS
      Cr.1 0.0 0.0 0.0
      Cr.2 2.5 0.0 0.0
  END

  Lattice
```

(continues on next page)

(continued from previous page)

```
   5.0 0.0 0.0
  End
End

Engine Band
  Title method_2_run_1

  KSpace
      Quality Good
  End

  NumericalQuality Basic

  SCF Method=DIIS

  Basis
    Type DZ
    Core Large
  End

  Unrestricted
  StoreHamiltonian2
EndEngine

eor

# ============================   Method #2. Run #2   ========================


$AMSBIN/sgf   <<eor
TITLE Test for NEGF inputs
SAVE SIGMA
SURFACEGF
   RKFFileName lead_2.results/band.rkf
   SCMCode
   KT 0.001
   ContourQuality normal
END
eor

mv SigmaSCM Sigma.kf

# ============================   Method #2. Run #3   ========================

AMS_JOBNAME=scattering_2 $AMSBIN/ams <<eor

Task SinglePoint

System
  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
```

(continues on next page)

(continued from previous page)

```
      Cr.1R 7.5 0.0 0.0
      Cr.2R 10.0 0.0 0.0
  END
End

Engine Band
  Title method_2_run_3

  NumericalQuality Basic

  SCF Method=DIIS

  Basis
    Type DZ
    Core Large
  End

  Unrestricted
  NEGF
    LeadFile        lead_2.results/band.rkf
    SGFFile         Sigma.kf
    ApplyShift2     False
    ContourQuality  normal
    EMin           -5
    EMax            5
    NE 200
  End
EndEngine
eor

echo "Extract transmisstion from rkf file (Method 2)"
$AMSBIN/amsreport scattering_2.results/band.rkf -r "NEGF%transmission#12.5f##1"

# ============================================================================
#                Method #3: self consistent NEGF wit alignment run
# ============================================================================

# ======================   Method #3. Run #1   ================================


AMS_JOBNAME=lead_3 $AMSBIN/ams <<eor

Task SinglePoint

System
  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
End

Engine Band
  Title method_3_run_1
```

(continues on next page)

(continued from previous page)

```
  KSpace
      Quality Good
  End

  NumericalQuality Basic

  SCF Method=DIIS

  Basis
    Type DZ
    Core Large
  End

  Unrestricted
  StoreHamiltonian2
EndEngine
eor

# ============================   Method #3. Run #2   ============================


$AMSBIN/sgf   <<eor
TITLE Test for NEGF inputs
SAVE SIGMA
SURFACEGF
   RKFFileName lead_3.results/band.rkf
   SCMCode
   KT 0.001
   ContourQuality normal
END
eor

mv SigmaSCM Sigma.kf

# ============================   Method #3. Run #3   ============================

AMS_JOBNAME=align $AMSBIN/ams <<eor

Task SinglePoint

System
  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
End

Engine Band
  Title method_3_run_3

  NumericalQuality Basic
```

(continues on next page)

(continued from previous page)

```
  SCF Method=DIIS

  Basis
    Type DZ
    Core Large
  End

  Unrestricted
  NEGF
    DoAlignment True
    LeadFile lead_3.results/band.rkf
    SGFFile Sigma.kf
    ContourQuality normal
    EMin -5.0
    EMax 5.0
    NE 200
    AlignChargeTol 0.0001
  End
EndEngine
eor

# ============================   Method #3. Run #4  ============================

AMS_JOBNAME=scattering_3 $AMSBIN/ams <<eor

Task SinglePoint

System
  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
End

engine Band
  Title method_3_run_4

  NumericalQuality Basic

  SCF Method=DIIS

  Basis
    Type DZ
    Core Large
  End

  Unrestricted

  NEGF
```

(continues on next page)

(continued from previous page)

```
    LeadFile lead_3.results/band.rkf
    SGFFile Sigma.kf
    AlignmentFile align.results/band.rkf
    ContourQuality normal
    EMin -5.0
    EMax 5.0
    NE 200
  End
EndEngine
eor

echo "Extract transmisstion from rkf file (Method 2)"
$AMSBIN/amsreport scattering_3.results/band.rkf -r "NEGF%transmission#12.5f##1"
```

10.6.2 Example: NEGF with bias

Download NEGF_bias.run

```
#! /bin/sh

# This example shows how to use the NEGF 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 is used.
# The current is appended to a text file, which one could plot eg. with gnuplot.

# Note: Setting up a NEGF calculation is quite hard without the GUI.

AMS_JOBNAME=tight-binding $AMSBIN/ams <<eor

Task SinglePoint

System
  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
End

Engine Band
  TITLE tight-binding

  KSpace
    Quality VeryGood
  End

  SoftConfinement
    Quality Basic
  End

  Basis
```

(continues on next page)

(continued from previous page)

```
    Type DZ
    Core Large
  End

  StoreHamiltonian2
EndEngine
eor


$AMSBIN/sgf   <<eor
TITLE Test for NEGF inputs
SAVE SIGMA
SURFACEGF
   RKFFileName tight-binding.results/band.rkf
   SCMCode
   KT 0.001
   ContourQuality normal
END
eor

mv SigmaSCM Sigma.kf

REPORT=Li-CuAg.report
touch $REPORT

for bias in -0.01 0.01
do


AMS_JOBNAME=negf $AMSBIN/ams <<eor

Task SinglePoint

System
  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
End

Engine Band
  TITLE bias=$bias

  SoftConfinement
    Quality Basic
  End
```

(continues on next page)

(continued from previous page)

```
  Basis
    Type DZ
    Core Large
  End

  NEGF
    LeadFile tight-binding.results/band.rkf
    SGFFile Sigma.kf
    EMin -5.0
    EMax 5.0
    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
EndEngine

eor

current=`$AMSBIN/amsreport negf.results/band.rkf 'NEGF%current'`
echo "NEGFREPORT: Bias=$bias, Current=$current" >> $REPORT

echo "start of transmission (bias=$bias)"
cat Transmission_*.plt
echo "end of transmission"

rm Transmission_*.plt

rm -r negf.results

done

echo "Start of report"
cat $REPORT
echo "End of report"
```

10.6.3 Example: NEGF using the non-self consistent method

Download NEGF_Conductance.run

```
#! /bin/sh

# In this example we demonstrate how to run a Band-NEGF calculation using the non
# self consistent approach (using the conductance program). In the first example
# we study the conductivity of a mono-atomic gold chain with a CO molecule
# adsorbed on top. Such calculation consists of three separate runs. See the
# documentation for more details.

# ============================================================================
#                              CO on gold chain
# ============================================================================

# =======
# Au lead
# =======

AMS_JOBNAME=Au_lead $AMSBIN/ams <<eor

Task SinglePoint

System
    ATOMS
        Au.1 0.0       0.0  0.0
        Au.2 2.884996  0.0  0.0
        Au.3 5.769992  0.0  0.0
    END

    Lattice
        8.654988  0.0  0.0
    End
End

Engine Band
    TITLE Au_lead

    KSpace
        Quality VeryGood
    End

    SoftConfinement
        Quality Basic
    End

    Basis
        Type DZ
        Core Large
    End

    StoreHamiltonian2
EndEngine
eor

# =============
# Au scattering
```

(continues on next page)

(continued from previous page)

```
# =============

AMS_JOBNAME=Au_scattering $AMSBIN/ams <<eor

Task SinglePoint

System
    ATOMS
        Au.1L -20.194972  0.0  0.0
        Au.2L -17.309976  0.0  0.0
        Au.3L -14.42498   0.0  0.0
        Au.C  -11.539984  0.0  0.0
        Au.C  -8.654988   0.0  0.0
        Au.C  -5.769992   0.0  0.0
        Au.C  -2.884996   0.0  0.0
        Au.C   0.0        0.0  0.20
        Au.C   2.884996   0.0  0.0
        Au.C   5.769992   0.0  0.0
        Au.C   8.654988   0.0  0.0
        Au.C   11.539984  0.0  0.0
        O.C    0.0        0.0  3.12
        C.C    0.0        0.0  1.96
        Au.1R  14.42498   0.0  0.0
        Au.2R  17.309976  0.0  0.0
        Au.3R  20.194972  0.0  0.0
    END

    Lattice
        43.27494  0.0  0.0
    End
End

Engine Band
    TITLE Au_scattering

    SoftConfinement
        Quality Basic
    End

    Basis
        Type DZ
        Core Large
    End

    StoreHamiltonian2
    StoreHamAsMol
EndEngine

eor

# ==============
# Au Conductance
# ==============

$AMSBIN/conductance <<EOF
EnergyGrid min=-3.5 max=3 num=200
Files
```

(continues on next page)

(continued from previous page)

```
  Leads         Au_lead.results/band.rkf
  Scattering    Au_scattering.results/band.rkf
End
EOF

mv ConductanceResults.kf Au_ConductanceResults.kf


echo "Extract DOS from the kf file (AuCO):"
$AMSBIN/amsreport Au_ConductanceResults.kf -r "results%dos#12.5f##1"

echo "Extract the transmission from the kf file (AuCO):"
$AMSBIN/amsreport Au_ConductanceResults.kf -r "results%transmission#12.5f##1"


# ============================================================================
#                           Spin-unrestricted Cr chain
# ============================================================================

# =======
# Cr Lead
# =======

AMS_JOBNAME=Cr_lead $AMSBIN/ams <<eor

Task SinglePoint

System
    ATOMS
        Cr.1 1.18995235  0.0  0.0
        Cr.2 4.00745359  0.0  0.0
        Cr.3 6.82495483  0.0  0.0
    END

    Lattice
        8.45250372  0.0  0.0
    End
End

Engine Band
    TITLE Cr_lead

    KSpace
        Quality VeryGood
    End

    SoftConfinement
        Quality Basic
    End

    Basis
        Type DZ
        Core Large
    End

    UNRESTRICTED
```

(continues on next page)

(continued from previous page)

```
    StoreHamiltonian2
EndEngine

eor

# =============
# Cr Scattering
# =============

AMS_JOBNAME=Cr_scattering $AMSBIN/ams <<eor

Task SinglePoint

System
    ATOMS
        Cr.1L -10.08005261  0.0  0.0
        Cr.2L  -7.26255137  0.0  0.0
        Cr.3L  -4.44505013  0.0  0.0
        Cr.C   -1.62754889  0.0  0.0
        Cr.C    1.18995235  0.0  0.0
        Cr.C    4.00745359  0.0  0.0
        Cr.1R   6.82495483  0.0  0.0
        Cr.2R   9.64245607  0.0  0.0
        Cr.3R  12.45995731  0.0  0.0
    END

    Lattice
        25.35751116  0.0  0.0
    End
End

Engine Band
    TITLE Cr_scattering

    KSpace
        Quality Good
    End

    SoftConfinement
        Quality Basic
    End

    Basis
        Type DZ
        Core Large
    End

    UNRESTRICTED
    StoreHamiltonian2
    StoreHamAsMol
EndEngine

eor

# ==============
# Cr Conductance
# ==============
```

(continues on next page)

(continued from previous page)

```
$AMSBIN/conductance <<EOF
EnergyGrid min=-4 max=4 num=200
Files
  Leads      Cr_lead.results/band.rkf
  Scattering Cr_scattering.results/band.rkf
End

EOF

mv ConductanceResults.kf Cr_ConductanceResults.kf

echo "Extract DOS from the kf file (Cr):"
$AMSBIN/amsreport Cr_ConductanceResults.kf -r "results%dos#12.5f##1"

echo "Extract the transmission from the kf file (Cr):"
$AMSBIN/amsreport Cr_ConductanceResults.kf -r "results%transmission#12.5f##1"
```

10.7 Structure and Reactivity

10.7.1 Example: NaCl: Bulk Crystal

Download NaCl.run

```
#! /bin/sh

# 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 input line FractionalCoords True means that atomic positions are input as
# coefficients in terms of the lattice vectors, rather than as absolute
# (Cartesian) coordinate values.

AMS_JOBNAME=NaCl $AMSBIN/ams <<eor

Task SinglePoint

System
  FractionalCoords True

  Atoms
    Na 0.0 0.0 0.0
    Cl 0.5 0.5 0.5
  End

  Lattice
    0.0   2.75   2.75
    2.75  0.0    2.75
    2.75  2.75   0.0
  End
End
```

(continues on next page)

(continued from previous page)

```
Engine Band
  Title NaCl

  Kspace
    Symmetric KInteg=3
  End

  Basis
    Type SZ
    Core None
  End

  Print AtomicChargesDetails

EndEngine

eor

# 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).

AMS_JOBNAME=NaCl-restart $AMSBIN/ams <<eor

Task SinglePoint

LoadSystem
  File NaCl.results/ams.rkf
  Section InputMolecule
End

Engine Band
  Title NaCl DOS analysis (restart)

  Kspace
    Symmetric KInteg=3
  End

  Basis
    Type SZ
    Core None
  End

  Restart
    File NaCl.results/band.rkf
    SCF
  End

  DOS
```

(continues on next page)

(continued from previous page)

```
    CalcPDos  True
    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

  Print AtomicChargesDetails
EndEngine
eor

# Finally, we copy the contents of the DOS result file to standard output

echo ""
echo Contents of DOS file
cat NaCl.dos
echo "The End"
```

10.7.2 Example: Transition-State search using initial Hessian

Download COChainFreqTS.run

```
#! /bin/sh

# This example demonstrates in the first step how to calculate the Hessian.
# The second run uses the pre-calculated Hessian and performs a transition
# state search along the frequency mode with the smallest frequency.


# First run: Calculate Hessian
# ============================

AMS_JOBNAME=hessian $AMSBIN/ams << EOF
```

(continues on next page)

(continued from previous page)

```
   Task SinglePoint

   Properties
      Hessian True
   End

   System
      Atoms
         C  0.0  0.0  0.0
         O  1.5  0.5  0.0
      End
      Lattice
         3.2  0.0  0.0
      End
   End

   Engine Band
      Basis Type=DZP
      KSpace Quality=Good
   EndEngine

EOF


# Second run: TS search with initial Hessian
# ==========================================

AMS_JOBNAME=TS $AMSBIN/ams << EOF

   Task TransitionStateSearch

   System
      Atoms
         C  0.0  0.0  0.0
         O  1.5  0.5  0.0
      End
      Lattice
         3.2  0.0  0.0
      End
   End

   GeometryOptimization
      Convergence Gradients=1.0e-4
      InitialHessian
         # Load the pre-calculated Hessian as the initial Hessian for the
         # transition state search using the Quasi-Newton based optimizer.
         Type FromFile
         File hessian.results/band.rkf
      End
   End

   Properties
      # Also calculate normal modes in the end, so we can see if we actually
      # found a transition state.
      NormalModes True
   End
```

(continues on next page)

(continued from previous page)

```
   Engine Band
      Basis Type=DZP
      KSpace Quality=Good
   EndEngine

EOF
```

10.7.3 Example: Atomic energies

Download H_ref.run

```
#! /bin/sh

# 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

# XYZ file of H atom with large 2 lattice
cat << eor > H.xyz
1

H 0.0 0.0 0.0
VEC1 10.583544212 0.0          0.0
VEC2 0.0          10.583544212 0.0
eor


$AMSBIN/ams <<eor

Task SinglePoint

System
   GeometryFile H.xyz
End

Engine Band
   Title Formation energy of the H-atom w.r.t. spherical atom

   Print AtomicChargesDetails

   Kspace
      Symmetric KInteg=5
   End
   Integration
      Accint 5.0
   End

   Convergence
```

(continues on next page)

(continued from previous page)

```
      Criterion 1E-6
   End

   AtomType H
      Dirac H
        1  0
      VALENCE
         1S 1
      End

      BasisFunctions
         1S   1.58
         2P   1.0
      End

      FitFunctions
      End
   End
EndEngine
eor

rm -r ams.results

$AMSBIN/ams <<eor

Task SinglePoint

System
   GeometryFile H.xyz
End

Engine Band
   Title Spin polarization energy of the H-atom w.r.t. spherical atom

   Print AtomicChargesDetails

   Kspace
      Symmetric KInteg=5
   End
   Integration
      Accint 5.0
   End

   Convergence
      Criterion 1E-6
   End

   Unrestricted

   AtomType H
      Dirac H
        1  0
      VALENCE
         1S 1
      End

      BasisFunctions
```

(continues on next page)

(continued from previous page)

```
         1S   1.58
         2P   1.0
      End

      FitFunctions
      End
   End
EndEngine
eor

rm -r ams.results


$AMSBIN/ams <<eor

Task SinglePoint

System
   GeometryFile H.xyz
End

Engine Band
   Title Spin polarization (relativistic) energy of the H-atom w.r.t. spherical atom

   Print AtomicChargesDetails

   Kspace
      Symmetric KInteg=5
   End
   Integration
      Accint 5.0
   End

   Convergence
      Criterion 1E-6
   End

   Unrestricted

   Relativity
      Level Scalar
   End

   AtomType H
      Dirac H
        1  0
      VALENCE
         1S 1
      End

      BasisFunctions
         1S   1.58
         2P   1.0
      End

      FitFunctions
      End
```

(continues on next page)

(continued from previous page)

```
   End
EndEngine
eor

rm -r ams.results


$AMSBIN/ams <<eor

Task SinglePoint

System
   GeometryFile H.xyz
End

Engine Band
   Title Spin polarization energy of the H-atom w.r.t. spin unrestricted atom

   Print AtomicChargesDetails

   Kspace
      Symmetric KInteg=5
   End
   Integration
      Accint 5.0
   End

   Convergence
      Criterion 1E-6
   End

   Unrestricted
   UnrestrictedReference

   AtomType H
      Dirac H
        1  0
      VALENCE
         1S 1
      End

      BasisFunctions
         1S   1.58
         2P   1.0
      End

      FitFunctions
      End
   End
EndEngine
eor

rm -r ams.results


$AMSBIN/ams <<eor
```

(continues on next page)

(continued from previous page)

```
Task SinglePoint

System
   GeometryFile H.xyz
End

Engine Band
   Title Spin polarization (relativistic) energy of the H-atom w.r.t. spin␣
↪unrestricted atom

   Print AtomicChargesDetails

   Kspace
      Symmetric KInteg=5
   End
   Integration
      Accint 5.0
   End

   Convergence
      Criterion 1E-6
   End

   UnrestrictedReference
   Unrestricted

   Relativity
      Level Scalar
   End

   AtomType H
      Dirac H
        1  0
      VALENCE
         1S 1
      End

      BasisFunctions
         1S   1.58
         2P   1.0
      End

      FitFunctions
      End
   End
EndEngine
eor
```

10.7.4 Example: Calculating the atomic forces

Download BNForce.run

```
#!/bin/sh

# This example shows how to calculate the gradient of the energy with respect to
# nuclear displacements, by requesting Properties => Gradients Yes

$AMSBIN/ams <<eor

Task SinglePoint

System
   Atoms
    B   0.0     0.0     0.0
    N   0.86544 0.86544 0.86544
   end
   Lattice
     0.0 1.8 1.8
     1.8 0.0 1.8
     1.8 1.8 0.0
   End
End

Properties
  Gradients Yes
End

Engine Band
   Title BN zincblende structure (force calculation)

   NumericalQuality Basic  ! for speed, not very accurate

   Basis
      Type TZ2P
      Core Large
   End

   Relativity
      Level None
   End
EndEngine

eor
```

10.7.5 Example: Optimizing the geometry

Download H2BulkGeo.run

```
#! /bin/sh

# This example shows how to optimize the geometry.

# This example consists of two runs. The first run performs 5 iterations
# regarding the geometry optimization. And the second run exploits the
```

(continues on next page)

(continued from previous page)

```
# possibility to restart a geometry optimization based on the rkf of a
# previous, presumably non-converged run.

# ---------------------------- first run -------------------------

AMS_JOBNAME=First $AMSBIN/ams <<eor

Task GeometryOptimization

System
  ATOMS [Bohr]
    H   0.0 0.0 0.0
    H   1.0 0.0 0.0
  End

  Lattice [Bohr]
    5.0      0        0
    0        5.0      0
    0        0        5.0
  End
End

GeometryOptimization
  MaxIterations 5
  Convergence Gradients=1e-6 Step=1.0e-3
End

Engine Band
  Basis
    Type DZP
  End
EndEngine

eor


# In the next run we use the result file to continue the geometry optimization.

# ---------------------------- second run -------------------------

$AMSBIN/ams <<eor

Task GeometryOptimization

LoadSystem
  File First.results/ams.rkf
End

GeometryOptimization
  MaxIterations 5
  Convergence Gradients=1e-6 Step=1.0e-3
End

Engine Band
  Basis
     Type DZP
  End
```

(continues on next page)

(continued from previous page)

```
EndEngine
eor


echo 'Extract optimized geometry from the rkf file'
$AMSBIN/amsreport ams.results/ams.rkf -r 'Molecule%Coords##3'

echo 'Extract number of steps from the rkf file'
$AMSBIN/amsreport ams.results/ams.rkf -r 'History%nEntries'
```

10.8 Time dependent DFT

10.8.1 Example: TD-CDFT for MoS2 Monolayer (NewResponse)

Download NewResp_2DMoS2Restart.run

```
#!/bin/sh

# 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) 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.

# =========================
# MoS2 Monolayer .xyz file:
# =========================

cat << eor > MoS2_2D_1L.xyz
3

S          0.00000000       0.00300000      -7.76123300
S          0.00000000       0.00000000      -4.53876700
Mo         1.58000000       0.91221300      -6.15000000
VEC1       3.16000000       0.00000000       0.00000000
VEC2       1.58000000       2.73664028       0.00000000

eor

# ================================================
# Simple single point calculation (no properties)
# ================================================

AMS_JOBNAME=MoS2 $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile MoS2_2D_1L.xyz
End
```

(continues on next page)

(continued from previous page)

```
Engine Band
  UseSymmetry False

  NumericalQuality good

  DEPENDENCY BASIS=1e-10

  Tails bas=1e-10

  KSpace
    Regular
      NumberOfPoints 5 5
    End
  End

  Basis
    Type DZP
    Core Large
  End

  Convergence
   Criterion 1E-8
  End
EndEngine

eor

# ==================================
# Restart and compute some properties
# ==================================

# Caution!
# One has to make sure to use the same
# Symmetry/NumericalQuality/KSpace/Basis/ZORA/... options for the
# ground state calculation and for the restart calculation! Otherwise a normal
# ground state SCF optimization will be performed in the restart calculation.


AMS_JOBNAME=MoS2_restart $AMSBIN/ams <<eor

  Task SinglePoint

  System
    GeometryFile MoS2_2D_1L.xyz
  End

  Engine Band

    UseSymmetry False

    NumericalQuality good

    Tails bas=1e-10

    KSpace
      Regular
```

(continues on next page)

(continued from previous page)

```
      NumberOfPoints 5 5
    End
  End

  Basis
    Type DZP
    Core Large
  End

  Convergence
   Criterion 1E-8
  End

  Restart
    File MoS2.results/band.rkf
    SCF
  End

  NewResponse
    nFreq        3
    FreqLow      2.0
    FreqHigh     2.7
    ActiveESpace 10.0
    ActiveXYZ    T T F
  End

  NewResponseSCF
    nCycle       50
    Criterion    1E-3
    DIIS  MinSamples=3 MixingFactor=0.5 MaximumCoefficient=20
  End

  NewResponseKSPACE
    subsimp 10
    eta     1e-6
  End
EndEngine

eor

# ============
# Extract info
# ============

$AMSBIN/amsreport MoS2_restart.results/band.rkf RESPDIELRE
$AMSBIN/amsreport MoS2_restart.results/band.rkf RESPDIELIM

# The results are accessible via the standard output or via the prop.kf file.
# For the latter, one can use the AMSreport command $AMSBIN/amsreport prop.kf
# RESPDIELRE and $AMSBIN/amsreport prop.kf 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:

# Real part
# Frequency (au)  epsilon_1(XX)  epsilon_1(YY)  epsilon_1(ZZ)
# 0.0735          8.1622063      8.1788067      1.8845925
```

(continues on next page)

(continued from previous page)

```
# 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

# 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 AMSspectra. Just type: $AMSBIN/amsspectra prop.kf
```

10.8.2 Example: TD-CDFT for Copper (NewResponse)

Download NewResp_3DCopper.run

```
#! /bin/sh

$AMSBIN/ams <<eor

Task SinglePoint

System
   Lattice :: FCC
      0     1.805 1.805
      1.805 0     1.805
      1.805 1.805 0
   End

   Atoms
      Cu  0.00  0.00  0.00
   End
End

Engine Band
   Title NewResponse of Cu within ALDA

   NumericalQuality basic
```

(continues on next page)

(continued from previous page)

```
   KSpace
     Regular
       NumberOfPoints 5 5 5
     End
   End

   NewResponse
      nfreq         10
      freqLow       0.1
      freqHigh      10.0
      activeEspace  10
   END

   NewResponseSCF
      Criterion       0.1
      LowFreqAlgo     true
      COApproach      true
      COApproachBoost true
   End

   NewResponseKSPACE
      subsimp       5
   End

   Basis
      Type TZ2P
      Core Large
   End
EndEngine

eor
```

10.8.3 Example: TDCDFT: Plot induced density (NewResponse)

Download NewResp_PlotInducedDensity.run

```
#!/bin/sh

AMS_JOBNAME=polyethylene $AMSBIN/ams <<eor

  Task SinglePoint

  System
    Lattice
      2.553395923     0.000000000     0.000000000
    end

    Atoms
       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
    end
```

(continues on next page)

(continued from previous page)

```
  End

  Engine Band

    Title Polyethylene

    KSPACE
       Regular
        NumberOfPoints 11
      End
    End

    NumericalQuality basic

    DEPENDENCY BASIS=1e-10

    Tails bas=1e-10

    NEWRESPONSE
       nFreq        10
       FreqLow      6.0
       FreqHigh     8.0
       ActiveXYZ    T F F
       ActiveESpace 2.0
    END

    Relativity Level=None

    NEWRESPONSESCF
       nCycle     50
       DIIS MixingFactor=0.075
       Criterion  0.01
    End

    Basis
      Type  TZP
      Core       small
    End
  EndEngine
eor

# ======================================
# Restart and compute Induced Densities
# ======================================

export NSCM=1
$AMSBIN/ams -n 1 <<EOF

  Task SinglePoint

  LoadSystem
    File polyethylene.results/ams.rkf
    Section InputMolecule
  End

  Engine Band
    Title Polyethylene Plot Induced Response Density
```

(continues on next page)

(continued from previous page)

```
    UseSymmetry False

    NumericalQuality basic

    DEPENDENCY BASIS=1e-10

    Tails bas=1e-10

    KSpace
      Regular
        NumberOfPoints 11
      End
    End

    Basis
      Type TZP
      Core Small
    End

    Restart
      File polyethylene.results/band.rkf
      ResponseInducedDensityPlot
    End

    ResponseInducedDensityPlot
     xcomponent 1 2
     xcomponent 5
    End

    Relativity Level=None

    Grid
    End

    debug BlockPropertyModule
  EndEngine
EOF

echo ""
echo "Begin TOC of tape41"
export NSCM=1
$AMSBIN/pkf -n 1 ams.results/FILE_BLOCKPROPERTIES
echo "End TOC of tape41"
```

10.8.4 Example: TD-CDFT for bulk diamond (OldResponse)

Download OldResp_Diamond.run

```
#! /bin/sh

# Response calculation for diamond

$AMSBIN/ams <<eor
```

(continues on next page)

(continued from previous page)

```
Task SinglePoint

System
   LATTICE
      0      1.785  1.785
      1.785  0      1.785
      1.785  1.785  0
   END
   ATOMS
      C  0.0   0.0   0.0
      C  0.8925  0.8925  0.8925
   END
End

Engine Band
   TITLE DIAMOND

   Integration
      Accint 5
   End

   KSPACE
      Symmetric KInteg=2
   End

   Dependency  Basis=1.e-6

   OLDRESPONSE
      Enabled  True
      nfreq    7
      strtfr   0.0
      endfr    19.0480
   END

   Basis
      Type DZ
   End
EndEngine

eor
```

10.9 Spectroscopy

10.9.1 Example: Hyperfine A-tensor

Download TiF3a.run

```
#!/bin/sh

# Example for an ESR A-tensor calculation.

# Be aware that the calculation must be spin unrestricted and the ATENSOR
# keyword must be present, too.
```

(continues on next page)

(continued from previous page)

```
$AMSBIN/ams <<eor

Task SinglePoint

System
   Atoms
      Ti  0.0     0.0               0.0
      F   1.78    0.0               0.0
      F  -0.89    1.541525218736    0.0
      F  -0.89   -1.541525218736    0.0
   end
End

Engine Band
   Title TiF3

   Unrestricted True

   EFG
      Enabled True
   End

   ATensor
      Enabled True
   End

   Basis
      Type TZP
      Core None
   End
EndEngine
eor
```

10.9.2 Example: Zeeman g-tensor

Download TiF3g.run

```
#!/bin/sh

# Example for an ESR g-tensor calculation. More information in the documentation
# of ESR

# Be aware that this calculation must include the spin-orbit, relativistic
# approximation (Relativistic ZORA Spin)!

$AMSBIN/ams <<eor

Task SinglePoint

System
   Atoms
      Ti  0.0              0.0   0.0
      F   0.0              1.78  0.0
      F  -1.541525218736  -0.89  0.0
```

(continues on next page)

(continued from previous page)

```
      F   1.541525218736  -0.89  0.0
   end
End

Engine Band
   Title TiF3

   Relativity
      Level Spin-Orbit
   End

   ESR
      Enabled True
   end

   Basis
      Type DZ
      Core None
   End
EndEngine

eor
```

10.9.3 Example: NMR

Download PE-NMR.run

```
#! /bin/sh

# With the NMR 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 Basis key block
# to NONE.

$AMSBIN/ams <<eor

Task SinglePoint

System
   Atoms
      C      -0.638348981     0.000000000     0.424302756
      C       0.638348981     0.000000000    -0.424302756
      H      -0.638348981     0.889974570     1.053624029
      H      -0.638348981    -0.889974570     1.053624029
      H       0.638348981     0.889974570    -1.053624029
      H       0.638348981    -0.889974570    -1.053624029
   End

   Lattice
      2.553395923     0.000000000     0.000000000
   end
End
```

(continues on next page)

(continued from previous page)

```
Engine Band
   NMR
      Enabled True
      nmratom 1
      ms0 1.
   end

   XC
      GGA Always Becke Perdew
   end

   Dependency
      Basis 1e-10
   End

   Kspace
      Symmetric KInteg=3
   End
   Integration
      Accint 5
   End

   Relativity Level=None

   Basis
      Type TZ2P
      Core NONE
   End
EndEngine

eor
```

10.9.4 Example: EFG

Download SnO_EFG.run

```
#!/bin/sh

# The calculation of the electric field gradient is invoked by the EFG key
# block.

# Since Sn is quite an heavy atom we use the scalar relativistic option.

$AMSBIN/ams <<eor

Task SinglePoint

System
   FractionalCoords True

   Lattice
      3.8029  0.0  0.0
      0.0  3.8029  0.0
      0.0  0.0  4.8382
   End
```

(continues on next page)

(continued from previous page)

```
   Atoms
      O   0.0  0.0  0.0
      O   0.5  0.5  0.0
      Sn  0.0  0.5  0.2369
      Sn  0.5  0.0 -0.2369
   End
End

Engine Band
   Title SnO EFG

   NumericalQuality Basic ! Only for speed

   Tails bas=1e-8 ! Only for reproducibility with nr. of cores

   ! useful for Moessbauer spectroscopy: density and coulomb pot. at nuclei
   PropertiesAtNuclei
   End

   EFG
      Enabled True
   End

   Basis
      Type DZ
      Core none
   End
EndEngine
eor
```

10.9.5 Example: Phonons

Download GraphenePhonons.run

```
#!/bin/sh

# 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.

echo "Geometry optimization"

AMS_JOBNAME=GO $AMSBIN/ams <<eor

System
  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
```

(continues on next page)

(continued from previous page)

```
End

Task GeometryOptimization

GeometryOptimization
  OptimizeLattice true
  Convergence Gradients=1e-5
  Method Quasi-Newton
End

Engine Band
  Title Graphene geometry optimization

  ! For Graphene we need to use a symmetric grid
  KSpace
    Symmetric KInteg=5
    Type Symmetric
  End

  StrainDerivatives
    Analytical false
  End

  Basis
    Type DZ
  end
EndEngine
eor

# In the second calculation we use the pre-optimized geometry. (See details of
# the Restart key block) Then we define a supercell and perform a phonon run by
# using Task and Phonons keys. Note that KSpace can be chosen
# a bit lower, since we now have a bigger unit cell.

echo "Phonon calculation"

AMS_JOBNAME=Phonons $AMSBIN/ams <<eor

LoadSystem
  File GO.results/ams.rkf
End

Task SinglePoint

Properties
  Phonons True
End

NumericalPhonons
  stepSize 0.0913
  SuperCell
    2 0
    0 2
  End
end

Engine Band
```

(continues on next page)

(continued from previous page)

```
  Title Graphene phonon calc

  KSpace
    Symmetric KInteg=3
    Type Symmetric
  End

  Basis
    Type DZ
  end
EndEngine

eor

NSCM=1
export NSCM
echo ""
echo "Begin TOC"

$AMSBIN/dmpkf -n 1 Phonons.results/band.rkf --toc

echo "End TOC"
```

10.10 GW

10.10.1 Example: eigenvalue-only self-consistent GW@PBE calculation: H2O

Download GW_H2O.run

```
#! /bin/sh

# GW calculation for Water. By default. The highest 5 occupied and lowest 5
# unoccupied states are calculated.

# We use an all-electron basis set since core-correlation effects are important.
# A QZ basis set is recommended for relatively converged QP energies

# Fot this example we will use the GGA PBE.
# This is NOT a recommended starting point for a G0W0 calculation.
# However, the eigenvalue-only self-consistency removes most of the #
# starting point dependence of the functional.
# Thererfore, PBE is a reasonable choice.

# RECOMMENDED: VeryGood numerical quality,
# especially in self-consistent GW calculations.

$AMSBIN/ams << eor
Symmetry
  SymmetrizeTolerance 0.001
End

System
  Atoms
```

(continues on next page)

(continued from previous page)

```
    O    2.220871067   0.026716792   0.000620476
    H    2.597492682  -0.411663274   0.766744858
    H    2.593135384  -0.449496183  -0.744782026
  End
  Symmetrize Yes
End

task SinglePoint

Engine band
  Basis
    Core None
    Type Corr/QZ6P
  end

  NumericalQuality VeryGood

  Dependency
     AllowBasisDependency True
     basis 1e-4
  End

  RIHartreeFock
     DependencyThreshold 1e-3
  End

  SoftConfinement
     Quality Excellent
  End

  usesymmetry False

  XC
    gga PBE
  end

  GW
     selfconsistency evGW
  END
EndEngine
eor
```

10.11 Analysis

10.11.1 Example: CO absorption on a Cu slab: fragment option and densityplot

Download Frags_COCu.run

```
#! /bin/sh

# This example illustrates the usage of fragments 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
```

(continues on next page)

(continued from previous page)

```
# 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.results/band.rkf'.


# ---------------------------- CO molecule -------------------------

AMS_JOBNAME=CO $AMSBIN/ams <<eor

Task SinglePoint

System
  ! CO molecules far apart

  Atoms [Bohr]
    C   0 0 0
    O   0 0 2.18
  End

  Lattice [Bohr]
    25.0  0.0   0.0
    0.0   25.0  0.0
  End
End

Engine Band
  Title The CO fragment

  Print AtomicChargesDetails

  Comment
   Technical
     Zero order k space integration
   Features
     Lattice   : 2D, large lattice vectors
     Unit cell : 2 atoms, 1x1, quasi molecular
     Basis     : NO+STO w/ core
  End

  Print Eigens

  Kspace
    Quality GammaOnly  ! neglect dispersion
  End

  Basis
    Type DZ
    Core Large
  End

  DOS
     CalcPDOS True
```

(continues on next page)

(continued from previous page)

```
    Energies 300
  End
EndEngine
eor


# 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 --------------------------

AMS_JOBNAME=COCu $AMSBIN/ams <<eor

Task SinglePoint

System
  Lattice [Bohr]
    4.822 0.0   0.0
    0.0   4.822 0.0
  End
  Atoms [Bohr]
    C   0     0     3.44
    O   0     0     5.62
    Cu  0.0   0.0   0.0
  End
End

Engine Band
  Title Cu slab with CO adsorbed

  Print AtomicChargesDetails

  Comment
   Technical
     Quadratic K space integration (low)
   Features
     Lattice   : 2D
     Unit cell : 3 atoms, 1x1
     Basis     : NO+STO w/ core
     Options   : Molecular fragment
                 Analysis: DOS, PDOS, COOP
  End

  KSpace
    Symmetric KInteg=3
  End

  ! fragment specification

  Fragment
    filename CO.results/band.rkf
    atommapping
      1 1
      2 2
```

(continues on next page)

(continued from previous page)

```
    End
    Labels ! let us give them some labels
      2Sigma
      2Sigma*
      1Pi_x
      1Pi_y
      3Sigma
      1Pi_x*
      1Pi_y*
      3Sigma*
    End
  End

  ! use fragment basis in dos
  DosBas
    Fragment 1
  End

  DOS   ! Analysis
    CalcPDOS   True
    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

  Basis
    Type DZ
    Core Large
  End
EndEngine
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.
```

(continues on next page)

(continued from previous page)

```
echo ""
echo "Contents of DOS file"
cat pdos.CO_Cu

# 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 -------------------------

AMS_JOBNAME=Cu $AMSBIN/ams <<eor

Task SinglePoint

System
  Lattice [Bohr]
    4.822 0.0 0.0
    0.0   4.822 0.0
  End
  Atoms [Bohr]
    Cu  0.0   0.0   0.0
  End
End


Engine Band
  Title Cu slab

  Print AtomicChargesDetails

  Comment
   Technical
     Quadratic K space integration (low)
   Features
     Lattice   : 2D
     Unit cell : 3 atoms, 1x1
     Basis     : NO+STO w/ core
     Options   :
  End

  Kspace
    Symmetric KInteg=3
  End

  Basis
    Type DZ
    Core Large
  End

  DOS
     CalcPDOS True
     Energies 300
  End
EndEngine
eor
```

(continues on next page)

(continued from previous page)

```
# Now we are all set to do our final calculation. We have the two fragment files
# CO.results/band.rkf and Cu.results/band.rkf, and the restart file COCu.results/band.
↪rkf. 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 -------------------------

NSCM=1
export NSCM

AMS_JOBNAME=Final $AMSBIN/ams <<eor

Task SinglePoint

System
  Lattice [Bohr]
    4.822 0.0 0.0
    0.0   4.822 0.0
  End

  Atoms [Bohr]
    C   0     0     3.44
    O   0     0     5.62
    Cu  0.0   0.0   0.0
  End
End

Engine Band
  Title Cu slab with CO adsorbed (restart density plot)

  Print AtomicChargesDetails

  debug BlockPropertyModule

  Kspace
    Symmetric KInteg=3
  End

  Restart
    File COCu.results/band.rkf
    DensityPlot
  End

  Grid
    Type Coarse
  End

  DensityPlot
    rho(deformation/fit) !FITDENSITY_deformation_scf
  End

  ! fragment specification

  Fragment
    filename CO.results/band.rkf
```

(continues on next page)

(continued from previous page)

```
    atommapping
    1 1
    2 2
    End
  End

  Fragment
    filename Cu.results/band.rkf
    atommapping
      1 3
    End
  End

  Basis
    Type DZ
    Core Large
  End

  DOS
     CalcPDOS True
     Energies 300
  End
EndEngine
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 AMSview. On the TAPE41 file are now three
# fields shown in AMSview 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 AMSview you can add the fields of the two fragments, and then create
# another field that holds the difference.

NSCM=1
export NSCM
echo ""
echo "Begin TOC of tape41"

$AMSBIN/dmpkf -n 1 Final.results/FILE_BLOCKPROPERTIES --toc

echo "End TOC of tape41"
```

10.11.2 Example: Grid key for plotting results

Download GridKey.run

```
#!/bin/sh

SYSTEM=$AMSHOME/atomicdata/Molecules/TestMols/Methane.xyz

# Initial run

AMS_JOBNAME=methane $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile $SYSTEM
End

Engine Band
  Basis
    Type TZP
  End
EndEngine
eor


# Use the grid

AMS_JOBNAME=auto_grid $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile $SYSTEM
End

Engine Band
  Restart
    File methane.results/band.rkf
    DensityPlot
  End

  Grid
   Type Coarse
   ExtendX 21.1671 [Angstrom]
  End

  DensityPlot
    rho(fit)
  End

  Basis
    Type TZP
  End
EndEngine
eor
```

(continues on next page)

(continued from previous page)

```
echo ""
echo "Begin TOC of tape41"

$AMSBIN/dmpkf -n 1 auto_grid.results/TAPE41 --toc

echo "End TOC of tape41"



# Use a completely user specified regular grid

AMS_JOBNAME=user_grid $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile $SYSTEM
End

Engine Band
  Restart
    File methane.results/band.rkf
    DensityPlot
  End

  Grid
    UserDefined  # in Bohr
      -2.0 -1.3 -2.5
      1.0 0.0 0.0 0.02
      0 1 0.0 0.02
      0.0 0.0 1.0 0.02
      20 30 40
    End
  End

  DensityPlot
    rho(fit)
  End

  Basis
    Type TZP
  End
EndEngine
eor

echo ""
echo "Begin TOC of tape41"

$AMSBIN/dmpkf -n 1 user_grid.results/TAPE41 --toc

echo "End TOC of tape41"


# Use a text file to import the (arbitrary grid in Bohr)

cat << eor > coords.txt
```

(continues on next page)

(continued from previous page)

```
-3.0 0.0 0.0
-2.0 0.1 0.0
 0.0 0.2 0.0
eor

AMS_JOBNAME=file_grid $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile $SYSTEM
End

Engine Band
  Restart
    File methane.results/band.rkf
    DensityPlot
    vtkFile result.txt
  End

  Grid
    Filename coords.txt
  End

  DensityPlot
    rho(fit)
  End

  Basis
    Type TZP
  End
EndEngine

eor

echo ""
echo "Begin of result.txt"
cat result.txt
echo "End of result.txt"




AMS_JOBNAME=generate_cube $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile $SYSTEM
End

Engine Band
  Restart
    File methane.results/band.rkf
    DensityPlot
    vtkFile CUBE
  End
```

(continues on next page)

(continued from previous page)

```
  Grid
  Type Coarse
  End

  DensityPlot
    rho(fit)
  End

  Basis
    Type TZP
  End
EndEngine

eor

echo ""
echo "Begin of cube files"
cat rho*.cube
echo "End of cube files"

AMS_JOBNAME=generate_cube_with_name $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile $SYSTEM
End

Engine Band
  Restart
    File methane.results/band.rkf
    DensityPlot
    vtkFile densf.cube
  End

  Grid
  Type Coarse
  End

  DensityPlot
    rho(fit)
  End

  Basis
    Type TZP
  End
EndEngine

eor

echo ""
echo "Begin of named cube files"
cat densf*.cube
echo "End of named cube files"
```

(continues on next page)

(continued from previous page)

```
# Use a completely user specified regular grid, now in angstrom

AMS_JOBNAME=user_grid_angstrom $AMSBIN/ams <<eor

Task SinglePoint

System
  GeometryFile $SYSTEM
End

Engine Band
  Restart
    File methane.results/band.rkf
    DensityPlot
  End

  Grid
    UserDefined  [Angstrom]
      -2.0 -1.3 -2.5
      1.0 0.0 0.0 0.02
      0 1 0.0 0.02
      0.0 0.0 1.0 0.02
      20 30 40
    End
  End

  DensityPlot
    rho(fit)
  End

  Basis
    Type TZP
  End
EndEngine
eor

echo ""
echo "Begin TOC of tape41"

$AMSBIN/dmpkf -n 1 user_grid.results/TAPE41 --toc

echo "End TOC of tape41"
```

10.11.3 Example: H2 on [PtCl4]2-: charged molecules and PEDA

Download PEDA_0D_PtCl4H2.run

```
#! /bin/sh

# This example shows that the pEDA formalism can be applied to
# molecules. Here, there is no periodic boundary condition
# necessary. Hence, charged fragments or final molecules can be
# investigated!

#
```

(continues on next page)

(continued from previous page)

```
#
# Fragment 1 is the [PtCl4]2- fragment
#
#


AMS_JOBNAME=Frag1 $AMSBIN/ams <<eor

Task SinglePoint

System
    ATOMS
        Pt 0.0 0.0 0.0
        Cl 0.0 -2.308048739 0.0
        Cl 0.0 2.308048739 0.0
        Cl -2.308048739 0.0 0.0
        Cl 2.308048739 0.0 0.0
    END

    Charge -2
End

Engine Band
    TITLE PtCl4 2- fragment
    Relativity
        Level Scalar
    End

    Basis
    Type DZP
    Core Large
    End

    XC
    GGA Becke Perdew
    END

    UseSymmetry False
EndEngine
eor

#
#
# Fragment 2 is the H2 fragment
#
#

AMS_JOBNAME=Frag2 $AMSBIN/ams <<eor

Task SinglePoint

System
    ATOMS
        H 0.0 0.0 3.84182655
        H 0.0 0.0 2.952808836
    END
End
```

(continues on next page)

(continued from previous page)

```
Engine Band
    TITLE H2 fragment


    Relativity
        Level Scalar
    End

    Basis
    Type DZP
    Core Large
    End

    XC
    GGA Becke Perdew
    END

    UseSymmetry False
EndEngine
eor

#
#
# The energy decomposition run for the complex ([PtCl4]H2)2- complex
#
#

$AMSBIN/ams <<eor

Task SinglePoint

System
    ATOMS
        Pt 0.0 0.0 0.0
        Cl 0.0 -2.308048739 0.0
        Cl 0.0 2.308048739 0.0
        Cl -2.308048739 0.0 0.0
        Cl 2.308048739 0.0 0.0
        H 0.0 0.0 2.952808836
        H 0.0 0.0 3.84182655
    END
    Charge -2
End

Engine Band
    Relativity
        Level Scalar
    End

    Basis
    Type DZP
    Core Large
    End

    XC
    GGA Becke Perdew
```

(continues on next page)

(continued from previous page)

```
    END

    fragment
        filename Frag1.results/band.rkf
        AtomMapping
            1 1
            2 2
            3 3
            4 4
            5 5
        End
    end

    fragment
        filename Frag2.results/band.rkf
        AtomMapping
            1 7
            2 6
        End
    end

    PEDA

    UseSymmetry False
EndEngine

eor
```

10.11.4 Example: CO absorption on a MgO slab: fragment option and PEDA

Download PEDA_MgO+CO.run

```
#! /bin/sh

# This example shall illustrate the use of the Fragment keywords in combination
# with the PEDA 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 -------------------------

AMS_JOBNAME=MgO $AMSBIN/ams <<eor

Task SinglePoint

System
  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
```

(continues on next page)

(continued from previous page)

```
     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
End

Engine Band
  Title MgO surface

  skip dos

  KSpace
    Regular
      NumberOfPoints 3 3
    End
  End

  XC
     GGA PBE
  End

  Basis
     Type TZP
     Core small
  End
EndEngine

eor

#------------------------------ CO fragment ------------------------

AMS_JOBNAME=CO $AMSBIN/ams <<eor

Task SinglePoint

System
  Atoms
     C   0.00000000       0.00000000       2.61000000
     O   0.00000000       0.00000000       3.737000000
  End

  Lattice
        3.00520382      -3.00520382       0.00000000
        3.00520382       3.00520382       0.00000000
  End
End

Engine Band
  Title CO fragment
```

(continues on next page)

(continued from previous page)

```
  KSpace
    Regular
      NumberOfPoints 3 3
    End
  End

  XC
     GGA PBE
  End

  Basis
     Type TZP
     Core small
  End
EndEngine
eor

# == PEDA calculation ==

# The two result files, MgO.kf and CO.kf, 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 ------------------------

$AMSBIN/ams <<eor

Task SinglePoint

System
  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
End

Engine Band
  Title PEDA
```

(continues on next page)

(continued from previous page)

```
  KSpace
    Regular
      NumberOfPoints 3 3
    End
  End

  XC
     GGA PBE
  End

  fragment
    filename MgO.results/band.rkf
    AtomMapping
       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
  end

  fragment
    filename CO.results/band.rkf
    AtomMapping
       2  13
       1  14
     End
  end

  PEDA

  Basis
     Type TZP
     Core small
  End
EndEngine

eor

# In the output file the results can be found in the PEDA block after the Energy
# Analysis.
```

10.11.5 Example: CO absorption on a MgO slab: fragment option, PEDA and PEDANOCV

Download PEDANOCV_MgO+CO.run

```
#! /bin/sh

# This example shall illustrate the use of the Fragment keywords in combination
# with the PEDA and PEDANOCV 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 gamma point

# == Fragment calculations ==

# ------------------------------ MgO slab ------------------------

AMS_JOBNAME=MgO $AMSBIN/ams <<eor

Task SinglePoint

System
   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
End

Engine Band
   Title MgO fragment

   skip dos

    KSpace
      Regular
        NumberOfPoints 1 1
      End
    End

   BeckeGrid
      quality basic
   End
```

(continues on next page)

(continued from previous page)

```
   Relativity
      Level Scalar
   End

   XC
      GGA PBE
   End

   Basis
      Type TZP
      Core none
   End
EndEngine
eor

#------------------------------ CO fragment ------------------------

AMS_JOBNAME=CO $AMSBIN/ams <<eor

Task SinglePoint

System
   Atoms
      C   0.00000000       0.00000000       2.61000000
      O   0.00000000       0.00000000       3.737000000
   End

   Lattice
         3.00520382      -3.00520382       0.00000000
         3.00520382       3.00520382       0.00000000
   End
End

Engine Band
   Title CO fragment

   KSpace
      Regular
         NumberOfPoints 1 1
      End
   End

   BeckeGrid
      quality basic
   End

   Relativity
      Level Scalar
   End

   XC
      GGA PBE
   End

   Basis
      Type TZP
      Core none
```

(continues on next page)

(continued from previous page)

```
   End
EndEngine
eor

# == PEDANOCV calculation ==

# The two result files, MgO.kf and CO.kf, 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 gamma
# point for all calculations.

#------------------------------ PEDANOCV calculation ------------------------


AMS_JOBNAME=decomp $AMSBIN/ams <<eor

Task SinglePoint

System
   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       3.737000000
   End

   Lattice
          3.00520382      -3.00520382       0.00000000
          3.00520382       3.00520382       0.00000000
   End
End

Engine Band
   Title Mg+CO

   KSpace
      Regular
         NumberOfPoints 1 1
      End
   End

   BeckeGrid
      quality basic
   End

   Relativity
```

(continues on next page)

(continued from previous page)

```
      Level Scalar
   End

   XC
      GGA PBE
   End

   fragment
      filename MgO.results/band.rkf
      AtomMapping
         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
   end

   fragment
      filename CO.results/band.rkf
      AtomMapping
         1  13
         2  14
      End
   end

   PEDA

   PEDANOCV
      Enabled True
      EigvalThresh 0.001
   End

   Basis
      Type TZP
      Core none
   End
EndEngine
eor

# 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
# AMSview or by a restart calculation. In the latter case, one adds the Restart
# block key with the options File decomp.kf 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 and NOCVOrbitalPlot block key. In both
# blocks the line 1 Band 1 5 means, that for k-point 1 the densities/orbitals 1
```

(continues on next page)

(continued from previous page)

```
# to 5 are calculated.



export NSCM=1
$AMSBIN/ams <<eor

Task SinglePoint

System
   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       3.737000000
   End

   Lattice
         3.00520382      -3.00520382       0.00000000
         3.00520382       3.00520382       0.00000000
   End
End

Engine Band
   Title Restart Calculation

   Restart
      File decomp.results/band.rkf
      NOCVdRhoPlot
   End

   NOCVdRhoPlot
      1 Band 1
   End

   Grid
      Type coarse
   End

   KSpace
      Regular
         NumberOfPoints 1 1
      End
   End

   BeckeGrid
      quality basic
```

(continues on next page)

(continued from previous page)

```
   End

   Relativity
      Level Scalar
   End

   XC
      GGA PBE
   End

   Basis
      Type TZP
      Core none
   End

   debug BlockPropertyModule
EndEngine
eor

echo ""
echo "Begin TOC of tape41"
export NSCM=1
$AMSBIN/pkf -n 1 ams.results/FILE_BLOCKPROPERTIES

echo "End TOC of tape41"

# The important output of this calculation is the TAPE41 file. Renaming it to
# foobar.t41 will allow AMSview to read and interpret the data stored on this
# file.
```

10.11.6 Example: Bader analysis

Download Li2O_Bader.run

```
#! /bin/sh

# To get the Quantum Theory of Atoms In Molecules and Crystals (QT-AIMAC)
# analysis use the GridBasedAIM 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.

$AMSBIN/ams <<eor

Task SinglePoint

System
   Lattice [Bohr]
      0.0   4.365 4.365
      4.365 0.0   4.365
      4.365 4.365 0.0
   end

   Atoms [Bohr]
```

(continues on next page)

(continued from previous page)

```
      O   0.0     0.0      0.0
      Li  2.1825  2.1825   2.1825
      Li  6.5475  2.1825   2.1825
   end
End

Engine Band
   Title Li2O bulk (fluorite structure)

   KSpace
      Symmetric KInteg=3
   End

   IntegrationMethod Voronoi

   Integration
      Accint 4
      accsph 6
      accpyr 6
   end

   GridBasedAIM
      Enabled Yes
   End

   Dependency basis=1e-9 fit=1e-8

   DIIS
      dimix 0.2
      ncycledamp 0
   end

   scf
      mixing 0.4
   end

   xc
      gga scf bp86
   end

   Basis
      Type TZ2P
      Core small
   end

   Relativity Level=None
EndEngine
eor
```

10.11.7 Example: Properties at nuclei

Download PropertiesAtNuclei.run

```
#!/bin/sh

# One can obtain the values of some properties near the nucleus. (see
# PropertiesAtNuclei)

# Note: Instead of calculating the properties at a point in space an average is
# taken over a tiny sphere around this point.

$AMSBIN/ams <<eor

Task SinglePoint

System
   Atoms
      O   0.000  0.000  0.000
      O   0.000  0.000  1.208
   end
End

Engine Band
   Title Properties at nuclei for O2

   Unrestricted Yes

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

   Basis
      Type DZ
      Core None
   End

   Relativity Level=None

   XC
      gga always pbe
   END
EndEngine
eor
```

10.11.8 Example: Band structure plot

Download Li_BZPlot.run

```
#! /bin/sh

# In the first example we use the automatic k-path through the Brillouin zone
# (see BandStructure key-block). The results can be visualized with the BandStructure
# Gui Module.


AMS_JOBNAME=auto $AMSBIN/ams <<eor

Task SinglePoint

System
  Atoms
     Li 0.0 0.0 0.0
  END
  Lattice
    -1.745  1.745  1.745
     1.745 -1.745  1.745
     1.745  1.745 -1.745
  End
End

Engine Band
  NumericalQuality Basic
  Relativity Level=None

  BandStructure
     Enabled   true
     Automatic true
     FatBands  false
     EnergyAboveFermi 2.0
  end
EndEngine
eor

# In the second example we specify the path through the Brillouin zone by hand.
# We set automatic to false and then specify the path with the BZPath key block,
# using one or more path subkeys. Here, the second run will produce exactly the
# same path as the automatic one.

AMS_JOBNAME=user $AMSBIN/ams <<eor

Task SinglePoint

System
  Atoms
     Li 0.0 0.0 0.0
  END
  Lattice
    -1.745  1.745  1.745
     1.745 -1.745  1.745
     1.745  1.745 -1.745
  End
End
```

(continues on next page)

(continued from previous page)

```
Engine Band
  NumericalQuality Basic
  Relativity Level=None

  BandStructure
     Enabled   true
     Automatic false
     FatBands  false
     EnergyAboveFermi 2.0
  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
     End
     path
        0.25  0.25  0.25  P
        0.00  0.00  0.50  N
     End
  end
EndEngine
eor

export NSCM=1

# The band structure is best visualized using the BandStructure GUI module.

echo 'Extract the band_curves section from the rkf files:'
$AMSBIN/dmpkf auto.results/band.rkf 'band_curves'
$AMSBIN/dmpkf user.results/band.rkf 'band_curves'
```

10.11.9 Example: Effective Mass (electron mobility)

Download EffectiveMass.run

```
#! /bin/sh

# 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 key
# block (so leave out the NumAbove, NumBelow, and UniqueKPoints).

# == Example 1D ==

echo "example 1D"
```

(continues on next page)

(continued from previous page)

```
AMS_JOBNAME=EffectiveMass1D $AMSBIN/ams <<eor

Task SinglePoint

System
  Atoms
     Al 0.0 0.0 0.0
  END
  Lattice
     2.12440502 0.0 0.0
  End
End

Engine Band
  TITLE 1D Al Chain

  EffectiveMass
    Enabled True
    KPointCoord -0.783
    StepSize 0.001
    NumAbove 4
    NumBelow 2
  End

  Basis
     Type DZ
     Core Large
  End
EndEngine
eor


# == Example 2D ==

echo "example 2D"

AMS_JOBNAME=EffectiveMass2D $AMSBIN/ams <<eor

Task SinglePoint

System
  Atoms
     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
End

Engine Band
  TITLE MoS2Slab
```

(continues on next page)

(continued from previous page)

```
  EffectiveMass
    Enabled True
  End

  Basis
     Type DZ
     Core Large
  End
EndEngine

eor

# == Example 3D ==

echo "example 3D"

AMS_JOBNAME=EffectiveMass3D $AMSBIN/ams <<eor

Task SinglePoint
System
  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
End

Engine Band
  TITLE ZnO

  NumericalQuality Basic

  KSpace
    Quality Normal
  End
  tails bas=1e-8

  EffectiveMass
    Enabled True
    NumAbove 1
    NumBelow 1
  End

  Basis
     Type DZ
     Core Large
  End
EndEngine

eor
```

10.11.10 Example: Generating an Excited State with and Electron Hole

Download Si_ElectronHole.run

```
#! /bin/sh

# There is the possibility define the excitation of an electron from a low
# lying, localized band to a virtual band. The ElectronHole key does allow the
# specification of the original band and the spin of the electron. The
# EnforcedSpinPolarization key allows to restrict the spin polarization of the
# whole system.

$AMSBIN/ams <<eor

Task SinglePoint

System
  Atoms
    Si.frozen_core -0.67875 -0.67875 -0.67875
    Si              0.67875  0.67875  0.67875
  End
  Lattice
      0.000  2.715  2.715
      2.715  0.000  2.715
      2.715  2.715  0.000
  End
End

Engine Band

  TITLE Untitled

  Basis
    Type DZP
    Core None
    PerAtomType Symbol=Si.frozen_core Type=DZP Core=Large
  End

  XC
     LDA SCF VWN
  END

  Unrestricted Yes

  ElectronHole
      BandIndex 1
      SpinIndex 1
  End

  EnforcedSpinPolarization 0
EndEngine
eor
```

10.11.11 Example: LDOS (STM) for a BN slab

The Local DOS (LDOS) is the partial density due to bands withing a certain energy interval. It has been related to STM images.

Download BNSlabLDOS.run

```
# The BN slab has a band gap
# The bottom of the conduction band  (BOCB, 0.05 hartree above the fermi level) ␣
↪consists of p_z orbitals on B
# The top of the valence band (TOVB, 0.2 hartree under the fermi level)  consists of␣
↪p_z oribtals on N

# It is recommended to inspect BNSlab.results with amsbands
# and BNSlab.ldos.tovb.results with amsview (and the same for bocb)

system=BNSlab

export AMS_JOBNAME=$system

rm -rf $AMS_JOBNAME.results

$AMSBIN/ams<<EOF

Task SinglePoint

System
   Atoms
               B      -0.615000000    -0.355070416     0.000000000
               N       0.615000000     0.355070416     0.000000000
    End
        Lattice
             2.460000000     0.000000000     0.000000000
             1.230000000     2.130422493     0.000000000
End

End

Engine Band
        BandStructure Enabled=yes
        Dos CalcPDOS=yes
        kspace quality=good
EndEngine

EOF

fermi_energy=`$AMSBIN/amsreport $AMS_JOBNAME.results/band.rkf -r "BandStructure
↪%FermiEnergy"`
bcb=`$AMSBIN/amsreport $AMS_JOBNAME.results/band.rkf -r "BandStructure
↪%BottomConductionBand"`
tvb=`$AMSBIN/amsreport $AMS_JOBNAME.results/band.rkf -r "BandStructure%TopValenceBand
↪"`

echo "Fermi energy (Hartree): $fermi_energy"
echo "tvb (Hartree): $tvb"
echo "bcb (Hartree): $bcb"

rel_tvb=`$AMSBIN/amspython -c "print($fermi_energy-$tvb)"`
```

(continues on next page)

(continued from previous page)

```
rel_bcb=`$AMSBIN/amspython -c "print($bcb-$fermi_energy)"`

echo "rel_bcb (Hartree): $rel_bcb"
echo "rel_tvb (Hartree): $rel_tvb"

# we want to sample just a bit above/below the bcb and tvb: both need a margin
# the applied margins here are quite arbitrary, but ensure that the example works␣
↪exactly as before

rel_tvb_with_margin=`$AMSBIN/amspython -c "print($rel_tvb+0.0585153984080739)"`
rel_bcb_with_margin=`$AMSBIN/amspython -c "print($rel_bcb+0.0205273425101303)"`

echo "rel_bcb_with_margin (Hartree): $rel_bcb_with_margin"
echo "rel_tvb_with_margin (Hartree): $rel_tvb_with_margin"


export AMS_JOBNAME=$system.ldos.bocb

rm -rf $AMS_JOBNAME.results


$AMSBIN/ams  --delete-old-results << EOF
Task SinglePoint

LoadSystem
    File $system.results/ams.rkf
End

Engine BAND
Restart
   File $system.results/band.rkf
   DensityPlot
End

Grid
   Type Coarse
End

DensityPlot
   LDOS
End

LDOS
   DeltaNeg 0.0 # fermi energy, halfway the gap
   DeltaPos $rel_bcb_with_margin   # bit above the bcb
End
EndEngine
EOF



export AMS_JOBNAME=$system.ldos.tovb

rm -rf $AMS_JOBNAME.results


$AMSBIN/ams  --delete-old-results << EOF
```

(continues on next page)

(continued from previous page)

```
Task SinglePoint

LoadSystem
    File $system.results/ams.rkf
End

Engine BAND
Restart
   File $system.results/band.rkf
   DensityPlot
End

Grid
   Type Coarse
End

DensityPlot
   LDOS
End

LDOS
   DeltaNeg $rel_tvb_with_margin  # bit below the tvb
   DeltaPos 0.0 # fermi energy, halfway the gap
End
EndEngine
EOF


echo "Begin TOC of tape41 (tovb)"

$AMSBIN/dmpkf -n 1 $system.ldos.tovb.results/TAPE41 --toc | grep LDOS

echo "End TOC of tape41"


echo "Begin TOC of tape41 (bocb)"

$AMSBIN/dmpkf -n 1 $system.ldos.bocb.results/TAPE41 --toc | grep LDOS

echo "End TOC of tape41"
```

10.12 List of Examples

- *BasisDefaults* (page 228)
- *BeO_tape41* (page 246)
- *BetaIron* (page 211)
- *BField* (page 212)
- *BFieldLdotB* (page 213)
- *BNForce* (page 276)
- *BSSE* (page 232)
- *COChainFreqTS* (page 269)
- *EffectiveMass* (page 319)
- *EField* (page 218)
- *FiniteNucleus* (page 219)
- *Frags_COCu* (page 292)
- *Graphene_Dispersion* (page 214)
- *GraphenePhonons* (page 289)
- *GridKey* (page 299)
- *H2BulkGeo* (page 276)
- *H_ref* (page 271)
- *HonPerovskite_Solvation* (page 216)
- *Li2O_Bader* (page 315)
- *Li_BZPlot* (page 318)
- *Multiresolution_H2O* (page 230)
- *NaCl* (page 267)
- *NEGF* (page 253)
- *NEGF with bias* (page 260)
- *NEGF_Conductance* (page 263)
- *NewResp_3DCopper* (page 281)
- *NewResp_PlotInducedDensity*

CHAPTER

ELEVEN

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.

11.1 General References

For calculations with the periodic structures BAND program, version 2020.1:

1. G. te Velde and E.J. Baerends, *Precise density-functional method for periodic structures*, Physical Review B 44, 7888 (1991) (https://doi.org/10.1103/PhysRevB.44.7888).
2. BAND 2025.1, 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, M.C.W.M. Pols, P. Romaniello, M. Raupach, D.G. Skachkov, J.G. Snijders, C.J.O. Verzijl, J.A. Celis Gil, J. M. Thijssen, G. Wiesenekker, C. A. Peeples, G. Schreckenbach, 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.

11.2 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) (https://doi.org/10.1088/0022-3719/21/23/012).

G. Wiesenekker and E.J. Baerends, *Quadratic integration over the three-dimensional Brillouin zone*, Journal of Physics: Condensed Matter 3, 6721 (1991) (https://doi.org/10.1088/0953-8984/3/35/005).

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) (https://doi.org/10.1002/jcc.23323).

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) (https://doi.org/10.1021/ct500172n).

11.2.1 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) (https://doi.org/10.1080/00268970701598063).

11.2.2 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) (https://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) (https://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) (https://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) (https://doi.org/10.1103/PhysRevB.73.075115).

Suggested book references

F. Kootstra, Ph.D. thesis (http://downloads.scm.com/Doc/ft439.pdf), Rijksuniversiteit Groningen, Groningen (2001).

P. Romaniello, Ph.D. thesis (http://downloads.scm.com/Doc/Thesis_Pina.pdf), Rijksuniversiteit Groningen, Groningen (2006).

A. Berger, Ph.D. thesis (http://downloads.scm.com/Doc/Thesis_Arjan.pdf), Rijksuniversiteit Groningen, Groningen (2006).

11.2.3 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) (https://doi.org/10.1063/1.2780146).

11.2.4 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) (https://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) (https://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) (https://doi.org/10.1103/PhysRevB.75.035116).

11.2.5 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) (https://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) (https://doi.org/10.1139/v11-050).

11.2.6 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) (https://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) (https://doi.org/10.1021/jp805466c).

11.2.7 NEGF

Lead

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 (https://doi.org/10.1021/jp3044225).

11.2.8 Electron energy density

Lead

Stefano Racioppi and Martin Rahm, *In-Situ Electronegativity and the Bridging of Chemical Bonding Concepts*, Chemistry – A European Journal 72 18156-18167 (2021) (https://doi.org/10.1002/chem.202103477).

Stefano Racioppi, Per Hyldgaar and Martin Rahm, *Quantifying Atomic Volume, Partial Charge, and Electronegativity in Condensed Phases*, The Journal of Physical Chemistry C 128.9 (2024): 4009 (https://doi.org/10.1021/acs.jpcc.3c07677).

11.3 External programs and Libraries

Third party software used in the 2025.1 version of the Amsterdam Modeling Suite can be found in the file titled “third-party-software.txt” in the root of your AMS installation.

CHAPTER

TWELVE

KEYWORDS

12.1 Links to manual entries

ams:

- *System* (page 131)

band:

- *AIMCriticalPoints* (page 154)
- *ATensor* (page 125)
- *BField* (page 49)
- *BZPath* (page 150)
- *BandStructure* (page 148)
- *Basis* (page 54)
- *BeckeGrid* (page 71)
- *BerryPhase* (page 131)
- *CPVector* (page 111)
- *Convergence* (page 84)
- *DIIS* (page 87)
- *DOS* (page 143)
- *DensityPlot* (page 183)
- *Dependency* (page 107)
- *EFG* (page 126)
- *ESR* (page 125)
- *EffectiveMass* (page 128)
- *ElectronHole* (page 192)
- *EnforcedSpinPolarization* (page 33)
- *Fermi* (page 109)
- *FermiSurface* (page 152)
- *FormFactors* (page 130)
- *Fragment* (page 156)
- *FuzzyPotential* (page 51)
- *GW* (page 137)
- *Grid* (page 181)
- *GridBasedAIM* (page 153)
- *GrossPopulations* (page 146)
- *HubbardU* (page 25)
- *Integration* (page 74)
- *IntegrationMethod* (page 74)
- *KGrpX* (page 111)
- *KSpace* (page 66)
- *LDOS* (page 187)
- *MBPT* (page 102)
- *MultiSecantConfig* (page 90)
- *NEGF* (page 170)
- *NMR* (page 127)
- *NOCVOrbitalPlot* (page 186)
- *NOCVdRhoPlot* (page 186)
- *NewResponse* (page 115)
- *NewResponseKSpace* (page 120)
- *NewResponseSCF* (page 117)
- *NuclearModel* (page 52)
- *NumericalQuality* (page 53)
- *Occupations* (page 192)
- *OldResponse* (page 121)
- *OrbitalPlot* (page 184)
- *OverlapPopulations* (page 147)
- *PEDA* (page 157)
- *PEDANOCV* (page 158)
- *PeriodicSolvation* (page 40)
- *PotentialNoise* (page 192)
- *RIHartreeFock* (page 79)
- *RadialDefaults* (page 73)
- *Relativity* (page 33)
- *ResponseInducedDensityPlot* (page 185)
- *Restart* (page 179)
- *SCF* (page 81)
- *Save* (page 189)
- *Screening* (page 108)
- *SoftConfinement* (page 62)
- *Solvation* (page 36)
- *SolvationSM12* (page 42)

- *StoreHamiltonian2* (page 177)
- *SubSymmetry* (page 191)
- *Tails* (page 106)
- *Unrestricted* (page 32)
- *UseSymmetry* (page 189)
- *XC* (page 29)
- *ZlmFit* (page 75)

12.2 Summary of all keywords

12.2.1 Engine Band

`AIMCriticalPoints`

Type
Block

Description
Compute the critical points of the density (Atoms In Molecules). The algorithm starts from a regular mesh of points, and from each of these it walks towards its corresponding critical point.

`Enabled`

Type
Bool

Default value
No

GUI name
: Critical points and bond paths

Description
Compute the critical points of the density (Atoms In Molecules). The algorithm starts from a regular mesh of points, and from each of these it walks towards its corresponding critical point.

`EqvPointsTol`

Type
Float

Default value
0.27

Unit
Bohr

Description
If the distance between two critical points is smaller than this value, the two critical points are considered to be the same point.

`GridPadding`

Type
Float

Default value
0.7

Unit
Bohr

Description
How much extra space is added to the starting guess domain in the search for the critical points

GridSpacing

Type
Float

Default value
0.5

Unit
Bohr

Description
The distance between the initial trial points.

Allow

Type
String

Recurring
True

Description
Debugging feature to let the program continue even when intermediate results seem to be wrong or very inaccurate

ATensor

Type
Block

Description
Hyperfine A-tensor.

Enabled

Type
Bool

Default value
No

GUI name
:A-tensor

Description
Compute the hyperfine A-tensor.

Note: Unrestricted calculation is required.

AtomType

Type
Block

Recurring
True

Description
Explicit basis set definition for given atom type.

AutomaticGaussians

Type
Non-standard block

Description
Definition of the automatic gaussians

`BasisFunctions`

Type
Non-standard block

Description
Definition of the extra Slater-type orbitals

`ContractedGaussians`

Type
Non-standard block

Description
Definition of the contracted gaussians

`Dirac`

Type
Non-standard block

Description
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

`FitFunctions`

Type
Non-standard block

Description
Slater-type fit functions. Obsolete feature.

`BandStructure`

Type
Block

Description
Options for the calculation of the band structure.

`Automatic`

Type
Bool

Default value
Yes

GUI name
Automatic generate path

Description
If True, BAND will automatically generate the standard path through the Brillouin zone.

If False BAND will use the user-defined path in BZPath.

`DeltaK`

Type
Float

Default value
0.1

Unit
1/Bohr

GUI name
Interpolation delta-K

Description
Step (in reciprocal space) 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.

`Enabled`

Type
Bool

Default value
No

GUI name
Calculate band structure

Description
If True, Band will calculate the band structure and save it to file for visualization.

`EnergyAboveFermi`

Type
Float

Default value
0.75

Unit
Hartree

GUI name
Energy above Fermi level

Description
Bands with minimum energy larger then FermiEnergy + EnergyAboveFermi are not saved to file. Increasing the value of EnergyAboveFermi will result in more unoccupied bands to be saved to file for visualization.

`EnergyBelowFermi`

Type
Float

Default value
10.0

Unit
Hartree

GUI name
Energy below Fermi level

Description
Bands with maximum energy smaller then FermiEnergy - EnergyBelowFermi are not saved to file. Increasing the value of EnergyBelowFermi will result in more occupied core bands to be saved to file for visualization. Note: EnergyBelowFermi should be a positive number!

`FatBands`

Type
Bool

Default value
Yes

GUI name
Calculate fatbands

Description
If True, BAND will compute the fat bands (only if BandStructure%Enabled is True).

The Fat Bands are the periodic equivalent of the Mulliken population analysis.

`KPathFinderConvention`

Type
Multiple Choice

Default value
Setyawan-Curtarolo

Options
[Setyawan-Curtarolo, Hinuma]

Description
This option determines how the path through the Brillouin zone is generated when using the automatic k-point mode.

Available options:

- `Setyawan-Curtarolo` (default for 1D and 2D lattices): Uses our built-in KPath program to find a path through high-symmetry points based on the method by Setyawan and Curtarolo (https://doi.org/10.1016/j.commatsci.2010.05.010). For 2D lattices, the path is derived from the intersection of the 3D Brillouin zone with a plane. For 1D lattices, the path is simply Gamma-Z.
- `Hinuma`: Uses the external SeeKPath utility to generate the k-path (https://github.com/giovannipizzi/seekpath and https://doi.org/10.1016/j.commatsci.2016.10.015).

`UseSymmetry`

Type
Bool

Default value
Yes

GUI name
Use symmetry

Description
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 key does not influence the symmetry of the band structure sampling. Only available for Setyawan and Curtarolo convention (see `KPathFinderConvention`).

`Basis`

Type
Block

Description
Definition of the basis set

`Core`

Type
Multiple Choice

Default value
Large

Options
[None, Small, Medium, Large]

GUI name
Frozen core

Description
Select the size of the frozen core you want to use.

Small, Medium, and Large will be interpreted within the basis sets available (of the selected quality), and might refer to the same core in some cases.

`Folder`

Type
String

Description
Path to a folder containing the basis set files. This can be used for special use-defined basis sets. Cannot be used in combination with 'Type'

`PerAtomType`

Type
Block

Recurring
True

Description
Defines the basis set for all atoms of a particular type.

`Core`

Type
Multiple Choice

Options
[None, Small, Medium, Large]

Description
Size of the frozen core.

`File`

Type
String

Description
The path to the basis set file. The path can be absolute or relative to $AMSRESOURCES/Band. Specifying the path to the basis file explicitly overrides the automatic basis file selection via the Type and Core subkeys.

`Symbol`

Type
String

Description
The symbol for which to define the basis set.

`Type`

Type
Multiple Choice

Options
[SZ, DZ, DZP, TZP, TZ2P, QZ4P]

Description
The basis sets to be used.

`PerRegion`

Type
Block

Recurring
True

Description
Defines the basis set for all atoms in a region. If specified, this overwrites the values set with the Basis%Type and Basis%PerAtomType keywords for atoms in that region. Note that if this keyword is used multiple times, the chosen regions may not overlap.

`Core`

Type
Multiple Choice

Default value
Large

Options
[None, Small, Medium, Large]

Description
Size of the frozen core.

`Region`

Type
String

Description
The identifier of the region for which to define the basis set. Note that this may also be a region expression, e.g. 'myregion+myotherregion' (the union of two regions).

`Type`

Type
Multiple Choice

Default value
DZ

Options
[SZ, DZ, DZP, TZP, TZ2P, QZ4P]

Description
The basis sets to be used.

`Type`

Type
Multiple Choice

Default value
DZ

Options
[SZ, DZ, DZP, TZP, TZ2P, QZ4P, STO/TZ2P, STO/SZ, STO/DZ, STO/DZP, STO/QZ4P, CORR/QZ6P, CORR/TZ3P, GTO/CC-PV5Z, GTO/DEF2-QZVPPD, GTO/CC-PV6Z, GTO/CC-PVQZ, GTO/CC-PVTZ, GTO/CC-PVDZ, GTO/DEF2-SVP, GTO/DEF2-TZVP, GTO/DEF2-TZVPP, GTO/DEF2-QZVP, GTO/AUG-CC-PVDZ, GTO/AUG-CC-PVTZ, GTO/AUG-CC-PVQZ, GTO/POB-TZVP]

GUI name
Basis set

Description
Select the basis set to use.

SZ : Single Z

DZ : Double Z

DZP : Double Z, 1 polarization function

TZP : Triple Z, 1 polarization function

TZ2P : Triple Z, 2 polarization functions

QZ4P : Quadruple Z, 4 polarization function

The basis set chosen will apply to all atoms in your structure. If a matching basis is not found a better type might be used.

`BeckeGrid`

Type
Block

Description
Options for the numerical integration grid, which is a refined version of the fuzzy cells integration scheme developed by Becke.

`Quality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, Basic, Normal, Good, VeryGood, Excellent]

Description
Quality of the integration grid. For a description of the various qualities and the associated numerical accuracy see reference. If 'Auto', the quality defined in the 'NumericalQuality' will be used.

`QualityPerRegion`

Type
Block

Recurring
True

Description
Sets the grid quality for all atoms in a region. If specified, this overwrites the globally set quality.

`Quality`

Type
Multiple Choice

Options
[Basic, Normal, Good, VeryGood, Excellent]

Description
The region's integration grid quality.

`Region`

Type
String

Description
The identifier of the region for which to set the quality.

`RadialGridBoost`

Type
Float

Default value
1.0

Description
The number of radial grid points will be boosted by this factor. Some XC functionals require very accurate radial integration grids, so BAND will automatically boost the radial grid by a factor 3 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.

`BerryPhase`

Type
Bool

Default value
No

Description
Boolean that determines whether the dipole as determined through the Berry phase approach should be calculated.

BField

Type
Block

Description
The effect of a magnetic filed can be approximated by the following potential: mu * sigma_i * B, where mu is the Bohr magneton, sigma_i are the Pauli matrices and B is the magnetic field

Bx

Type
Float

Default value
0.0

Unit
Tesla

Description
Value of the x component of the BField

By

Type
Float

Default value
0.0

Unit
Tesla

Description
Value of the y component of the BField

Bz

Type
Float

Default value
0.0

Unit
Tesla

Description
Value of the z component of the BField

Dipole

Type
Bool

Default value
No

GUI name
Bfield is: Atomic dipole

Description
Use an atomic dipole as magnetic field instead of a uniform magnetic field.

`DipoleAtom`

Type
Integer

Default value
1

GUI name
on atom number

Description
Atom on which the magnetic dipole should be centered (if using the dipole option)

`Method`

Type
Multiple Choice

Default value
NR_SDOTB

Options
[NR_SDOTB, NR_LDOTB, NR_SDOTB_LDOTB]

Description
There are two terms coupling to an external magnetic field.

One is the intrinsic spin of the electron, called S-dot-B, the other one is the orbital momentum call L-dot-B.

The L.B is implemented non-relativistically, using GIAOs in the case of a homogeneous magnetic field (not for the dipole case).

`Unit`

Type
Multiple Choice

Default value
tesla

Options
[tesla, a.u.]

Description
Unit of magnetic filed. The a.u. is the SI version of a.u.

`BZPath`

Type
Block

Description
Definition of the user-defined path in the Brillouin zone for band structure plotting.

`path`

Type
Non-standard block

Recurring
True

Description
Definition of the k-points in a path. The vertices of your path should be defined in fractional coordinates (wrt the reciprocal lattice vectors)

`Comment`

Type
Non-standard block

Description
The content of this block will be copied to the output header as a comment to the calculation.

`Convergence`

Type
Block

Description
Options and parameters related to the convergence behavior of the SCF procedure.

`Criterion`

Type
Float

Description
Criterion for termination of the SCF procedure. The default depends on the NumericalQuality and on the number of atoms in the system.

Can be used for EngineAutomations

`CriterionFactor`

Type
Float

Default value
1.0

Description
Multiply Criterion (which depends on system and quality) with this factor.

Can be used for EngineAutomations

`Degenerate`

Type
String

Default value
default

Description
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.

`ElectronicTemperature`

Type
Float

Default value
0.0

Unit
Hartree

Description
(KT) Specify this key for a gradient independent electronic temperature

`InitialDensity`

Type
Multiple Choice

Default value
rho

Options
[rho, psi, frompot]

Description
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`

Type
Bool

Default value
No

Description
If smoothing of occupations over nearly degenerate orbitals is applied (see Degenerate 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.

`ModestCriterion`

Type
Float

Default value
-1.0

Description
If this is specified band will consider the SCF converged if the error is below this criterion (after using the maximum number of iterations).

`NoDegenerate`

Type
Bool

Default value
No

Description
This key prevents any internal automatic setting of the key DEGENERATE.

NumBoltz

Type
Integer

Default value
10

Description
The electronic temperature is done with a Riemann Stieltjes numerical integration, between zero and one occupation. This defines the number of points to be used.

SpinFlip

Type
Integer List

GUI name
Flip spin for atoms

Description
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.

SpinFlipEnabled

Type
Bool

Default value
Yes

Description
If set to False, the keys SpinFlip and SpinFlipRegion are ignored. Only useful/convenient when trying to compare in a script the effect of spin flip.

SpinFlipRegion

Type
String

Recurring
True

GUI name
Flip spin for region

Description
Specify here the region 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

Type
Bool

Default value
Yes

Description
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 key.

`StartWithMaxSpinForSO`

Type
Bool

Default value
No

Description
Same as the StartWithMaxSpin option. In case of spin-orbit band always used to split the potential. Now will use maxspin in case of SpinFlip. With this option it will always do that.

`CPVector`

Type
Integer

Default value
128

GUI name
Vectorlength (blocksize)

Description
The code is vectorized and this key can be used to set the vector length

`DensityPlot`

Type
Non-standard block

Description
Plots of the density. Goes together with the Restart%DensityPlot and Grid keys.

`Dependency`

Type
Block

Description
Criteria for linear dependency of the basis and fit set

`AllowBasisDependency`

Type
Bool

Default value
Yes

Description
Project out the dependent part of the basis set (associated with small eigenvalues of the overlap matrix).

`Basis`

Type
Float

Default value
1e-08

GUI name
Dependency criterion

Description
Criteria for linear dependency of the basis: smallest eigenvalue of the overlap matrix of normalized Bloch functions.

`Core`

Type
Float

Default value
0.8

Description
The program verifies that the frozen core approximation is reasonable, by checking the smallest eigen value of the overlap matrix of the core (Bloch) orbitals (which should ideally be one) is bigger than this criterion.

`CoreValence`

Type
Float

Default value
1e-05

Description
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-corevalence

`Fit`

Type
Float

Default value
5e-06

Description
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.

`DIIS`

Type
Block

Description
Parameters for the DIIS procedure to obtain the SCF solution

`Adaptable`

Type
Bool

Default value
Yes

Description
Change automatically the value of dimix during the SCF.

CHuge

Type
Float

Default value
20.0

GUI name
No DIIS (but damping) when coefs >

Description
When the largest coefficient in the DIIS expansion exceeds this value, damping is applied

CLarge

Type
Float

Default value
20.0

GUI name
Reduce DIIS space when coefs >

Description
When the largest DIIS coefficient exceeds this value, the oldest DIIS vector is removed and the procedure re-applied

Condition

Type
Float

Default value
1000000.0

Description
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.

DiMix

Type
Float

Default value
0.2

GUI name
Bias DIIS towards latest vector with

Description
Mixing parameter for the DIIS procedure

DiMixMax

Type
Float

Default value
-1.0

Description
For adaptive diis: A negative value means automatic, see DiMixatnvctrx. If positive it is an absolute upper bound for (adaptive) dimix

DiMixMin

Type
Float

Default value
0.01

Description
An absolute lower bound for adaptive dimix.

NCycleDamp

Type
Integer

Default value
1

GUI name
Do not start DIIS before cycle

Description
Number of initial iterations where damping is applied, before any DIIS is considered

NVctrx

Type
Integer

Default value
20

GUI name
Size of DIIS space

Description
Maximum number of DIIS expansion vectors

Variant

Type
Multiple Choice

Default value
DIIS

Options
[DIIS, LISTi, LISTb, LISTd]

Description
Which variant to use. In case of problematic SCF convergence, first try MultiSecant, and if that does not work the LISTi is the advised method. Note: LIST is computationally more expensive per SCF iteration than DIIS.

DOS

Type
Block

Description
Density-Of-States (DOS) options

`CalcDOS`

Type
Bool

Default value
Yes

GUI name
Calculate DOS

Description
Whether or not to calculate the density of states.

`CalcPDOS`

Type
Bool

Default value
No

GUI name
Calculate PDOS

Description
Whether or not to calculate the partial DOS (projections on basis functions). This can be significantly more expensive than calculating the total DOS

`CalcPopulationAnalysis`

Type
Bool

Default value
Yes

GUI name
Calculate Mulliken charges

Description
Whether or not to calculate the population analysis. Population analysis can become very expensive when there are many symmetry operators, such as in a super cell.

`CompensateDeltaE`

Type
Bool

Default value
Yes

Description
Only relevant when IntegrateDeltaE=yes. If set to true then after integrating each interval over DeltaE the result is divided by DeltaE, so that the unit is DOS.

`DeltaE`

Type
Float

Default value
0.005

Unit
Hartree

GUI name
Delta E

Description
Energy step for the DOS grid. Using a smaller value (e.g. half the default value) will result in a finer sampling of the DOS.

`Energies`

Type
Integer

Description
Number of equidistant energy-values for the DOS grid. This keyword is superseded by the 'DeltaE' keyword.

`File`

Type
String

Description
Write the DOS (plain text format) to the specified file instead of writing it to the standard output.

`IntegrateDeltaE`

Type
Bool

Default value
Yes

Description
This subkey handles which algorithm is used to calculate the data-points in the plotted DOS. If 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)]. If 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.

`Max`

Type
Float

Unit
Hartree

Description
User defined upper bound energy (with respect to the Fermi energy)

`Min`

Type
Float

Unit
Hartree

Description
User defined lower bound energy (with respect to the Fermi energy)

`StoreCoopPerBasPair`

Type
Bool

Default value
No

GUI name
Calculate COOP

Description
Calculate the COOP (crystal orbital overlap population).

`DosBas`

Type
Non-standard block

Description
Used to specify the fragment basis for the DOS.

`DumpBasisOnly`

Type
Bool

Default value
No

Description
Dump basis and fit set files use for each atom.

`EffectiveMass`

Type
Block

Description
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. The easiest way to use this key is to enabled it without specifying any extra options.

`Enabled`

Type
Bool

Default value
No

GUI name
Effective mass

Description
Compute the EffectiveMass.

KPointCoord

Type
Float List

Unit
1/Bohr

Recurring
True

GUI name
At K-point

Description
Coordinate of the k-points for which you would like to compute the effective mass.

NumAbove

Type
Integer

Default value
1

GUI name
Include N bands above

Description
Number of bands to take into account above the Fermi level.

NumBelow

Type
Integer

Default value
1

GUI name
Include N bands below

Description
Number of bands to take into account below the Fermi level.

StepSize

Type
Float

Default value
0.001

Description
Size of the step taken in reciprocal space to perform the numerical differentiation

UseBandStructureInfoFromPath

Type
Bool

Default value
Yes

Description
The (automatic) location of the HOMO and LUMO can be determined via band interpolation, or from the path as used by the BandStructure feature. The latter works better when they are located on the path. See also comments in the BandStructure block. To reproduce results from before ams2025 set to no.

EFG

Type
Block

Description
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.

Enabled

Type
Bool

Default value
No

GUI name
EFG (electric field gradient): Calculate

Description
Compute the EFG tensor (for nuclear quadrupole interaction).

EigThreshold

Type
Float

Default value
0.01

Description
Threshold for printing the eigenvectors coefficients (Print Eigens)

ElectronHole

Type
Block

Description
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.

BandIndex

Type
Integer

Description
Which occupied band shall be depopulated.

SpinIndex

Type
Integer

Description
Defines the spin of the shifted electron (1 or 2).

EmbeddingPotential

Type
Block

Description
An external potential can be read in and will be added to the effective Kohn-Sham potential. It has to be on the becke grid

Filename

Type
String

Default value
Description
Name of the file containing the embedding potential.

PotentialName

Type
String

Default value
Description
Name of variable containing the potential.

EnforcedSpinPolarization

Type
Float

GUI name
Spin polarization

Description
Enforce a specific spin-polarization instead of occupying according to the aufbau principle. The spin-polarization is the difference between the number of alpha and beta electron.

Thus, a value of 1 means that there is one more alpha electron than beta electrons.

The number may be anything, including zero, which may be of interest when searching for a spin-flipped pair, that may otherwise end up in the (more stable) parallel solution.

ESR

Type
Block

Description
Zeeman g-tensor. 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 following options are necessary for ESR: 'Relativistic zora spin' and 'Kspace 1'

Enabled

Type
Bool

Default value
No

GUI name
ESR: g-tensor

Description
Compute Zeeman g-tensor.

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 following options are necessary for ESR: 'Relativistic zora spin' and 'Kspace 1'

Excitations

Type
Block

Description
Excitation energies: UV/Vis

Fermi

Type
Block

Description
Technical parameter used in determining the Fermi energy, which is carried out at each cycle of the SCF procedure.

Delta

Type
Float

Default value
0.0001

Description
Convergence criterion: upper and lower bounds for the Fermi energy and the corresponding integrated charge volumes must be equal within delta.

Eps

Type
Float

Default value
1e-10

Description
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.

MaxTry

Type
Integer

Default value
15

Description
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.

RefinePostSCFFactor

Type
Integer

Description
Use a finer k-grid after the scf to calculate a refined fermi level. Makes only sense for metals. Works like DoubleCount. Use 1,2,3

FermiSurface

Type
Block

Description
Calculation of the Fermi surface for metals

Enabled

Type
Bool

Default value
No

GUI name
Calculate Fermi surface

Description
Calculate the Fermi surface if the system has no band gap (i.e. is a metal). The result can be visualized with amsbands.

KIntegForSymmetricKGrid

Type
Integer

Default value
-1

Description
If the (default) regular k-grid is used, a symmetric one is created to determine the Fermi surface. If this key is not specified an automatic value of kInteg is used. Odd values trigger quadratic interpolation.

NMesh

Type
Integer

Default value
7

Description
Improves the matching of the interpolated quadratic surface. For better results it makes more sense to increases KIntegForSymmetricKGrid.

FormFactors

Type
Integer

Default value
2

Description
Number of stars of K-vectors for which the form factors are computed

Fragment

Type
Block

Recurring
True

Description
Defines a fragment. You can define several fragments for a calculation.

AtomMapping

Type
Non-standard block

Description
Format '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.

Filename

Type
String

Description
Filename of the fragment. Absolute path or path relative to the executing directory.

Labels

Type
Non-standard block

Description
This gives the possibility to introduce labels for the fragment orbitals. See examples.

FuzzyPotential

Type
Non-standard block

Description
Atomic (fuzzy cell) based, external, electric potential. See example.

FuzzyUnitCellGrid

Type
Block

Description
Undocumented.

AtomRadiusLSG

Type
Float

Default value
0.0

Description
Undocumented.

CellPartitionDelta

Type
Float

Default value
4.0

Description
Undocumented.

CellPartitionInterpolationCubic

Type
Bool

Default value
No

Description
Undocumented.

CellPartitionInterpolationMesh

Type
Integer

Default value
100

Description
Undocumented.

CellPartitionVersion

Type
Integer

Default value
2

Description
Undocumented.

CentralizeNaturalLSG

Type
Bool

Default value
No

Description
Undocumented.

`InterpolateCellPartition`

Type
Bool

Default value
No

Description
Undocumented.

`NumIntExtraL`

Type
Integer

Default value
0

Description
Undocumented.

`NumIntExtraRad`

Type
Integer

Default value
0

Description
Undocumented.

`PartitionFunctionTol`

Type
Float

Default value
1e-08

Description
Undocumented.

`PruneLatticeSummedGrid`

Type
Bool

Default value
Yes

Description
Undocumented.

`ReduceAccuracyLSG`

Type
Bool

Default value
No

Description
Undocumented.

SimpleLatticeSummedGrid

Type
Bool

Default value
No

Description
Undocumented.

Grid

Type
Block

Description
Options for the regular grid used for plotting (e.g. density plot). Used ICW the restart option.

ExtendX

Type
Float

Default value
0.0

Unit
Bohr

Description
Extend the default regular grid along the x-direction by the specified amount: [x_min, x_max] => [x_min - ExtendX/2, x_max + ExtendX/2].

ExtendY

Type
Float

Default value
0.0

Unit
Bohr

Description
Extend the default regular grid along the y-direction by the specified amount: [y_min, y_max] => [y_min - ExtendY/2, y_max + ExtendY/2].

ExtendZ

Type
Float

Default value
0.0

Unit
Bohr

Description
Extend the default regular grid along the z-direction by the specified amount: [z_min, z_max] => [z_min - ExtendZ/2, z_max + ExtendZ/2].

FileName

Type
String

Default value

Description
Read in the grid from a file. The file format of the grid is: three numbers per line (defining the x, y and z coordinates of the points).

Type

Type
Multiple Choice

Default value
coarse

Options
[coarse, medium, fine]

Description
The default regular grids.

UserDefined

Type
Non-standard block

Description
One can define the regular grid specification in this block. See example. Default unit is Bohr

GridBasedAIM

Type
Block

Description
Invoke the ultra fast grid based Bader analysis.

Enabled

Type
Bool

Default value
No

GUI name
Bader (AIM): Atomic properties

Description
Invoke the ultra fast grid based Bader analysis.

Iterations

Type
Integer

Default value
40

Description
The maximum number of steps that may be taken to find the nuclear attractor for a grid point.

SmallDensity

Type
Float

Default value
1e-06

Description
Value below which the density is ignored. This should not be chosen too small because it may lead to unassignable grid points.

UseStartDensity

Type
Bool

Default value
No

Description
Whether the analysis is performed on the startup density (True) or on the final density (False).

GrossPopulations

Type
Non-standard block

Description
Partial DOS (pDOS) are generated for the gross populations listed under this key. See example.

GW

Type
Block

Description
Perform a GW calculation. G0W0 is the default for GW%SelfConsistency.

AdaptiveMixing

Type
Float List

Description
Requests to use adaptive mixing instead of DIIS and sets the staring mixing parameter for mixing of Green's function in case of self-consistency.

Adapative mixing is recommended in case a qsGW calculation does not converge with DIIS.

It is ignored in non-selfconsistent calculation and overwritten by DIIS when DIIS is also present.

AnalyticalIntegration

Type
Block

Description
Use analytical integration to calculate the self-energy. Very slow, unless the system is very small but useful to check the accuracy of the frequency integration

Enabled

Type
Bool

Default value
No

GUI name
analytical integration

Description
Enable the calculation of the GW quasi-particle energies via analytical integration.

`SpectralFunctionResolution`

Type
Integer

Default value
800

Description
Number of points at which spectral function is evaluated.

`TDA`

Type
Bool

Default value
No

Description
Solve the linear response equations in the Tamm-Dancoff approximation.

`eta`

Type
Float

Default value
0.001

Description
Artificial (positive) broadening parameter for evaluation of self-energy in analytical integration.

Ideally should be as small as possible but this might lead to convergence issues in partially self-consistent approaches.

In this case, a value of up to 0.1 is possible.

`Converge`

Type
Block

Description
Sets convergence criteria for the GW calculation in self-consistent case

`Density`

Type
Float List

Default value
[1e-08, 1e-05]

Description
First Criterion for self-consistency procedure to terminate.

Criterion is the trace of the density matrix. Ignored in non-selfconsistent Calculation and in eigenvalue self-consistent GW

It is possible to run a qsGW calculation with an inner SCF loop which updates the static part of the elf-energy only. This can be useful to accelerate the convergence in case linear mixing is used. It is not recommended to use linear mixing, so it is also not recommended to use that inner loop as well. The second number in this list specifies the convergence criterion for the inner SCF loop.

HOMO

Type
Float

Default value
0.003

Unit
eV

GUI name
HOMO energy convergence

Description
Criterion for self-consistency procedure to terminate.

The self-consistent GW calculation terminates, when the difference between the HOMO QP energies between 2 consecutive iterations is below this number.

The LUMO energy converged faster than the HOMO energy so when the HOMO energy is converged according to this criterion, the LUMO energy will be converged as well.

In non-selfconsistent Calculation, this criterion is ignored.

DIIS

Type
Integer

Default value
10

Description
Requests to use DIIS. This is the Default. Number of expansion coefficients can be requested as well. Ignored in non-selfconsistent calculation

Enabled

Type
Bool

Default value
No

GUI name
Calculate GW quasi-particle energies

Description
Enable the calculation of the GW quasi-particle energies.

`LinearMixing`

Type
Float List

Description
Requests to use linear mixing instead of DIIS and sets the mixing parameter for linear mixing of Green's function in case of self-consistency.

It is ignored in non-selfconsistent calculation and overwritten by DIIS when DIIS is also present.

`LinearizeQPequations`

Type
Bool

Default value
No

Description
Instead of solving the non-linear QP equations in a G0W0 (or evGW calculation) by bisection exacly, linearize them by first-order Taylor expansion.

This is not recommended since it does not save computational time when used together with analytical continuation (as implemented in AMS). It might however be useful for benchmarking or for validating results.

If the results os the linearization differ by a lot (for instance, more than 0.1 eV in frontier QP energies) from the non-linearized results, this might indicate that the GW calculation is not reliable.

`Polarizability`

Type
Multiple Choice

Default value
RPA

Options
[RPA, BSE, G4W1, G4V1, TDHF]

Description
Sets the expression for the Polarizability used in the GW calculation.

RPA is the Default and amounts to a standard GW calculation.

BSE denotes screening in the Bethe-Salpeter-equation formalism.

`PrintAllSolutions`

Type
Bool

Default value
No

Description
Print out all solutions for all requested states. Detects multiple solutions of the QP equations.

`PrintSpectralFunction`

Type
Bool

Default value
No

Description
Plot the self-energy as a function of frequency. Automatically done in case of analytical continuation. However, this is expensive in the analytical integration formalism.

`QPHamiltonian`

Type
Multiple Choice

Default value
KSF2

Options
[KSF1, KSF2, SRG, LQSGW]

Description
The quasi-particle Hamiltonian can be constructed in different ways.

KSF1 refers to the original construction by Kotani, Van Schilfgaarde anf Faleev (KSF) which is also implemented in TURBOMOLE.

KSF2 refers to an alternative construction by KSF.

KSF1 is not recommended since it is numerically less stable than KSF2 in case analytical continuation is used (the default).

In case the analytical integration algorithm is used, only KSF1 is implemented. Therefore, make sure that KSF1 is specified. The results are typically very similar.

The QP energies at which the matrix elements are evaluated can be tweaked further, see the two subsequent keys: However, KSF2 is recommended since it typically leads to QP energies with the best agreement with experiment.

Ignored when not a quasi-particle self-consistent GW calculation is performed

`ScissorShift`

Type
Bool

Default value
No

Description
Only calculate the HOMO and LUMO QP energies and shift the remaining QP energies by the same amount.

This is a rather crude approximation and not recommended.

It might again be useful for benchmarking purposes.

`SelfConsistency`

Type
Multiple Choice

Default value
G0W0

Options
[G0W0, EVGW0, EVGW, QSGW0, QSGW]

Description
Sets the level of self-consistency in a GW calculation.

G0W0 calculates a one-shot, perturbative correction to the KS eigenvalues.

In evGW and evGW0, the quasi-particle energies are updated until self-consistency is reached.

evGW0 requests that the Green's function is evaluated self-consistently but not the screened interaction.

In qsGW, the density is updated as well, however, the self-energy is mapped to a static effective potential and the Dyson equation is solved by diagonalization instead of inversion. The results of a qsGW are independent of the choice of the underlying exchange-correlation functional and are usually the most accurate ones.

The same is done in qsGW0, but the screened interaction is not updated.

`SelfEnergy`

Type
Multiple Choice

Default value
GW

Options
[HF, GW, G3W2, SOSEX, GWGamma, G3W2dynamic, GWGammaInf, GWGammaInfDyn]

Description
Controls the form of the self-energy.

GW is the default and corresponds to the standard GW calculation.

G3W2 is a GW calculation plus a perturbative second-order statically screened exchange correction (second order expansion in the self-energy). Note, that there the self-energy is always static.

`nIterations`

Type
Integer List

Default value
[10]

GUI name
Number of iterations

Description
The maximum number of iterations within the (partially or fully) self-consistent GW calculation has to converge.

Ignored when Formalism is set to G0W0

`nLowest`

Type
Integer

Default value
1

GUI name
N Lowest

Description
Number of lowest occupied QP levels to be evaluated, overwrites nStates'

`nStates`

Type
Integer

Default value
5

GUI name
N states

Description
Number of Quasiparticle States to be printed to output.

The default is 5 states which in this case means that min(5, Number of particle states) occupied and min(5, Number of hole states) hole states are printed. The whole list of states can be printed by setting this parameter to -1'

`HubbardU`

Type
Block

Description
Options for Hubbard-corrected DFT calculations.

`Atom`

Type
Block

Recurring
True

Description
Specify Hubbard parameters (U,l) for a certain element

`Element`

Type
String

Description
Name of the element, such as Cu or Zn

`LValue`

Type
Multiple Choice

Default value
s

Options
[s, p, d, f]

Description
L value of the shell to apply the Hubbard model to

UValue

Type
Float

Default value
0.0

Unit
Hartree

Description
Hubbard U value.

PrintOccupations

Type
Bool

Default value
Yes

Description
Whether or not to print the occupations during the SCF.

Region

Type
Block

Recurring
True

Description
Specify Hubbard parameters (U,l) for all atoms in a certain region

LValue

Type
Multiple Choice

Default value
s

Options
[s, p, d, f]

Description
L value of the shell to apply the Hubbard model to

Name

Type
String

Description
Name of the region

UValue

Type
Float

Default value
0.0

Unit
Hartree

Description
Hubbard U value.

`Integration`

Type
Block

Description
Options for the Voronoi numerical integration scheme. Deprecated. Use BeckeGrid instead.

`AccInt`

Type
Float

Default value
3.5

Description
General parameter controlling the accuracy of the Voronoi integration grid. A value of 3 would be basic quality and a value of 7 would be good quality.

`IntegrationMethod`

Type
Multiple Choice

Default value
Becke

Options
[Becke, Voronoi]

Description
Choose the real-space numerical integration method. Note: the Voronoi integration scheme is deprecated.

`KGrpX`

Type
Integer

Default value
5

GUI name
Number of K-points at once

Description
Absolute upper bound on the number of k-points processed together. This only affects the computational performance.

`KSpace`

Type
Block

Description
Options for the k-space integration (i.e. the grid used to sample the Brillouin zone)

`Quality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, GammaOnly, Basic, Normal, Good, VeryGood, Excellent]

GUI name
K-space

Description
Select the quality of the K-space grid used to sample the Brillouin Zone. If 'Auto', the quality defined in the 'NumericalQuality' will be used. If 'GammaOnly', only one point (the gamma point) will be used.

The actual number of K points generated depends on this option and on the size of the unit cell. The larger the real space cell, the fewer K points will be generated.

The CPU-time and accuracy strongly depend on this option.

`Regular`

Type
Block

Description
Options for the regular k-space integration grid.

`NumberOfPoints`

Type
Integer List

Description
Use a regular grid with the specified number of k-points along each reciprocal lattice vector.

For 1D periodic systems you should specify only one number, for 2D systems two numbers, and for 3D systems three numbers.

`Symmetric`

Type
Block

Description
Options for the symmetric k-space integration grid.

`KInteg`

Type
Integer

GUI name
Accuracy

Description
Specify the accuracy for the Symmetric method.

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

2: linear tetrahedron method, coarsest spacing

3: quadratic tetrahedron method, coarsest spacing

4,6,… (even): linear tetrahedron method

5,7…. (odd): quadratic method

The tetrahedron method is usually by far inferior.

Type

Type
Multiple Choice

Default value
Regular

Options
[Regular, Symmetric]

GUI name
K-space grid type

Description
The type of k-space integration grid used to sample the Brillouin zone (BZ) used.

'Regular': simple regular grid.

'Symmetric': symmetric grid for the irreducible wedge of the first BZ (useful when high-symmetry points in the BZ are needed to capture the correct physics of the system, graphene being a notable example).

LDOS

Type
Block

Description
Local Density-Of-States information. This can be used to generate STM images in the Tersoff-Hamann approximation (see https://doi.org/10.1103/PhysRevB.31.805)

DeltaNeg

Type
Float

Default value
0.0001

Unit
Hartree

Description
Lower bound energy (Shift-DeltaNeg)

DeltaPos

Type
Float

Default value
0.0001

Unit
Hartree

Description
Upper bound energy (Shift+DeltaPos)

Shift

Type
Float

Default value
0.0

Unit
Hartree

Description
The energy bias with respect to the Fermi level.

MBPT

Type
Block

Description
Technical aspects of the MP2 algorithm.

Dependency

Type
Bool

Default value
Yes

Description
If true, to improve numerical stability, almost linearly-dependent combination of basis functions are removed from the Green's function that are used in the MBPT equations. Disabling this key is strongly discouraged. Its value can however be changed. The key to adjust this value is RiHartreeFock%DependencyThreshold

ExcludeCore

Type
Bool

Description
If active, excludes core states from the calculation of the optimal imaginary time and frequency grids.

The core states are still included in all parts of the calculations.

In case a frozen care calculation is performed, this key is ignored.

For MP2 and double hybrid calculation, it defaults to false. For RPA and GW calculations, it defaults to true.

FitSetQuality

Type
Multiple Choice

Default value
Auto

Options
[Auto, VeryBasic, Basic, Normal, Good, VeryGood]

Description
Specifies the fit set to be used in the MBPT calculation.

'Normal' quality is generally sufficient for basis sets up to and including TZ2P.

For larger basis sets (or for benchmarking purposes) a 'VeryGood' fit set is recommended. Note that the FitSetQuality heavily influences the computational cost of the calculation.

If not specified or 'Auto', the RIHartreeFock%FitSetQuality is used.

`Formalism`

Type
Multiple Choice

Default value
Auto

Options
[Auto, RI, LT, All]

Description
Specifies the formalism for the calculation of the MP2 correlation energy.

'LT' means Laplace Transformed MP2 (also referred to as AO-PARI-MP2),

'RI' means that a conventional RI-MP2 is carried out.

If 'Auto', LT will be used in case of DOD double hybrids and SOS MP2, and RI will be used in all other cases.

'All' means that both RI and LT formalisms are used in the calculation.

For a RPA or GW calculation, the formalism is always LT, irrespective of the formalism specified with this key.

`FrequencyGridType`

Type
Multiple Choice

Default value
LeastSquare

Options
[LeastSquare, GaussLegendre]

Description
Use Gauss-legendre grid for imaginary frequency integration in RPA and GW calculations instead of the usually used Least-Square optimized ones. Has the advantage that it can be systematically converged and an arbitrary number of grid points can be used. Typically more grid points will be needed to get the same level of accuracy. However, the convergence of the results with the size of the grid can be more systematic. These grids can only be used when Formalism is set to RI.

`IntegrationQuality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood]

Description
Specifies the integration quality to be used in the MBPT calculation. If not specified, the RIHartreeFock%IntegrationQuality is used.

`SigmaFunctionalParametrization`

Type
Multiple Choice

Default value
S1re

Options
[W1, W2, S1, S2, S1re]

Description
Only relevant if a sigma-functional calculation is performed. Possible choices for the parametrization of the sigma-functional. Not all options are supported for all functionals.

`ThresholdQuality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
Controls the distances between atomic centers for which the product of two basis functions is not fitted any more. Especially for spatially extended, large systems, 'VERYBASIC' and 'BASIC' can lead to large computational savings, but the fit is also more approximate. If not specified, the RIHartreeFock%ThresholdQuality is used.

`UseScaledZORA`

Type
Bool

Default value
Yes

Description
If true, use the scaled ZORA orbital energies instead of the ZORA orbital energies in the MBPT equations.

`frozencore`

Type
Bool

Default value
No

Description
Freeze core states in correlation part of MBPT calculation

`nCore`

Type
Integer

Default value
0

GUI name
Number of core levels

Description
Number of core states which will be excluded from the correlated calculation.

Will be ignored if frozencore is false.

In case nothing is specified, the number of core levels will be determined automatically.

Needs to be smaller than the number of occupied states.

nFreqIntegration

Type
Integer

Default value
32

GUI name
N freq points for G3W2 integration

Description
Number of imaginary frequency points for G3W2 integration

nFrequency

Type
Integer

Default value
12

GUI name
N freq points

Description
Number of imaginary frequency points. This key is only relevant for RPA and GW and will be ignored if used in an AO-PARI-MP2 calculation. 12 Points is the default for a RPA calculation. It is technically possible to use a different number of imaginary frequency points than for imaginary time. The maximum number of points which can be requested for imaginary frequency integration is 42. Important note: The computation time and memory requirements roughly scale linearly with the number of imaginary frequency points. However, memory can be an issue for RPA and GW when the number of imaginary frequency points is high. In case a job crashes, it is advised to increase the number of nodes since the necessary memory distributes over all nodes.

nLambda

Type
Integer

Default value
1

GUI name
Number of lambda points

Description
Size of coupling constant integration grid for SOSEX variants in RPA. Default is 4 points

nTime

Type
Integer

GUI name
Number of time points

Description
Number of imaginary time points (only relevant in case the Laplace Transformed (LT) formalism is used).

In the many-body-perturbation theory module in ADF, the polarizability (or Kohn-Sham density response function) is evaluated in imaginary time to exploit sparsity in the AO basis. For MP2, this is often referred to as a Laplace transform. For MP2, 9 points are the default. This is a safe choice, guaranteeing accuracies higher than 1 Kj/mol for most systems (For many simple organic systems, 6 points are sufficient for good accuracy).

Only for systems with a very small HOMO-LUMO gap or low-lying core states (heavy elements starting from the 4th row of the periodic table) more points might be necessary.

In principle, the same considerations apply for RPA and GW as well, however, the accuracy requirements are somewhat higher and 12 point are the default for RPA. In a GW calculation, the number of points is adjusted according to the numerical quality. Using less than 9 points is strongly discouraged except for the simplest molecules.

In ADF2019, it can happen that the algorithm determining the imaginary time grid does not converge. In this case, the usual reason is that the number of points is too small and more points need to be specified. Starting from AMS2020, this does not happen any more. In case the imaginary time grid does not converge, the number of points is automatically adjusted until it does.

The computation time of AO-PARI-MP2, RPA, and GW scales linearly with the number of imaginary time points.

`useGreenXgrids`

Type
Bool

Default value
No

Description
Use GreenX library to generate grid points. This is recommended for larger number of grid points (> 20). Up to 34 points can be requested.

`MolecularNMR`

Type
Block

Description
Options for the calculations of the NMR shielding tensor for molecules, excluding periodic systems. Implements the Schreckenbach method like ADF.

`Enabled`

Type
Bool

Default value
No

Description
Compute NMR shielding.

`MultiSecantConfig`

Type
Block

Description
Parameters for the Multi-secant SCF convergence method.

`CMax`

Type
Float

Default value
20.0

GUI name
Max coeff

Description
Maximum coefficient allowed in expansion

`InitialSigmaN`

Type
Float

Default value
0.1

GUI name
Initial

Description
This is a lot like a mix factor: bigger means bolder

`MaxSigmaN`

Type
Float

Default value
0.3

GUI name
Max

Description
Upper bound for the SigmaN parameter

`MaxVectors`

Type
Integer

Default value
20

GUI name
Number of cycles to use

Description
Maximum number of previous cycles to be used

`MinSigmaN`

Type
Float

Default value
0.01

GUI name
Min

Description
Lower bound for the SigmaN parameter

NEGF

Type
Block

Description
Options for the NEGF (non-equilibrium green function) transport calculation.

AlignChargeTol

Type
Float

Default value
0.1

Description
In an alignment run you want to get the number of electrons in the center right. This number specifies the criterion for that.

AlignmentFile

Type
String

Default value
Description
Band result file (.rkf) corresponding to the alignment calculation.

Alpha

Type
Float

Default value
1e-05

Description
A charge error needs to be translated in a potential shift. DeltaV = alpha * DeltaQ

ApplyShift1

Type
Bool

Default value
Yes

Description
Apply the main shift, obtained from comparing matrix elements in the leads with those from the tight-binding run. Strongly recommended.

ApplyShift2

Type
Bool

Default value
Yes

Description
Apply the smaller alignment shift. This requires an extra alignment run. Usually this shift is smaller.

`AutoContour`

Type
Bool

Default value
Yes

Description
Use automatic contour integral.

`BiasPotential`

Type
Float

Default value
0.0

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

`BoundOccupationMethod`

Type
Integer

Default value
1

Description
See text. Only relevant with NonEqDensityMethod equal 2 or 3.

`CDIIS`

Type
Bool

Default value
No

Description
Make the normal DIIS procedure aware of the align charge error

`CheckOverlapTol`

Type
Float

Default value
0.01

Description
BAND checks how well the TB overlap matrix 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.

ContourQuality

Type
Multiple Choice

Default value
good

Options
[basic, normal, good, verygood]

Description
The density matrix is calculated numerically via a contour integral. Changing the quality influences the number of points. This influences a lot the performance.

DEContourInt

Type
Float

Default value
-1.0

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

DERealAxisInt

Type
Float

Default value
-1.0

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

DeltaPhi0

Type
Float

Default value
0.0

Description
Undocumented.

DeltaPhi1

Type
Float

Default value
0.0

Description
Undocumented.

DoAlignment

Type
Bool

Default value
No

Description
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.

EMax

Type
Float

Default value
5.0

Unit
eV

Description
The maximum energy for the transmission grid (with respect to the Fermi level of the lead)

EMin

Type
Float

Default value
-5.0

Unit
eV

Description
The minimum energy for the transmission grid (with respect to the Fermi level of the lead)

Eta

Type
Float

Default value
1e-05

Description
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.

IgnoreOuterLayer

Type
Bool

Default value
Yes

Description
Whether or not to ignore the outer layer.

KT

Type
Float

Default value
0.001

Description
k-Boltzmann times temperature.

LeadFile

Type
String

Default value
Description
File containing the tight binding representation of the lead.

NE

Type
Integer

Default value
100

Description
The number of energies for the transmission energy grid.

NonEqDensityMethod

Type
Integer

Default value
1

Description
See text.

SGFFile

Type
String

Default value
Description
The result from the SGF program. Contains the Fermi energy of the lead.

YContourInt

Type
Float

Default value
0.3

Description
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.

YRealaxisInt

Type
Float

Default value
1e-05

Description
The non-Equilibrium density is calculated near the real axis.

`NeutralizingDensity`

Type
Multiple Choice

Default value
None

Options
[None, rho(atoms), rho(valence/atoms), rho(neutralizing/atoms), rho(homogeneous)]

Description
For charged systems an artificial compensating density can be used to make it neutral again. This fictitious density only affects the Coulomb potential.

For charged periodic systems neutralization is required, as otherwise the Coulomb potential diverges.

`NeutralizingDensityDetails`

Type
Block

Description

`DiffuseFactor`

Type
Float

Default value
1.0

Description
The bigger this number, the more diffuse (extended) the neutralizing density becomes. Works only for rho(neutralizing/atoms)

`HomogeneousDensity`

Type
Block

Description
xxx

`Origin`

Type
Float List

Default value
[0.0, 0.0, 0.0]

Unit
Bohr

Description

`Range`

Type
Float

Default value
10.0

Unit
Bohr

Description

Width

Type
Float

Default value
1.0

Unit
Bohr

Description

NewResponse

Type
Block

Description
The TD-CDFT calculation to obtain the dielectric function is computed when this block is present in the input. Several important settings can be defined here.

ActiveESpace

Type
Float

Default value
5.0

Unit
eV

GUI name
Active energy space

Description
Modifies the energy threshold (DeltaE^{max}_{thresh} = omega_{high} + ActiveESpace) for which single orbital transitions (DeltaEpsilon_{ia} = Epsilon_{a}^{virtual} - Epsilon_{i}^{occupied}) are taken into account.

ActiveXYZ

Type
String

Default value
t

Description
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.

DensityCutOff

Type
Float

Default value
0.001

GUI name
Volume cutoff

Description
For 1D and 2D systems the unit cell volume is undefined. Here, the volume is calculated as the volume bordered by the isosurface for the value DensityCutoff of the total density.

EShift

Type
Float

Default value
0.0

Unit
eV

GUI name
Shift

Description
Energy shift of the virtual crystal orbitals.

FreqHigh

Type
Float

Default value
3.0

Unit
eV

Description
Upper limit of the frequency range for which response properties are calculated (omega_{high}).

FreqLow

Type
Float

Default value
1.0

Unit
eV

Description
Lower limit of the frequency range for which response properties are calculated. (omega_{low})

NFreq

Type
Integer

Default value
5

Description
Number of frequencies for which a linear response TD-CDFT calculation is performed.

NewResponseKSpace

Type
Block

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

Eta

Type
Float

Default value
1e-05

Description
Defines the small, finite imaginary number i*eta which is necessary in the context of integration weights for single-particle transitions in reciprocal space.

SubSimp

Type
Integer

Default value
3

Description
determines into how many sub-integrals 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. Note: the computing time for the weights is linear for 1D, quadratic for 2D and cubic for 3D!

NewResponseSCF

Type
Block

Description
Details for the linear-response self-consistent optimization cycle. Only influencing the NewResponse code.

Bootstrap

Type
Integer

Default value
0

Description
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!

COApproach

Type
Bool

Default value
Yes

Description
The program automatically decides to calculate the integrals and induced densities via the Bloch expanded atomic orbitals (AO approach) or via the cyrstal orbitals (CO approach). The option COApproach overrules this decision.

COApproachBoost

Type
Bool

Default value
No

GUI name
CO Approach Boost

Description
Keeps the grid data of the Crystal Orbitals in memory.

Requires significantly more memory for a speedup of the calculation. One might have to use multiple computing nodes to not run into memory problems.

Criterion

Type
Float

Default value
0.001

Description
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.

DIIS

Type
Block

Description
Parameters influencing the DIIS self-consistency method

Enabled

Type
Bool

Default value
Yes

Description
If not enabled simple mixing without DIIS acceleration will be used.

MaxSamples

Type
Integer

Default value
10

Description
Specifies the maximum number of samples considered during the direct inversion of iteration of subspace (DIIS) extrapolation of the atomic charges during the SCC iterations. A smaller number of samples potentially leads to a more aggressive convergence acceleration, while a larger number often guarantees a more stable iteration. Due to often occurring linear dependencies within the set of sample vectors, the maximum number of samples is reached only in very rare cases.

MaximumCoefficient

Type
Float

Default value
10.0

Description
When the diis expansion coefficients exceed this threshold, the solution is rejected. The vector space is too crowded. The oldest vector is discarded, and the expansion is re-evaluated.

MinSamples

Type
Integer

Default value
-1

Description
When bigger than one, this affects the shrinking of the DIIS space on linear dependence. It will not reduce to a smaller space than MinSamples unless there is extreme dependency.

MixingFactor

Type
Float

Default value
0.2

Description
The parameter used to mix the DIIS linear combination of previously sampled atomic charge vectors with an analogous linear combination of charge vectors resulting from population analysis combination. It can assume real values between 0 and 1.

LowFreqAlgo

Type
Bool

Default value
Yes

GUI name
Low Frequency Algorithm

Description
Numerically more stable results for frequencies lower than 1.0 eV. Note: for a graphene monolayer the conical intersection results in a very small band gap (zero band gap semi-conductor). This leads ta a failing low frequency algorithm. One can then chose to use the algorithm as originally proposed by Kootstra by setting the input value to *false*. But, this can result in unreliable results for frequencies lower than 1.0 eV!

NCycle

Type
Integer

Default value
20

GUI name
Cycles

Description
Number of SCF cycles for each frequency to be evaluated.

XC

Type
Integer

Default value
1

Description
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).

NMR

Type
Block

Description
Options for the calculations of the NMR shielding tensor.

Correction_r

Type
Bool

Default value
Yes

Description
Undocumented.

Enabled

Type
Bool

Default value
No

Description
Compute NMR shielding.

MS0

Type
Float

Default value
0.01

Description
Undocumented.

NMRAtom

Type
Integer

Default value
0

Description
The index of the atom atom (in input order) for which NMR should be computed.

Numeric

Type
Bool

Default value
No

Description
Undocumented.

Original

Type
Bool

Default value
No

Description
Undocumented.

Print_jp

Type
Bool

Description
Print paramagnetic current.

SuperCell

Type
Bool

Default value
Yes

Description
This is the switch between the two methods, either the super cell (true), or the single-dipole method (false)

Test

Type
Bool

Description
Key for printing all intrinsic tensors.

Test_E

Type
Bool

Description
Test of energy levels.

Test_S

Type
Bool

Description
Test of overlap matrix.

UseSharedMemory

Type
Bool

Default value
Yes

Description
Whether or not to use shared memory in the NMR calculation.

NOCVdRhoPlot

Type
Non-standard block

Description
Goes together with the Restart%NOCVdRhoPlot and Grid keys. See example.

NOCVOrbitalPlot

Type
Non-standard block

Description
Goes together with the Restart%NOCVOrbitalPlot and Grid keys. See example.

NuclearModel

Type
Multiple Choice

Default value
PointCharge

Options
[PointCharge, Gaussian, Uniform]

Description
Specify what model to use for the nucleus.

For the Gaussian model the nuclear radius is calculated according to the work of Visscher and Dyall (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))

`NUElstat`

Type
Integer

Default value
50

Description
Number of outward (parabolic) integration points (for elliptical integration of the electrostatic interaction)

`NumericalQuality`

Type
Multiple Choice

Default value
Normal

Options
[Basic, Normal, Good, VeryGood, Excellent]

Description
Set the quality of several important technical aspects of a BAND calculation (with the notable exception of the basis set). It sets the quality of: BeckeGrid (numerical integration), ZlmFit (density fitting), KSpace (reciprocal space integration), and SoftConfinement (basis set confinement). Note: the quality defined in the block of a specific technical aspects supersedes the value defined in NumericalQuality (e.g. if I specify 'NumericalQuality Basic' and 'BeckeGrid%Quality Good', the quality of the BeckeGrid will be 'Good')

`NVElstat`

Type
Integer

Default value
80

Description
Number of angular (elliptic) integration points (for elliptical integration of the electrostatic interaction)

`Occupations`

Type
Non-standard block

Description
Allows one to input specific occupations numbers. Applies only for calculations that use only one k-point (i.e. pseudo-molecule calculations). See example.

`OldResponse`

Type
Block

Description
Options for the old TD-CDFT implementation.

`Berger2015`

Type
Bool

Default value
No

Description
Use the parameter-free polarization functional by A. Berger (Phys. Rev. Lett. 115, 137402). This is possible for 3D insulators and metals. Note: The evaluation of response properties using the Berger2015 is recommend for 3D systems only!

`CNT`

Type
Bool

Description
Use the 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.

`CNVI`

Type
Float

Default value
0.001

Description
The first convergence criterion for the change in the fit coefficients for the fit functions, when fitting the density.

`CNVJ`

Type
Float

Default value
0.001

Description
the second convergence criterion for the change in the fit coefficients for the fit functions, when fitting the density.

`Ebndtl`

Type
Float

Default value
0.001

Unit
Hartree

Description
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.

Enabled

Type
Bool

Default value
No

Description
If true, the response function will be calculated using the old TD-CDFT implementation

Endfr

Type
Float

Default value
3.0

Unit
eV

Description
The upper bound frequency of the frequency range over which the dielectric function is calculated

Isz

Type
Integer

Default value
0

Description
Integer indicating whether or not scalar zeroth order relativistic effects are included in the TD-CDFT calculation. 0 = relativistic effects are not included, 1 = relativistic effects are included. The current implementation does NOT work with the option XC%SpinOrbitMagnetization equal NonCollinear

Iyxc

Type
Integer

Default value
0

Description
integer for printing yxc-tensor (see http://aip.scitation.org/doi/10.1063/1.1385370). 0 = not printed, 1 = printed.

NewVK

Type
Bool

Description
Use the slightly modified version of the VK kernel (see https://aip.scitation.org/doi/10.1063/1.1385370). When using this option one uses effectively the static option, even for metals, so one should check carefully the convergence with the KSPACE parameter.

Nfreq

Type
Integer

Default value
5

Description
the number of frequencies for which a linear response TD-CDFT calculation is performed.

`QV`

Type
Bool

Description
Use the QV parametrization for the longitudinal and transverse kernels of the XC kernel of the homogeneous electron gas. Use this in conjunction with the NewVK option. (see reference).

`Shift`

Type
Float

Default value
0.0

Unit
eV

Description
energy shift for the virtual crystal orbitals.

`Static`

Type
Bool

Description
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.

`Strtfr`

Type
Float

Default value
1.0

Unit
eV

Description
is the lower bound frequency of the frequency range over which the dielectric function is calculated.

`OrbitalPlot`

Type
Non-standard block

Description
Goes together with the Restart%OrbitalPlot and Grid keys. See Example.

`Output`

Type
Block

Description
Control the output.

`Print`

Type
Block

Recurring
True

Description

`Level`

Type
Multiple Choice

Options
[None, Error, Warning, Minimal, Normal, Detail, TooMuchDetail]

Description

`Section`

Type
Multiple Choice

Options
[Prepare, SCF, Properties]

Description

`OverlapPopulations`

Type
Non-standard block

Description
Overlap population weighted DOS (OPWDOS), also known as the crystal orbital overlap population (COOP).

`PEDA`

Type
Bool

Default value
No

Description
If present in combination with the fragment block, the decomposition of the interaction energy between fragments is invoked.

`PEDANOCV`

Type
Block

Description
Options for the decomposition of the orbital relaxation (pEDA).

`EigvalThresh`

Type
Float

Default value
0.001

GUI name
Use NOCVs with ev larger than

Description
The threshold controls that for all NOCV deformation densities with NOCV eigenvalues larger than EigvalThresh the energy contribution will be calculated and the respective pEDA-NOCV results will be printed in the output

`Enabled`

Type
Bool

Default value
No

GUI name
Perform PEDA-NOCV analysis

Description
If true in combination with the fragment blocks and the pEDA key, the decomposition of the orbital relaxation term is performed.

`PeriodicRIHartreeFock`

Type
Block

Description
Technical options for periodic Hartree Fock.

`AllowLinearScaling`

Type
Bool

Default value
Yes

Description
Use function ranges and multipoles.

`AllowMultipoleExpansion`

Type
Bool

Default value
No

Description
Use the multipole expansion if possible.

`AllowPartialMultipoleExpansion`

Type
Bool

Default value
No

Description
Use the partial multipole expansion if possible. This has only effect if AllowMultipoleExpansion=yes.

AnalyzeCells

Type
Bool

Default value
No

Description
Determine the cell loop ranges needed and stop.

AvoidPeriodicKMatR

Type
Bool

Default value
No

Description
Undocumented.

AvoidPeriodicPMatR

Type
Bool

Default value
No

Description
Undocumented.

CellLoopPrecision

Type
Multiple Choice

Default value
Full

Options
[Minimal, Medium, Full]

Description
The range of the density matrix, R(P), combined with the range of basis products, R(C), determines the cell range contributing to the exchange matrix. The cell range used is.

Minimal: R(P).

Medium: R(P)+R(C).

All: R(P)+2R(C)

ExactLMaxMultiPerAtom

Type
Bool

Default value
No

Description
Used to be a hardcodes lMaxMulti=3. If specified it is determined per atom from the RIHF fit functions.

`MergeIterators`

Type
Bool

Default value
No

Description
If true, calculate terms I,II,III,and IV in parallel, otherwise term IV is done separately.

`PrintKDistanceDecay`

Type
Bool

Default value
No

Description
The decay of the norm of K with respect to the distance cell order.

`PrintKShellDecay`

Type
Bool

Default value
No

Description
The decay of the norm of K with respect to the shell cell order.

`PrintKTopoDecay`

Type
Bool

Default value
No

Description
The decay of the norm of K with respect to the topological cell order.

`RemapPMatR`

Type
Bool

Default value
No

Description
Possibly move elements from one cell to another, minimizing the distance between the atom pairs.

RemapPMatRNew

Type
Bool

Default value
No

Description
Possibly move elements from one cell to another, minimizing the distance between the atom pairs.

SaveKMatR

Type
Bool

Default value
No

Description
Undocumented.

SortFitFunctions

Type
Bool

Default value
No

Description
Sort the RIHartreeFock fit functions per atom, based on the range. This is needed when allowing for partial multipole atom pairs.

UseHalfKGrid

Type
Bool

Default value
No

Description
Let all RIHF settings be based on the assumption that the k-grid was twice as coarse. What will be more accurate is P(R), but it gets the range as if the coarser k-grid was used.

UseHelper

Type
Bool

Default value
No

Description
Provide the RIHartreeFock with knowledge about loop bounds.

UseInverseCellSymmetry

Type
Bool

Default value
No

Description
If true, only half of the K(R) is calculated, and the rest is obtained from K(-R)=transpose(K(R)).

`UseLatticeShiftForPairs`

Type
Bool

Default value
No

Description
Apply a pair based minimum image convention.

`UseM1V`

Type
Bool

Default value
No

Description
Multiply M1 by V, being faster for term II.

`PeriodicSolvation`

Type
Block

Description
Additional options for simulations of periodic structures with solvation.

`NStar`

Type
Integer

Default value
4

Description
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.

`RemovePointsWithNegativeZ`

Type
Bool

Default value
No

GUI name
Only above slab

Description
Whether the 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 (in the Z direction), then this option should be set to 'True'

`SymmetrizeSurfacePoints`

Type
Bool

Default value
Yes

Description
Whether or not the COSMO point should be symmetrized

PopThreshold

Type
Float

Default value
0.01

Description
Threshold for printing Mulliken population terms. Works with 'Print orbpop'

PotentialNoise

Type
Float

Default value
0.0001

Description
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.

Print

Type
String

Recurring
True

Description
One or more strings (separated by blanks) 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.

Programmer

Type
Block

Description
Miscellaneous technical options.

SharedMemorySandwichingThreshold

Type
Integer

Default value
5000

GUI name
Shared mem sandwiching thld

Description
When the nr. of basis functions exceeds this threshold shared memory will be used to calculate matrix elements.

Unless UseSharedMemoryForSandwiching is explicitly set in the input.

`StoreDOSPerBas`

Type
Bool

Default value
Yes

Description
Whether or not to store the parial DOS per basis function. This allows you to view any partial DOS with amsspectra and amsbands. Requires the CalcPDOS option to be on.

`StoreOrbitals`

Type
Bool

Default value
Yes

Description
Copy information on band.rkf needed for orbital plotting and restarts. This can be a lot of information. DOS and BandStructure require StoreOrbitals=true.

`UpdateSTDVec`

Type
Bool

Default value
Yes

Description
Shift atoms so that center of mass is at zero. As a results the detected symmetry may be higher.

`UseSharedMemoryForSandwiching`

Type
Bool

Default value
Yes

GUI name
Use shared memory

Description
When calculating matrix elements the array will be shared. This saves memory at the cost of locking overhead.

If not specified this will depend on the threshold SharedMemorySandwichingThreshold

`UseTurnoverRuleForXcMatrix`

Type
Bool

Default value
No

Description
Undocumented.

Usesharedmemory

Type
Bool

Default value
Yes

GUI name
Use shared memory

Description
When running more then one task, share memory between those tasks. This saves a lot of memory.

Only disable it in case of problems.

PropertiesAtNuclei

Type
Non-standard block

Description
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: vxc[rho(fit)], rho(fit), rho(scf), v(coulomb/scf), rho(deformation/fit), rho(deformation/scf).

RadialDefaults

Type
Block

Description
Options for the logarithmic radial grid of the basis functions used in the subprogram Dirac

NR

Type
Integer

Default value
3000

Description
Number of radial points. With very high values (like 30000) the Dirac subprogram may not converge.

NRPerType

Type
Integer List

Description
If present overrides NR. The list needs to be as long as there are atom types

RMax

Type
Float

Default value
100.0

Unit
Bohr

Description
Upper bound of the logarithmic radial grid

RMin

Type
Float

Default value
1e-06

Unit
Bohr

Description
Lower bound of the logarithmic radial grid

RMinPerType

Type
Float List

Unit
Bohr

Description
If specified overrides RMin. The list needs to be as long as there are atom types (different elements)

Relativity

Type
Block

Description
Options for relativistic effects.

Level

Type
Multiple Choice

Default value
Scalar

Options
[None, Scalar, Spin-Orbit]

GUI name
Relativity (ZORA)

Description
None: No relativistic effects.

Scalar: Scalar relativistic ZORA.

This option comes at very little cost.

SpinOrbit: 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 key.

ResponseInducedDensityPlot

Type
Non-standard block

Description
Goes together with Restart%ResponseInducedDensityPlot and Grid.

Restart

Type
Block

Description
Tells the program that it should restart with the restart file, and what to restart.

BandStructure

Type
Bool

Default value
No

Description
Calculate the band structure from a previous calculation. Does not work with model potentials and Hubbard.

CheckAtomicPositions

Type
Bool

Default value
Yes

Description
If set to True: For restarting the SCF the atomic positions will be checked, and may not deviate too much.

DOS

Type
Bool

Default value
No

Description
Calculate the DOS from a previous calculation. Does not work with model potentials and Hubbard.

DensityPlot

Type
Bool

Default value
No

Description
Goes together with the DensityPlot block and Grid blocks

File

Type
String

Default value
GUI name
Restart using

Description
Name of the restart file. The file should be a band.rkf file from a previous run.

LoadEigenSystem

Type
Bool

Default value
No

GUI name
Load: eigen system

Description
At each step of the SCF load the section eigensystem from the restart file, forcing constant eigenvalues and vectors.

NOCVOrbitalPlot

Type
Bool

Default value
No

Description
Goes together with the NOCVOrbitalPlot and Grid blocks.

NOCVdRhoPlot

Type
Bool

Default value
No

Description
Goes together with the NOCVdRhoPlot and Grid blocks.

OrbitalPlot

Type
Bool

Default value
No

Description
Goes together with the OrbitalPlot and Grid

ResponseInducedDensityPlot

Type
Bool

Default value
No

Description
Goes together with the ResponseInducedDensityPlot and Grid blocks.

SCF

Type
Bool

Default value
No

GUI name
Restart: SCF

Description
Continue the SCF procedure using the orbital coefficients and occupations from the restart file.

UseDensityMatrix

Type
Bool

Default value
No

Description
If set to True: For restarting the SCF the density matrix will be used. Requires you to set 'Save DensityMatrix' in the previous run.

VTKFile

Type
String

Default value
Description
If specified a vtk file with be created with this name. If the extension is '.txt', a text file is created. Setting it to 'CUBE' one or more (one for each component) files in the cube format are generated with an automatic naming scheme.

VoronoiGrid

Type
Bool

Default value
No

Description
Copy the section Num In Params to the current file.

RIHartreeFock

Type
Block

Description

DependencyCoreRange

Type
Float

Description
Basis functions may be given a core character based on the range. For now only active in Band and only if present in the input

DependencyThreshold

Type
Float

Default value
0.001

Description
To improve numerical stability, almost linearly-dependent combination of basis functions are removed from the Hartree-Fock exchange matrix.

If you obtain unphysically large bond energy in an Hybrid calculation, or an unphysically low correlation energy in an RPA, MP2, or double hybrid calculation, you might try setting the DependencyThreshold to a larger value (e.g. 3.0E-3)

Note, that in GW calculations and GW-BSE calculations the default for this key is 5.0e-3.

ExplicitThresholds

Type
Block

Description
Override the thresholds as implied by the ThresholdQuality.

Basis

Type
Float

Description
Threshold for the basis functions.

Fit

Type
Float

Description
Threshold for the fit functions.

Potential

Type
Float

Description
Threshold for the potential of the functions.

FitGenerationDetails

Type
Block

Description
Technical details about how the RI Hartree-Fock fit functions are generated.

BoostL

Type
Bool

Default value
No

Description
Add extra max(l)+1 diffuse function

When l denotes the highest angular momentum present in the primary basis,

FromBasisProducts will generate auxiliary fit functions with up to 2l angular momentum.

When this key is set to true, the maximum angular momentum in the auxiliary fit set becomes 2l+2.

Typically, this option is not needed and when precision issues arise, it is rather advised to adjust the OneCenterDependencyThreshold key to a smaller value.

LapackWorkAround

Type
Bool

Default value
No

Description
GetFitFunctionsForAtomType diagonalization done with Lapack instead of Scalapack

Method

Type
Multiple Choice

Default value
Auto

Options
[Auto, FromBasisProducts]

Description
The way in which fit functions are generated. The main distinction is whether it depends on the basis functions used.

When FromBasisProducts is used, the auxiliary basis is generated directly from the products of primary basis functions.

This has the advantage that the auxiliary fit adapts automatically to the basis set size.

Especially for basis sets of QZ quality or larger, this is often necessary to obtain highly precise correlation energies using RPA or double hybrids

FromBasisProducts option is also useful for GW or BSE calculations with basis sets of QZ quality or larger.

OneCenterDependencyThreshold

Type
Float

Default value
1e-08

Description
This key is only active when FromBasisProducts is chosen as method to generate the auxiliary basis.

This threshold controls the size, and at the samw time, the precision of the auxiliary basis set. A smaller number leads to a larger auxiliary fit set.

The default value of 1e-8 is typically sufficient to converge correlation energies and QP energies to a very high precision.

It corresponds to an auxiliary basis which is typically 8-9 times larger than the primary basis.

`UseBandRadialGrid`

Type
Bool

Default value
Yes

Description
Only applies to band. The band logarithmic grid ranges (by default) from 1e-6 to 100 with 3000 points. Otherwise 300 points will be used.

For 0-periodicity (molecules) it is advisable to set this key to false since lots of memory is needed to evaluate all necessary integrals.

`FitSetQuality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, VeryBasic, Basic, Normal, Good, VeryGood, Excellent, FromBasisProducts]

Description
The quality of auxiliary fit set employed in the RI scheme.

If 'Auto', the value of the RIHartreeFock Quality option will be used.

Normal quality is generally sufficient for basis sets up to and including TZ2P.

For larger basis sets (or for benchmarking purposes) a VeryGood fit set is recommended.

Note that the FitSetQuality heavily influences the computational cost of the calculation.

`IntegrationQuality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
Quality of the numerical integration for evaluating the integrals between basis functions and fit functions. If IntegrationQuality is not defined in input, the value defined in RI-HartreeFock%Quality will be used.

`Quality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
Numerical accuracy of the RI procedure. If 'Auto', the quality specified in the 'NumericalQuality' will be used.

`QualityPerRegion`

Type
Block

Recurring
True

Description
Sets the fit-set quality for all atoms in a region. If specified, this overwrites the globally set quality.

`Quality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
This region's quality of the auxiliary fit set employed in the RI scheme.

`Region`

Type
String

Description
The identifier of the region for which to set the quality.

`ResponseQuality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
Numerical accuracy of the RI procedure for the Response module.

`ThresholdQuality`

Type
Multiple Choice

Options
[VeryBasic, Basic, Normal, Good, VeryGood, Excellent]

Description
Linear scaling thresholds (also used for determining at what range the multiple approximation is used). To disable all linear scaling thresholds set this to Excellent.

UseMe

Type
Bool

Default value
Yes

Description
Set to False if you want to use the old RI scheme (ADF only)

Save

Type
String

Recurring
True

Description
Save scratch files or extra data that would be otherwise deleted at the end of the calculation. e.g. 'TAPE10' (containing the integration grid) or 'DensityMatrix'

SCF

Type
Block

Description
Controls technical SCF parameters.

Eigenstates

Type
Bool

Description
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).

Iterations

Type
Integer

Default value
300

GUI name
Maximum number of cycles

Description
The maximum number of SCF iterations to be performed.

Method

Type
Multiple Choice

Default value
MultiStepper

Options
[DIIS, MultiSecant, MultiStepper]

Description
Choose the general scheme used to converge the density in the SCF. In case of scf problems one can try the MultiSecant alternative at no extra cost per SCF cycle. For more details see the DIIS and MultiSecantConfig block.

`Mixing`

Type
Float

Default value
0.075

Description
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.

`MultiStepperPresetPath`

Type
String

Default value
DFTB/default2023.inc

Description
Name of file containing a SCFMultiStepper key block. This will be used if no Explicit SCF-MultiStepper block is in the input, and Method=MultiStepper.

If the path is not absolute, it is relative to $AMSHOME/data/presets/multi_stepper'

`PMatrix`

Type
Bool

Description
If present, evaluate the charge density from the P-matrix. See also the key Eigenstates.

`PrintAllOccupiedBands`

Type
Bool

Default value
Yes

Description
When printing the ranges of the bands, include all occupied ones.

`PrintAllVirtualBands`

Type
Bool

Default value
No

Description
When printing the ranges of the bands, include all virtual ones.

`PrintAlwaysBandRanges`

Type
Bool

Default value
No

Description
Normally the ranges of the bands are only printed at the last SCF cycle

`Rate`

Type
Float

Default value
0.99

Description
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 key Degenerate of block Convergence) or, if everything seems to fail, it will stop

`SCFMultiStepper`

Type
Block

Description
To solve the self-consistent problem multiple steppers can be tried during stints using the ones that give the best progress.

`AlwaysChangeStepper`

Type
Bool

Default value
No

Description
When the progress is fine there is no reason to change the stepper. In practice this is always set to true, because also the Stepper%ExpectedSlope can be used to achieve similar behavior.

`ErrorGrowthAbortFactor`

Type
Float

Default value
1000.0

Description
Abort stint when the error grows too much, compared to the error at the start of the stint.

FractionalStepFactor

Type
Float

Default value
-1.0

Description
Multiply the step by this factor. If smaller than zero this is not used.

MinStintCyclesForAbort

Type
Integer

Default value
0

Description
Look at ErrorGrowthAbortFactor only when a number of steps has been completed since the start of the stint. A value of 0 means always.

Stepper

Type
Block

Recurring
True

Description
??

AbortSlope

Type
Float

Default value
100.0

Description
If the slope (at the end of a stint) is larger than this: abort the stepper

DIISStepper

Type
Block

Description
DIIS stepper

EDIISAlpha

Type
Float

Default value
0.01

Description
The extra energy vector is weighed by this factor. .

MaxCoefficient

Type
Float

Default value
20.0

Description
The largest allowed value of the expansion coefficients. If exceed the number of vectors is reduces until the criterion is met.

`MaxVectors`

Type
Integer

Default value
10

Description
Maximum number of previous densities to be used (size of the history).

`MinVectors`

Type
Integer

Default value
-1

Description
Try to prevent to make nVectors shrink below this value, by allowing for significantly larger coefficients.

`Mix`

Type
Float

Default value
0.2

Description
Also known as greed. It determines the amount of output density to be used. May be changed by the MixAdapter.

`ErrorGrowthAbortFactor`

Type
Float

Default value
-1.0

Description
Abort stint when the error grows too much, compared to the error at the start of the stint. Overrides global ErrorGrowthAbortFactor when set to a value > 0

`ExpectedSlope`

Type
Float

Default value
-100.0

Description
If the slope of the total SCF is better than this keep on going.

`FractionalStepFactor`

Type
Float

Default value
-1.0

Description
Multiply the step by this factor. If smaller than zero this is not used.

`MaxInitialError`

Type
Float

Description
Only use the stepper when error is smaller than this.

`MaxIterationNumber`

Type
Integer

Default value
-1

Description
Stepper will only be active for iterations smaller than this number. (Negative value means: Ignore this option)

`MaxStintNumber`

Type
Integer

Default value
-1

Description
Stepper will only be active for stints smaller than this number. (Negative value means: Ignore this option)

`MinInitialError`

Type
Float

Description
Only use the stepper when error is larger than this.

`MinIterationNumber`

Type
Integer

Default value
-1

Description
Stepper will only be active for iterations larger than this number.

MinStintCyclesForAbort

Type
Integer

Default value
0

Description
Look at ErrorGrowthAbortFactor only when a number of steps has been completed since the start of the stint. A value of 0 means always. Overrides global value.

MinStintNumber

Type
Integer

Default value
-1

Description
Stepper will only be active for stints larger than this number.

MixAdapter

Type
Block

Description
Generic mix adapter

ErrorGrowthPanicFactor

Type
Float

Default value
10.0

Description
When the error increases more than this factor, this mix is reduced a lot.

GrowthFactor

Type
Float

Default value
1.1

Description
When the mix is considered too low it is multiplied by this factor. Otherwise it is divided by it.

MaxMix

Type
Float

Default value
0.3

Description
Do not grow the mix above this value.

MinMix

Type
Float

Default value
0.1

Description
Do not shrink the mix below this value.

NTrialMixFactors

Type
Integer

Default value
3

Description
Only used with Type=Trial. Must be an odd number.

TrialMode

Type
Multiple Choice

Default value
CurrentMixCentered

Options
[CurrentMixCentered, FullRange]

Description
How are the NTrialMixFactors chosen?

Type

Type
Multiple Choice

Default value
Error

Options
[Error, Energy, UnpredictedStep, Trial]

Description
Adapt the mix factor based on the observed progress (slope).

MixStepper

Type
Block

Description
Simple mixing stepper, only using the previous (in/out) density.

Mix

Type
Float

Default value
0.1

Description
???.

MultiSecantStepper

Type
Block

Description
Multi secant stepper.

MaxCoefficient

Type
Float

Default value
20.0

Description
???.

MaxVectors

Type
Integer

Default value
10

Description
???.

Mix

Type
Float

Default value
0.2

Description
???.

Variant

Type
Multiple Choice

Default value
MSB2

Options
[MSB1, MSB2, MSR1, MSR1s]

Description
There are several version of the Multi secant method.

StintLength

Type
Integer

Description
Override global StintLength.

`StintLength`

Type
Integer

Default value
10

Description
A stepper is active during a number of SCF cycles, called a stint.

`UsePreviousStintForErrorGrowthAbort`

Type
Bool

Default value
No

Description
The error is normally checked against the first error of the stint. With this option that will be the one from the previous stint, if performed with the same stepper.

`VSplit`

Type
Float

Default value
0.05

Description
To disturb degeneracy of alpha and beta spin MOs the value of this key is added to the beta spin potential at the startup.

`Screening`

Type
Block

Description
For the periodic solvation potential and for the old (not default anymore) fitting method, BAND performs lattice summations which are in practice truncated. The precision of the lattice summations is controlled by the options in this block.

`CutOff`

Type
Float

Description
Criterion for negligibility of tails in the construction of Bloch sums. Default depends on Accuracy.

`DMadel`

Type
Float

Description
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

NoDirectionalScreening

Type
Bool

Description
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. Screening is only applied in the periodicity directions. This key restores the original situation: screening in all directions

RCelx

Type
Float

Description
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

Type
Float

Description
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.

SelectedAtoms

Type
Integer List

Description
With this key you can select atoms. This has an effect on a few of options, like NMR and EFG.

Skip

Type
String

Recurring
True

Description
Skip the specified part of the Band calculation (expert/debug option).

SoftConfinement

Type
Block

Description
In order to make the basis functions more compact, the radial part of the basis functions is multiplied by a Fermi-Dirac (FD) function (this 'confinement' is done for efficiency and numerical stability reasons). 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.

Delta

Type
Float

Unit
Bohr

Description
Explicitly specify the delta parameter of the Fermi-Dirac function (if not specified, it will be 0.1*Radius).

`Quality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, Basic, Normal, Good, VeryGood, Excellent]

GUI name
Confinement

Description
In order to make the basis functions more compact, the radial part of the basis functions is multiplied by a Fermi-Dirac (FD) function (this 'confinement' is done for efficiency and numerical stability reasons). 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.

This key sets the two parameters 'Radius' and 'Delta'.

Basic: Radius=7.0, Delta=0.7;

Normal: Radius=10.0, Delta=1.0;

Good: Radius=20.0, Delta=2.0;

VeryGood and Excellent: no confinement at all.

If 'Auto', the quality defined in the 'NumericalQuality' will be used.

`Radius`

Type
Float

Unit
Bohr

Description
Explicitly specify the radius parameter of the Fermi-Dirac function.

`Solvation`

Type
Block

Description
Options for the COSMO (Conductor like Screening Model) solvation model.

`CVec`

Type
Multiple Choice

Default value
EXACT

Options
[EXACT, FITPOT]

GUI name
Calculate Coulomb interaction

Description
Choose how to calculate the Coulomb interaction matrix between the molecule and the point charges on the surface:

- EXACT: use exact density, and integrate against the potential of the point charges. This may have inaccuracies when integration points are close to the point charges.
- FITPOT: evaluate the molecular potential at the positions of the point charges, and multiply with these charges.

`Charge`

Type
Block

Description
Select the algorithm to determine the charges.

`Conv`

Type
Float

Default value
1e-08

Description
Charge convergence threshold in iterative COSMO solution.

`Corr`

Type
Bool

Default value
Yes

GUI name
Correct for outlying charge

Description
Correct for outlying charge.

`Iter`

Type
Integer

Default value
1000

Description
Maximum number of iterations to solve COSMO equations.

`Method`

Type
Multiple Choice

Default value
CONJ

Options
[CONJ, INVER]

GUI name
Charge determination method

Description
INVER: matrix inversion, CONJ: biconjugate gradient method.

The CONJ method is guaranteed to converge with small memory requirements and is normally the preferred method.

`Enabled`

Type
Bool

Default value
No

GUI name
Include COSMO solvation

Description
Use the Conductor like Screening Model (COSMO) to include solvent effects.

`Radii`

Type
Non-standard block

Description
The values are the radii of the atomic spheres. If not specified the default values are those by Allinger. Format: 'AtomType value'. e.g.: 'H 0.7'

`SCF`

Type
Multiple Choice

Default value
VAR

Options
[VAR, PERT, NONE]

GUI name
Handle charges

Description
Determine the point charges either Variational (VAR) or after the SCF as a Perturbation (PERT).

`Solvent`

Type
Block

Description
Solvent details

`Del`

Type
Float

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

Emp

Type
Float

Description
Emp is the empirical scaling factor x for the energy scaling.

Eps

Type
Float

Description
User-defined dielectric constant of the solvent (overrides the Eps value of the solvent defined in 'Name')

Name

Type
Multiple Choice

Default value
Water

Options
[AceticAcid, Acetone, Acetonitrile, Ammonia, Aniline, Benzene, BenzylAlcohol, Bromoform, Butanol, isoButanol, tertButanol, CarbonDisulfide, CarbonTetrachloride, Chloroform, Cyclohexane, Cyclohexanone, Dichlorobenzene, DiethylEther, Dioxane, DMFA, DMSO, Ethanol, EthylAcetate, Dichloroethane, EthyleneGlycol, Formamide, FormicAcid, Glycerol, HexamethylPhosphoramide, Hexane, Hydrazine, Methanol, MethylEthylKetone, Dichloromethane, Methylformamide, Methypyrrolidinone, Nitrobenzene, Nitrogen, Nitromethane, PhosphorylChloride, IsoPropanol, Pyridine, Sulfolane, Tetrahydrofuran, Toluene, Triethylamine, TrifluoroaceticAcid, Water]

GUI name
Solvent

Description
Name of a pre-defined solvent. A solvent is characterized by the dielectric constant (Eps) and the solvent radius (Rad).

Rad

Type
Float

Unit
Angstrom

Description
User-defined radius of the solvent molecule (overrides the Rad value of the solvent defined in 'Name').

Surf

Type
Multiple Choice

Default value
Delley

Options
[Delley, Wsurf, Asurf, Esurf, Klamt]

GUI name
Surface type

Description
Within the COSMO model the molecule is contained in a molecule shaped cavity.

Select one of the following surfaces to define the cavity:

- Wsurf: Van der Waals surface
- Asurf: solvent accessible surface
- Esurf: solvent excluding surface
- Klamt: Klamt surface
- Delley: Delley surface.

`SolvationSM12`

Type
Block

Description
Options for Solvation Model 12 (SM12).

`ARO`

Type
Float

Default value
0.0

Description
Square of the fraction of non-hydrogen atoms in the solvent that are aromatic carbon atoms (carbon aromaticity)

`Acid`

Type
Float

Default value
0.82

Description
Abraham hydrogen bond acidity parameter

`Base`

Type
Float

Default value
0.35

Description
Abraham hydrogen bond basicity parameter

BornC

Type
Float

Default value
3.7

Description
Coulomb constant for General Born Approximation

BornRadiusConfig

Type
Block

Description

MaxCellDistance

Type
Float

Default value
30.0

Unit
Bohr

Description
Max distance from the centra cell used when computing the Born radii for periodic systems

PointsPerBohr

Type
Integer

Default value
10

Description

UseLegendreGrid

Type
Bool

Default value
Yes

Description

Chgal

Type
Float

Default value
2.474

Description
Exponential of Pauli's bond order

Cust

Type
String

Description
Custom solvent input

Debug

Type
String

Description
Prints a lot of information about every pass on CDS and ENP code, keywords: ENP, CDS

EPS

Type
Float

Default value
78.36

Description
The dielectric constant

Enabled

Type
Bool

Default value
No

GUI name
Include SM12 solvation

Description
Whether to use the Solvation Model 12 (SM12) in the calculation.

HALO

Type
Float

Default value
0.0

Description
Square of the fraction of non-hydrogen atoms in the solvent molecule that are F, Cl, or Br (electronegative halogenicity)

Kappa

Type
Float

Default value
0.0

Description
Factor for Debye screening

PostSCF

Type
Bool

Default value
No

Description
Whether to apply the solvation potential during the SCF or only calculate the solvation energy after the SCF.

PrintSM12

Type
Bool

Default value
No

Description
Prints out an in-depth breakdown of solvation energies

RadSolv

Type
Float

Default value
0.4

Description
The radius distance between the solute and solvent

Ref

Type
Float

Default value
1.3328

Description
Refractive index of solvent

Solv

Type
Multiple Choice

Default value
WATER

Options
[ACETICACID, ACETONITRILE, ACETOPHENONE, ANILINE, ANISOLE, BENZENE, BENZONITRILE, BENZYLALCOHOL, BROMOBENZENE, BROMOETHANE, BROMOFORM, BROMOOCTANE, N-BUTANOL, SEC-BUTANOL, BUTANONE, BUTYLACETATE, N-BUTYLBENZENE, SEC-BUTYLBENZENE, T-BUTYLBENZENE, CARBONDISULFIDE, CARBONTETRACHLORIDE, CHLOROBENZENE, CHLOROFORM, CHLOROHEXANE, M-CRESOL, CYCLOHEXANE, CYCLOHEXANONE, DECALIN, DECANE, DECANOL, 1-2-DIBROMOETHANE, DIBUTYLETHER, O-DICHLOROBENZENE, 1-2-DICHLOROETHANE, DIETHYLETHER, DIISOPROPYLETHER, N-N-DIMETHYLACETAMIDE, N-N-DIMETHYLFORMAMIDE, 2-6-DIMETHYLPYRIDINE, DIMETHYLSULFOXIDE, DODECANE, ETHANOL, ETHOXYBENZENE, ETHYLACETATE, ETHYLBENZENE, FLUOROBENZENE,

1-FLUORO-N-OCTANE, HEPTANE, HEPTANOL, HEXADECANE, HEXADECYLIODIDE, HEXANE, HEXANOL, IODOBENZENE, ISOBUTANOL, ISOOCTANE, ISOPROPANOL, ISOPROPYLBENZENE, P-ISOPROPYLTOLUENE, MESITYLENE, METHANOL, METHOXYETHANOL, METHYLENECHLORIDE, N-METHYLFORMAMIDE, 2-METHYLPYRIDINE, 4-METHYL-2-PENTANONE, NITROBENZENE, NITROETHANE, NITROMETHANE, O-NITROTOLUENE, NONANE, NONANOL, OCTANE, OCTANOL, PENTADECANE, PENTANE, PENTANOL, PERFLUOROBENZENE, PHENYLETHER, PROPANOL, PYRIDINE, TETRACHLOROETHENE, TETRAHYDROFURAN, TETRAHYDROTHIOPHENEDIOXIDE, TETRALIN, TOLUENE, TRIBUTYLPHOSPHATE, TRIETHYLAMINE, 1-2-4-TRIMETHYLBENZENE, UNDECANE, WATER, XYLENE, 1-2-DIBROMOETHANE_WATER, 1-2-DICHLOROETHANE_WATER, BENZENE_WATER, CARBONTETRACHLORIDE_WATER, CHLOROBENZENE_WATER, CHLOROFORM_WATER, CYCLOHEXANE_WATER, DIBUTYLETHER_WATER, DIETHYLETHER_WATER, ETHYLACETATE_WATER, HEPTANE_WATER, HEXANE_WATER, NITROBENZENE_WATER, OCTANOL_WATER]

GUI name
Solvent

Description
List of predefined solvents

`Tens`

Type
Float

Default value
103.62

Description
Macroscopic surface tension of the solvent at the air/solvent interface at 298K (cal*mol^-1*Ang^-2)

`TopologicalExtrapolation`

Type
Block

Description
Method to extrapolate the long range Coulomb potential, needed for periodic calculations

`FirstCell`

Type
Integer

Default value
5

Description
First cell for the topological extrapolation of the long range part of the Coulomb Potential.

`LastCell`

Type
Integer

Default value
10

Description
Last cell for the topological extrapolation of the long range part of the Coulomb Potential.

Order

Type
Integer

Default value
3

Description
Order of the topological extrapolation of the long range part of the Coulomb Potential.

StopAfter

Type
String

Default value
BAND

Description
Specifies that the program is stopped after execution of a specified program-part (subroutine).

StoreHamAsMol

Type
Bool

Default value
No

Description
Undocumented, used for (at least) NEGF.

StoreHamiltonian

Type
Bool

Description
Undocumented.

StoreHamiltonian2

Type
Bool

Default value
No

Description
determine the tight-binding representation of the overlap an fock matrix. Used for (at least) NEGF.

StrainDerivatives

Type
Block

Description
Undocumented.

Analytical

Type
Bool

Description
Whether or not to use analytical strain derivatives. By default this is determined automatically, and used if possible.

`AnalyticalElectrostatic`

Type
Bool

Default value
No

Description
Undocumented.

`Analyticalkinetic`

Type
Bool

Default value
No

Description
Undocumented.

`Analyticalpulay`

Type
Bool

Default value
No

Description
Undocumented.

`Analyticalxc`

Type
Bool

Default value
No

Description
Undocumented.

`Celltopoorder`

Type
Integer

Default value
20

Description
Undocumented.

`Coreorthoption`

Type
Integer

Default value
2

Description
Undocumented.

`Fitrho0numintextrarad`

Type
Integer

Default value
0

Description
Undocumented.

`Fitrho0prune`

Type
Bool

Default value
Yes

Description
Undocumented.

`Kinviadagger`

Type
Bool

Default value
No

Description
Undocumented.

`Lmaxmultipoleexpansion`

Type
Integer

Default value
4

Description
Undocumented.

`Naiveelstat`

Type
Bool

Default value
No

Description
Undocumented.

`Numericaldefdef`

Type
Bool

Default value
Yes

Description
Undocumented.

Numericaldefdeflong

Type
Bool

Default value
No

Description
Undocumented.

Pairgridlowerangularorder

Type
Integer

Default value
5

Description
Undocumented.

Pairgridradpointsincrease

Type
Integer

Default value
0

Description
Undocumented.

Renormalizechargefitrho0

Type
Bool

Default value
No

Description
Undocumented.

Shiftmultipoleorigin

Type
Bool

Default value
Yes

Description
Undocumented.

Skipinlgwsmodule

Type
Bool

Default value
Yes

Description
Undocumented.

`SubtractAtomicXC`

Type
Bool

Default value
Yes

Description
Derive stress from xc energy difference of molecule versus atoms.

`Usesymmetry`

Type
Bool

Default value
No

Description
Undocumented.

`Usevstrainderrho`

Type
Bool

Default value
No

Description
Undocumented.

`fitrho0numintextral`

Type
Integer

Default value
0

Description
Undocumented.

`SubSymmetry`

Type
Integer List

Description
The indices of the symmetry operators to maintain.

`Tails`

Type
Block

Description
Ignore function tails.

Bas

Type
Float

Default value
1e-06

GUI name
Basis functions

Description
Cut off the basis functions when smaller than the specified threshold.

Title

Type
String

Default value

Description
Title of the calculation, which will be printed in the output file.

Unrestricted

Type
Bool

Default value
No

Description
Controls whether Band should perform a spin-unrestricted calculation. Spin-unrestricted calculations are computationally roughly twice as expensive as spin-restricted.

UnrestrictedOnlyReference

Type
Bool

Default value
No

Description
Undocumented.

UnrestrictedReference

Type
Bool

Default value
No

Description
Undocumented.

UnrestrictedStartup

Type
Bool

Default value
No

Description
Undocumented.

`UseInversionSymmetryInReciprocalSpace`

Type
Bool

Default value
Yes

GUI name
Use inversion sym in recip space

Description
Whether to use inversion symmetry in reciprocal space. This is almost always a valid assumption.

`UseSymmetry`

Type
Bool

Default value
Yes

Description
Whether or not to exploit symmetry during the calculation.

`XC`

Type
Block

Description
Exchange Correlation functionals

`DFTHalf`

Type
Block

Description
DFT-1/2 method for band gaps. See PRB vol 78,125116 2008. This method can be used in combination with any functional. For each active atom type (see ActiveAtomType) Band will perform SCF calculations at different screening cut-off values (see ScreeningCutOffs) and pick the cut-off value that maximizes the band gap. If multiple atom types are active, the screening cut-off optimizations are done one type at the time (in the same order as the ActiveAtomType blocks appear in the input).

`ActiveAtomType`

Type
Block

Recurring
True

Description
Use the DFT-1/2 method for the atom-type specified in this block.

`AtomType`

Type
String

Description
Atom-type to use. You can activate all atom-types by specifying 'All'.

`IonicCharge`

Type
Float

Default value
0.5

Description
The amount of charge to be removed from the atomic HOMO.

`ScreeningCutOffs`

Type
Float List

Default value
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0]

Unit
Bohr

Description
List of screening cut-offs (to screen the asymptotic IonicCharge/r potential). Band will loop over these values and find the cut-off that maximizes the band-gap. If only one number is provided, Band will simply use that value.

`Enabled`

Type
Bool

Default value
No

GUI name
Use method

Description
Whether the DFT-1/2 method will be used.

`Prepare`

Type
Bool

Default value
No

Description
Analyze the band structure to determine reasonable settings for an DFT-1/2 calculation. If this is possible the list of active atom types is written to the output. This can be used in a next run as the values for ActiveAtomType. The DFTHalf%Enabled key should be set to false

`SelfConsistent`

Type
Bool

Default value
Yes

Description
Apply the extra potential during the SCF, or only afterwards. Applying DFT-1/2 only post SCF increases the band gap, compared to the self-consistent one.

`DoubleHybrid`

Type
String

Description
Specifies the double hybrid functional that should be used during the SCF.

`EmpiricalScaling`

Type
Multiple Choice

Default value
None

Options
[None, SOS, SCS, SCSMI]

Description
Calculate the (SOS/SCS/SCSMI)-MP2 correlation energy.

`GALITSKIIMIGDAL`

Type
Bool

Default value
No

Description
Calculate the Galitskii-Migdal correlation energy after the SCF is completed.

`GLLBKParameter`

Type
Float

Default value
0.382

Description
K parameter for the GLLB functionals. See equation (20) of the paper.

`HartreeFock`

Type
Bool

Default value
No

Description
Stand alone HF calculation.

`MP2`

Type
Bool

Default value
No

Description
Calculate the MP2 correlation energy after the HF SCF is completed.

RPA

Type
Multiple Choice

Default value
None

Options
[None, Direct, Sigma, SOSEX, SOSSX]

Description
Specifies that RPA is used an possibly also a post-RPA method. By default, direct RPA is used

RangeSeparation

Type
String

Default value
Description
Intended to be used with HartreeFock (or hybrid functionals). Example: OMEGA= 0.110000 ALPHA= 0.250000 BETA= -0.250000 ERF-SHORTRANGE

diracgga

Type
String

Default value
Description
GGA for the dirac .

dispersion

Type
String

Default value
DEFAULT

Description
The dispersion correction model to be used.

gga

Type
String

Default value
NONE

Description
GGA XC functional.

lda

Type
String

Default value
VWN

Description
LDA XC functional.

libxc

Type
String

Default value
NONE

Description
Functional using the LicXC library.

libxcdensitythreshold

Type
Float

Default value
1e-10

Description
Density threshold for LibXC functionals.

metagga

Type
String

Default value
NONE

Description
MetaGG XC functional.

model

Type
String

Default value
LB94

Description
Model potential. The possible choices are LB94, GLLB-SC, BGLLB-VWN, and BGLLB-LYP

spinorbitmagnetization

Type
String

Default value
collinearz

Description
Type of Spin-Orbit magnetization.

tb_mbjafactor

Type
Float

Default value
-1.23456789

Description
a parameter for the TB-MBJ model potential.

tb_mbjbfactor

Type
Float

Default value
-1.23456789

Description
b parameter for the TB-MBJ model potential..

tb_mbjcfactor

Type
Float

Default value
-1.23456789

Description
c parameter for the TB-MBJ model potential..

tb_mbjefactor

Type
Float

Default value
-1.23456789

Description
e parameter for the TB-MBJ model potential..

usexcfun

Type
Bool

Default value
No

Description
Whether ot not the XCFun library should be used.

xcfun

Type
Bool

Default value
No

Description
Functional for the XCFun library.

ZlmFit

Type
Block

Description
Options for the density fitting scheme 'ZlmFit'.

`AllowBoost`

Type
Bool

Default value
Yes

Description
Allow automatic atom-dependent tuning of maximum l of spherical harmonics expansion. Whether or not this boost is needed for a given atom is based on an heuristic estimate of how complex the density around that atom is.

`DensityThreshold`

Type
Float

Description
Threshold below which the electron density is considered to be negligible. Depends on Quality and is normally 1.0e-7

`FGaussianW`

Type
Float

Default value
1.0

Description
Only for 3D periodic systems. Width of the Gaussian functions replacing the S and P Zlms for Fourier transform.

`FGridSpacing`

Type
Float

Description
Only for 3D periodic systems. Spacing for the Fourier grid. By default, this depends on the quality.

`FKSpaceCutOff`

Type
Float

Description
Only for 3D periodic systems. Cut-off of the grid in k-space for the Fourier transform.

`FirstTopoCell`

Type
Integer

Default value
5

Description
First cell for the topological extrapolation of the long range part of the Coulomb Potential.

`LMargin`

Type
Integer

Description
User-defined l-margin, i.e., l_max for fitting is max(lMargin + l_max_basis_function, 2*l_max_basis_function). Depends on Quality and normally is 4

`LastTopoCell`

Type
Integer

Default value
10

Description
Last cell for the topological extrapolation of the long range part of the Coulomb Potential.

`NumStarsPartitionFun`

Type
Integer

Default value
5

Description
Number of cell stars to consider when computing the partition function.

`OrderTopoTrick`

Type
Integer

Default value
3

Description
Order of the topological extrapolation of the long range part of the Coulomb Potential.

`PartitionFunThreshold`

Type
Float

Default value
0.0

Description
Threshold for the partition functions: if an integration point has a partition function weight smaller than this threshold, it will be discarded.

`Quality`

Type
Multiple Choice

Default value
Auto

Options
[Auto, Basic, Normal, Good, VeryGood, Excellent]

GUI name
Spline Zlm fit

Description
Quality of the density-fitting approximation. For a description of the various qualities and the associated numerical accuracy see reference. If 'Auto', the quality defined in the 'NumericalQuality' will be used.

`QualityPerRegion`

Type
Block

Recurring
True

Description
Sets the ZlmFit quality for all atoms in a region. If specified, this overwrites the globally set quality.

`Quality`

Type
Multiple Choice

Options
[Basic, Normal, Good, VeryGood, Excellent]

Description
The region's quality of the ZlmFit.

`Region`

Type
String

Description
The identifier of the region for which to set the quality.

12.2.2 conductance

`EnergyGrid`

Type
Block

Description
Energy grid for Transmission Function

`Max`

Type
Float

Default value
5.0

Unit
eV

Description
Max Energy (relative to Fermi energy)

Min

Type
Float

Default value
-5.0

Unit
eV

Description
Min energy (relative to Fermi energy)

Num

Type
Integer

Default value
200

Description
Number of energy values in which the interval Min-Max is subdivided

Files

Type
Block

Description
path of files

HamiltonianElectrode

Type
String

Default value
Description

HamiltonianMolecule

Type
String

Default value
Description

Leads

Type
String

Default value
Description
Path (either absolute or relative) of the lead results file

OverlapElectrode

Type
String

Default value
Description

`OverlapMolecule`

Type
String

Default value
Description

`Scattering`

Type
String

Default value
Description
Path (either absolute or relative) of the scattering region results

`Output`

Type
Block

Description
options describing what should be printed

`OldOutput`

Type
Bool

Default value
No

Description

`Physics`

Type
Block

Description
Block describing the physics of the system

`FermiEnergy`

Type
Block

Description
Block describing the physics of the system

`Electrode`

Type
Float

Default value
0.0

Description
Fermi energy of the electrode

`Technical`

Type
Block

Description
options describing technical parts of the calculation

`Eta`

Type
Float

Default value
1e-05

Description
To avoid poles of the Green's function, a small imaginary number is added to the energy

`overwriteLeads`

Type
Bool

Default value
Yes

Description
If true, Hamiltonians H_L and H_R are taken from the DFTB-leads calculation. If False, they are taken from the DFTB scattering-region calculation

`setOffDiagonalToZero`

Type
Bool

Default value
Yes

Description
If true, H_LR and S_LR are explicitly set to zero. If False, they are taken from the DFTB scattering-region calculation.

12.2.3 sgf

`Debug`

Type
String

Recurring
True

Description

`Save`

Type
String

Recurring
True

Description

SurfaceGF

Type
Block

Description

BANDMU

Type
Float

Default value
0.0

Description

CONTACT

Type
Integer

Default value
1

Description

CPMARGIN

Type
Float

Description

ContourQuality

Type
Multiple Choice

Default value
good

Options
[basic, normal, good, verygood, excellent]

Description

DELTA

Type
Float

Description

ETA

Type
Float

Default value
1e-05

Description

KGRID

Type
Integer

Description

KT

Type
Float

Description

MAXDOS

Type
Float

Description

MINDOS

Type
Float

Description

MINVALENCE

Type
Float

Description

MULOWER

Type
Float

Description

MUUPPER

Type
Float

Description

NARCPTS

Type
Integer

Description

NDOSPTS

Type
Integer

Description

NE

Type
Integer

Description

NEQLINEPTS

Type
Integer

Description

NFPOLES

Type
Integer

Description

NLAYERS

Type
Integer

Description

NNONEQLPTS

Type
Integer

Description

PHI1

Type
Float

Default value
0.0

Description

PHI2

Type
Float

Default value
0.0

Description

RKFFileName

Type
String

Default value
RUNKF

Description

SCMCODE

Type
Bool

Default value
No

Description

SECANT

Type
Bool

Default value
No

Description

SGFMaxIter

Type
Integer

Default value
500

Description

SGFTOL

Type
Float

Default value
1e-08

Description

TOL

Type
Float

Default value
0.0001

Description

TRANSDIR

Type
Integer

Default value
1

Description

TRUEMU

Type
Float

Default value
0.0

Description

deContourInt

Type
Float

Default value
-1.0

Description

`TightBinding`

Type
Block

Description

`DTol`

Type
Float

Default value
0.001

Description

`Eps`

Type
Float

Default value
1e-18

Description

`IODim`

Type
Integer

Description

`MaxMerit`

Type
Float

Description

`MaxRange`

Type
Float

Description

`XTol`

Type
Float

Default value
5e-06

Description

`Title`

Type
String

Description

CHAPTER

THIRTEEN

KF OUTPUT FILES

13.1 Accessing KF files

KF files are Direct Access binary files. KF stands for Keyed File: KF files are keyword oriented, which makes them easy to process by simple procedures. Internally all the data on KF files is organized into sections containing variables, so each datum on the file can be identified by the combination of section and variable.

All KF files can be opened using the KFbrowser GUI program:

```
$AMSBIN/kfbrowser path/to/ams.rkf
```

By default KFbrowser shows a just a curated summary of the results on the file, but you can make it show the raw section and variable structure by switching it to expert mode. To do this, click on **File → Expert Mode** or press **ctrl/cmd + e**.

KF files can be opened and read with Command line tools.

For working with the data from KF files, it is often useful to be able to read them from Python. Using the AMS Python Stack, this can easily be done with the AKFReader class:

```
>>> from scm.akfreader import AKFReader
>>> kf = AKFReader("path/to/ams.rkf")
>>> "Molecule%Coords" in kf
True
>>> kf.description("Molecule%Coords")
{
    '_type': 'float_array',
    '_shape': [3, 'nAtoms'],
    '_comment': 'Coordinates of the nuclei (x,y,z)',
    '_unit': 'Bohr'
}
>>> kf.read("Molecule%Coords")
array([[-11.7770694 ,  -4.19739597,   0.04934546],
       [ -9.37471321,  -2.63234227,  -0.13448698],
        ...
       [ 10.09508738,  -1.06191208,   1.45286913],
       [ 10.11689333,  -1.5080196 ,  -1.87916127]])
```

Tip: For a full overview of the available methods in AKFReader, see the AKFReader API documentation.

13.2 Sections and variables on band.rkf

AMSResults

Section content: Generic results of the BAND evaluation.

AMSResults%Bonds

Type
subsection

Description
Bond info

AMSResults%Bonds%Atoms

Type
archived_int_array

Description
?

AMSResults%Bonds%CellShifts

Type
archived_int_array

Description
?

AMSResults%Bonds%description

Type
string

Description
A string containing a description of how the bond orders were calculated / where they come from

AMSResults%Bonds%hasCellShifts

Type
bool

Description
Whether there are cell shifts (relevant only in case of periodic boundary conditions)

AMSResults%Bonds%Index

Type
archived_int_array

Description
index(i) points to the first element of Atoms, Orders, and CellShifts belonging to bonds from atom 'i'. Index(1) is always 1, Index(nAtoms+1) is always nBonds + 1

AMSResults%Bonds%Orders

Type
archived_float_array

Description
The bond orders.

AMSResults%BulkModulus

Type
float

Description
The Bulk modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

AMSResults%Charges

Type
float_array

Description
Net atomic charges as computed by the engine (for example, the Charges for a water molecule might be [-0.6, 0.3, 0.3]). The method used to compute these atomic charges depends on the engine.

Unit
e

Shape
[Molecule%nAtoms]

AMSResults%DipoleGradients

Type
float_array

Description
Derivative of the dipole moment with respect to nuclear displacements.

Shape
[3, 3, Molecule%nAtoms]

AMSResults%DipoleMoment

Type
float_array

Description
Dipole moment vector (x,y,z)

Unit
e*bohr

Shape
[3]

AMSResults%ElasticTensor

Type
float_array

Description
The elastic tensor in Voigt notation (6x6 matrix for 3D periodic systems, 3x3 matrix for 2D periodic systems, 1x1 matrix for 1D periodic systems).

Unit
hartree/bohr^nLatticeVectors

Shape
[:, :]

AMSResults%Energy

Type
float

Description
The energy computed by the engine.

Unit
hartree

AMSResults%fractionalOccupation

Type
bool

Description
Whether of not we have fractionally occupied orbitals (i.e. not all occupations are integer numbers).

AMSResults%Gradients

Type
float_array

Description
The nuclear gradients.

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

AMSResults%Hessian

Type
float_array

Description
The Hessian matrix

Unit
hartree/bohr^2

Shape
[3*Molecule%nAtoms, 3*Molecule%nAtoms]

AMSResults%HOMOEnergy

Type
float_array

Description
Molecular Orbital Info: energy of the HOMO.

Unit
hartree

Shape
[nSpin]

AMSResults%HOMOIndex

Type
int_array

Description
Molecular Orbital Info: index in the arrays orbitalEnergies and orbitalOccupations corresponding to the HOMO.

Shape
[nSpin]

AMSResults%HOMOLUMOGap

Type
float_array

Description
Molecular Orbital Info: HOMO-LUMO gap per spin.

Unit
hartree

Shape
[nSpin]

AMSResults%LUMOEnergy

Type
float_array

Description
Molecular Orbital Info: energy of the LUMO.

Unit
hartree

Shape
[nSpin]

AMSResults%LUMOIndex

Type
int_array

Description
Molecular Orbital Info: index in the arrays orbitalEnergies and orbitalOccupations corresponding to the LUMO.

Shape
[nSpin]

AMSResults%Molecules

Type
subsection

Description
Molecules

AMSResults%Molecules%AtCount

Type
archived_int_array

Description
shape=(nMolType), Summary: number of atoms per formula.

AMSResults%Molecules%Atoms

Type
archived_int_array

Description
shape=(nAtoms), atoms(index(i):index(i+1)-1) = atom indices of molecule i

AMSResults%Molecules%Count

Type
archived_int_array

Description
Mol count per formula.

AMSResults%Molecules%Formulas

Type
string

Description
Summary: unique molecule formulas

AMSResults%Molecules%Index

Type
archived_int_array

Description
shape=(nMol+1), index(i) = index of the first atom of molecule i in array atoms(:)

AMSResults%Molecules%Type

Type
archived_int_array

Description
shape=(nMol), type of the molecule, reference to the summary arrays below

AMSResults%nOrbitals

Type
int

Description
Molecular Orbital Info: number of orbitals.

AMSResults%nSpin

Type
int

Description
Molecular Orbital Info: number spins (1: spin-restricted or spin-orbit coupling, 2: spin unrestricted).

AMSResults%orbitalEnergies

Type
float_array

Description
Molecular Orbital Info: the orbital energies.

Unit
hartree

Shape
[nOrbitals, nSpin]

AMSResults%orbitalOccupations

Type
float_array

Description
Molecular Orbital Info: the orbital occupation numbers. For spin restricted calculations, the value will be between 0 and 2. For spin unrestricted or spin-orbit coupling the values will be between 0 and 1.

Shape
[nOrbitals, nSpin]

AMSResults%PESPointCharacter

Type
string

Description
The character of a PES point.

Possible values
['local minimum', 'transition state', 'stationary point with >1 negative frequencies', 'non-stationary point']

AMSResults%PoissonRatio

Type
float

Description
The Poisson ratio

AMSResults%ShearModulus

Type
float

Description
The Shear modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

AMSResults%SmallestHOMOLUMOGap

Type
float

Description
Molecular Orbital Info: the smallest HOMO-LUMO gap irrespective of spin (i.e. min(LUMO) - max(HOMO)).

Unit
hartree

AMSResults%StressTensor

Type
float_array

Description
The clamped-ion stress tensor in Cartesian notation.

Unit
hartree/bohr^nLatticeVectors

Shape
[:, :]

AMSResults%YoungModulus

Type
float

Description
The Young modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit
hartree/bohr^3

AngularBoost
Section content: Both the Becke grid and the Zlm fit grid may boost the angular grid for certain areas.

AngularBoost%boost

Type
bool_array

Description
Whether to use a booster grid per atom.

band_curves
Section content: Band dispersion curves.

band_curves%brav_type

Type
string

Description
Type of the lattice.

band_curves%Edge_#_bands

Type
float_array

Description
The band energies

Shape
[nBands, nSpin, :]

band_curves%Edge_#_direction

Type
float_array

Description
Direction vector.

Shape
[nDimK]

band_curves%Edge_#_fatBands

Type
float_array

Description
Fat band split up of the bands

Shape
[nBas, nBands, nSpin, :]

band_curves%Edge_#_kPoints

Type
float_array

Description
Coordinates for points along the edge.

Shape
[nDimK, :]

band_curves%Edge_#_labels

Type
lchar_string_array

Description
Labels for begin and end point of the edge.

Shape
[2]

band_curves%Edge_#_lGamma

Type
bool

Description
Is gamma point?

band_curves%Edge_#_nKPoints

Type
int

Description
The nr. of k points along the edge.

band_curves%Edge_#_vertices

Type
float_array

Description
Begin and end point of the edge.

Shape
[nDimK, 2]

band_curves%Edge_#_xFor1DPlotting

Type
float_array

Description
x Coordinate for points along the edge.

Shape
[:]

band_curves%indexLowestBand

Type
int

Description
?

band_curves%nBands

Type
int

Description
Number of bands.

band_curves%nBas

Type
int

Description
Number of basis functions.

band_curves%nDimK

Type
int

Description
Dimension of the reciprocal space.

band_curves%nEdges

Type
int

Description
The number of edges. An edge is a line-segment through k-space. It has a begin and end point and possibly points in between.

band_curves%nEdgesInPath

Type
int

Description
A path is built up from a number of edges.

band_curves%nSpin

Type
int

Description
Number of spin components.

Possible values
[1, 2]

band_curves%path

Type
int_array

Description
If the (edge) index is negative it means that the vertices of the edge abs(index) are swapped e.g. path = (1,2,3,0,-3,-2,-1) goes though edges 1,2,3, then there's a jump, and then it goes back.

Shape
[nEdgesInPath]

band_curves%path_source

Type
string

Description
Source or program used to generate the path.

Possible values
['input', 'kpath', 'seekpath']

band_curves%path_type

Type
string

Description
?

BandStructure

Section content: Info regarding the band structure.

BandStructure%BandGap

Type
float

Description
The band gap. For molecules this is the HOMO-LUMO gap.

Unit
hartree

BandStructure%bandsEnergyRange

Type
float_array

Description
The energy ranges (min/max) of the bands

Unit
hartree

Shape
[2, nBand, nSpin]

BandStructure%BottomConductionBand

Type
float

Description
The bottom of the conduction band

Unit
hartree

BandStructure%CoordsBottomConductionBand

Type
float_array

Description
The coordinates in k-space of the bottom of the conduction band

Unit
1/bohr

Shape
[nDimK]

BandStructure%CoordsTopValenceBand

Type
float_array

Description
The coordinates in k-space of the top of the valence band

Unit
1/bohr

Shape
[nDimK]

BandStructure%DerivativeDiscontinuity

Type
float

Description
Correction to be added to the band gap to get the fundamental gap. (band only)

Unit
hartree

BandStructure%FermiEnergy

Type
float

Description
Fermi level

Unit
hartree

BandStructure%HasGap

Type
bool

Description
Whether the system has a gap.

BandStructure%HomoBandIndex

Type
int

Description
The index of the highest occupied band

BandStructure%HomoDegeneracy

Type
int

Description
How many states are exactly at the HOMO level

BandStructure%HomoSpinIndex

Type
int

Description
In case of an unrestricted calculation: which of the two spins has the HOMO?

BandStructure%LumoBandIndex

Type
int

Description
The index of the lowest unoccupied band

BandStructure%LumoDegeneracy

Type
int

Description
How many states are exactly at the LUMO level

BandStructure%LumoSpinIndex

Type
int

Description
In case of an unrestricted calculation: which of the two spins has the LUMO?

BandStructure%nBand

Type
int

Description
The number of bands for which the band ranges are stored.

BandStructure%nDimK

Type
int

Description
The number of dimensions for the k-coordinates for CoordsTopValenceBand and CoordsBottomConductionBand.

`BandStructure%nSpin`

Type
int

Description
If 1: spin restricted calculation. For unrestricted results it has the value of 2.

Possible values
[1, 2]

`BandStructure%TopValenceBand`

Type
float

Description
The top of the valence band

Unit
hartree

BandStructure(FromPath)
Section content: Info regarding the band structure.

`BandStructure(FromPath)%BandGap`

Type
float

Description
The band gap. For molecules this is the HOMO-LUMO gap.

Unit
hartree

`BandStructure(FromPath)%bandsEnergyRange`

Type
float_array

Description
The energy ranges (min/max) of the bands

Unit
hartree

Shape
[2, nBand, nSpin]

`BandStructure(FromPath)%BottomConductionBand`

Type
float

Description
The bottom of the conduction band

Unit
hartree

BandStructure(FromPath)%CoordsBottomConductionBand

Type
float_array

Description
The coordinates in k-space of the bottom of the conduction band

Unit
1/bohr

Shape
[nDimK]

BandStructure(FromPath)%CoordsTopValenceBand

Type
float_array

Description
The coordinates in k-space of the top of the valence band

Unit
1/bohr

Shape
[nDimK]

BandStructure(FromPath)%DerivativeDiscontinuity

Type
float

Description
Correction to be added to the band gap to get the fundamental gap. (band only)

Unit
hartree

BandStructure(FromPath)%FermiEnergy

Type
float

Description
Fermi level

Unit
hartree

BandStructure(FromPath)%HasGap

Type
bool

Description
Whether the system has a gap.

BandStructure(FromPath)%HomoBandIndex

Type
int

Description
The index of the highest occupied band

BandStructure(FromPath)%HomoDegeneracy

Type
int

Description
How many states are exactly at the HOMO level

BandStructure(FromPath)%HomoSpinIndex

Type
int

Description
In case of an unrestricted calculation: which of the two spins has the HOMO?

BandStructure(FromPath)%LumoBandIndex

Type
int

Description
The index of the lowest unoccupied band

BandStructure(FromPath)%LumoDegeneracy

Type
int

Description
How many states are exactly at the LUMO level

BandStructure(FromPath)%LumoSpinIndex

Type
int

Description
In case of an unrestricted calculation: which of the two spins has the LUMO?

BandStructure(FromPath)%nBand

Type
int

Description
The number of bands for which the band ranges are stored.

BandStructure(FromPath)%nDimK

Type
int

Description
The number of dimensions for the k-coordinates for CoordsTopValenceBand and CoordsBottomConductionBand.

BandStructure(FromPath)%nSpin

Type
int

Description
If 1: spin restricted calculation. For unrestricted results it has the value of 2.

Possible values
[1, 2]

`BandStructure(FromPath)%TopValenceBand`

Type
float

Description
The top of the valence band

Unit
hartree

basis

Section content: Information on the basis set.

`basis%core functions/part`

Type
int_array

Description
Number of core functions per part.

Shape
[number of parts]

`basis%core functions/type`

Type
int_array

Description
Number of core functions per type.

Shape
[geometry%ntyp]

`basis%core_labels`

Type
lchar_string_array

Description
Labels for the core functions.

Shape
[ncores]

`basis%icpat`

Type
int_array

Description
See ifpat, but now for core functions.

Shape
[ncores, Molecule%nAtoms]

`basis%icpati`

Type
int_array

Description
See ifpati, but now for core functions.

Shape
[2, ncores]

basis%idosat

Type
int_array

Description
Atom i in dos order is atom idosat(i) as on input.

Shape
[Molecule%nAtoms]

basis%idosati

Type
int_array

Description
Atom i in input order is atom idosati(i) in dos order.

Shape
[Molecule%nAtoms]

basis%ifpat

Type
int_array

Description
If you specify the atom number i, as on input, and the basis function on that atom j, counting first all NAO's of that atom and then all STOs, the number of the basis function is { t ifpat(j,i)}.

Shape
[nbas, Molecule%nAtoms]

basis%ifpati

Type
int_array

Description
If you know the basis function k, it was function { t ifpati(2,k)} on atom { t ifpati(1,k)}.

Shape
[2, nbas]

basis%ilmdos

Type
int_array

Description
Used for DOS analysis. 1: atom (internal order), 2:l, 3: m.

Shape
[3, nbas+ncores]

basis%inputAtomPerBas

Type
int_array

Description
The input atom index for each (valence) basis function.

Shape
[nbas]

basis%is NAO all functions

Type
bool_array

Description
Whether a function is a NAO (a solution for a spherical atom), rather than an STO.

Shape
[ncores+nbas]

basis%lPerBas

Type
int_array

Description
l value for each (valence) basis function.

Shape
[nbas]

basis%Maximum l value fit

Type
int

Description
Maximum l value of the STO fit functions, if any.

basis%mPerBas

Type
int_array

Description
m value for each (valence) basis function.

Shape
[nbas]

basis%nbas

Type
int

Description
Number of (valence) basis functions used during the SCF.

basis%ncores

Type
int

Description
Number of frozen core functions.

basis%number of parts

Type
int

Description
Number of parts (fragments), normally atoms.

basis%Quantum numbers for all function

Type
int_array

Description
atom number,l,m for all functions, first core then valence. Atom index is in internal order.

Shape
[3, ncores+nbas]

basis%valence functions/part

Type
int_array

Description
Number of valence functions per part.

Shape
[number of parts]

basis%valence functions/type

Type
int_array

Description
Number of valence functions per type.

Shape
[geometry%ntyp]

basis%valence_labels

Type
lchar_string_array

Description
Labels for the valence functions.

Shape
[ncores]

BeckeGridConfig

Section content: Configuration used to create the Becke grid.

BeckeGridConfig%angLOrder

Type
int_array

Description
?.

Shape
[:]

BeckeGridConfig%beckeMapParams

Type
float_array

Description
Mapping parameter per atom.

Shape
[nAtoms]

BeckeGridConfig%includeRadialWeights

Type
bool

Description
Whether or not to include the radial weights. Normally you want this.

BeckeGridConfig%isSymmetryUnique

Type
bool_array

Description
Is an atom symmetry unique?

Shape
[nAtoms]

BeckeGridConfig%minimumRadius

Type
float

Description
To solve the exact singularity a small hard sphere around the nuclei can be used. The partition function starts beyond this radius.

BeckeGridConfig%mpvPartitionCheckSpheres

Type
bool

Description
Whether or not to check the spheres for the MPV partitioning.

BeckeGridConfig%nAtoms

Type
int

Description
Number of atoms.

BeckeGridConfig%nRadPoints

Type
int_array

Description
Number of radial points per atom.

Shape
[nAtoms]

BeckeGridConfig%oper

Type
float_array

Description
Point group part of the symmetry operators.

Shape
[3, 3, :]

BeckeGridConfig%partitionFunThresh

Type
float

Description
Threshold for the partition function.

BeckeGridConfig%qAtoms

Type
float_array

Description
Atomic number per atom.

Shape
[nAtoms]

BeckeGridConfig%quality

Type
string_fixed_length

Description
Quality used.

BeckeGridConfig%transl

Type
float_array

Description
Translational part of the symmetry operators.

Shape
[3, :]

BeckeGridConfig%vectors

Type
float_array

Description
Lattice vectors

Unit
bohr

BeckeGridConfig%xyzAtoms

Type
float_array

Description
Atom coordinates.

Unit
bohr

Shape
[3, nAtoms]

BeckeGridConfig(fit)
Section content: The Zlm fit employs also a becke grid, but one that is typically less dense.

`BeckeGridConfig(fit)%angLOrder`

Type
int_array

Description
?.

Shape
[:]

`BeckeGridConfig(fit)%beckeMapParams`

Type
float_array

Description
Mapping parameter per atom.

Shape
[nAtoms]

`BeckeGridConfig(fit)%includeRadialWeights`

Type
bool

Description
Whether or not to include the radial weights. Normally you want this.

`BeckeGridConfig(fit)%isSymmetryUnique`

Type
bool_array

Description
Is an atom symmetry unique?

Shape
[nAtoms]

`BeckeGridConfig(fit)%minimumRadius`

Type
float

Description
To solve the exact singularity a small hard sphere around the nuclei can be used. The partition function starts beyond this radius.

`BeckeGridConfig(fit)%mpvPartitionCheckSpheres`

Type
bool

Description
Whether or not to check the spheres for the MPV partitioning.

BeckeGridConfig(fit)%nAtoms

Type
int

Description
Number of atoms.

BeckeGridConfig(fit)%nRadPoints

Type
int_array

Description
Number of radial points per atom.

Shape
[nAtoms]

BeckeGridConfig(fit)%oper

Type
float_array

Description
Point group part of the symmetry operators.

Shape
[3, 3, :]

BeckeGridConfig(fit)%partitionFunThresh

Type
float

Description
Threshold for the partition function.

BeckeGridConfig(fit)%qAtoms

Type
float_array

Description
Atomic number per atom.

Shape
[nAtoms]

BeckeGridConfig(fit)%quality

Type
string_fixed_length

Description
Quality used.

BeckeGridConfig(fit)%transl

Type
float_array

Description
Translational part of the symmetry operators.

Shape
[3, :]

`BeckeGridConfig(fit)%vectors`

Type
float_array

Description
Lattice vectors

Unit
bohr

`BeckeGridConfig(fit)%xyzAtoms`

Type
float_array

Description
Atom coordinates.

Unit
bohr

Shape
[3, nAtoms]

Berry phase

Section content: The Berry phase method is a way to define a dipole in a periodic system.

`Berry phase%Dipole moment (a.u.)`

Type
float_array

Description
Dipole moment.

Unit
a.u.

Shape
[kspace%ndim]

Bond energies

Section content: Bond energies for various hard-coded functionals. The energies are not self consistent, and obtained from the same SCF density.

`Bond energies%*`

Type
float

Description
Bond energy according to the functional used during the SCF. The name is the name of the functional. Used if fnal bond energy is according the the SCF functional.

Unit
hartree

Bond energies%all

Type
float_array

Description
All 14 hardcoded bond energies in an array.

Unit
hartree

Shape
[14]

Bond energies%Becke (alt)

Type
float

Description
Bond energy according to the Becke (alt) functional.

Unit
hartree

Bond energies%Becke88

Type
float

Description
Bond energy according to the Becke88 (exchange) functional.

Unit
hartree

Bond energies%Becke88+LYP

Type
float

Description
Bond energy according to the Becke88+LYP functional.

Unit
hartree

Bond energies%Becke88+Perdew86

Type
float

Description
Bond energy according to the Becke88 plus Perdew86 (XC) functional.

Unit
hartree

Bond energies%EV93x+PW91c

Type
float

Description
Bond energy according to the EV93x+PW91c functional.

Unit
hartree

Bond energies%final bond energy

Type
float

Description
Bond energy according to the functional used during the SCF.

Unit
hartree

Bond energies%LDA

Type
float

Description
Bond energy according to the LDA functional.

Unit
hartree

Bond energies%PBE

Type
float

Description
Bond energy according to the PBE functional.

Unit
hartree

Bond energies%PBESOL

Type
float

Description
Bond energy according to the PBESOL functional.

Unit
hartree

Bond energies%Perdew-Wang (91) X

Type
float

Description
Bond energy according to the Perdew-Wang (91) (exchange) functional.

Unit
hartree

Bond energies%Perdew-Wang (91) X+C

Type
float

Description
Bond energy according to the Perdew-Wang (91) X+C functional.

Unit
hartree

`Bond energies%PW86x+Perdew86c`

Type
float

Description
Bond energy according to the PW86x+Perdew86c functional.

Unit
hartree

`Bond energies%RGE2`

Type
float

Description
Bond energy according to the RGE2 functional.

Unit
hartree

`Bond energies%RPBE`

Type
float

Description
Bond energy according to the RPBE functional.

Unit
hartree

`Bond energies%SELF-CONSISTENT`

Type
float

Description
Bond energy according to the functional used during the SCF, in case that it is different from the final bond energy.

Unit
hartree

Bond energies (meta) GGAs
Section content: XC energy terms according to some hardcoded list of functionals.

`Bond energies (meta) GGAs%BLYP`

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%BOP

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%BP

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%FT97

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%HCTH/120

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%HCTH/147

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%HCTH/407

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%HCTH/93

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%KCIS-modified

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%KCIS-original

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%KT1

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%KT2

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%LAK

Type
float

Description
?

Bond energies (meta) GGAs%LAKc

Type
float

Description
?

Bond energies (meta) GGAs%LAKx

Type
float

Description
?

Bond energies (meta) GGAs%LDA(VWN)

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%M06-L

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%M11-L

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%mPBE

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%mPBEKCIS

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%mPW

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%MS0

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%MS1

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%MS2

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%MVS

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%MVSx

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%OLYP

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%OPBE

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%OPerdew

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%PBE

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%PBEsol

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%PKZB

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%PKZBx-KCIScor

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%PW91

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%revPBE

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%revTPSS

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%RGE2

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%RPBE

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%SCAN

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%SCANc

Type
float

Description
?

Bond energies (meta) GGAs%SCANx

Type
float

Description
?

Bond energies (meta) GGAs%SOGGA

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%SOGGA11

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%SSB-D

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%TASK

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%TASKCC

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%TASKCCALDA

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%TASKLDA

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%TASKSCAN

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%TASKxc

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%tau-HCTH

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%TPSS

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%VS98

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%VS98-x(xc)

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%VS98-x-only

Type
float

Description
.

Unit
hartree

Bond energies (meta) GGAs%XLYP

Type
float

Description
.

Unit
hartree

Bond energy terms

Section content: Bond energy terms.

`Bond energy terms%Dispersion`

Type
float

Description
Empirical dispersion contribution to the bond energy.

Unit
hartree

`Bond energy terms%Electric field`

Type
float

Description
External electric field contribution to the bond energy.

Unit
hartree

`Bond energy terms%Electrostatic`

Type
float

Description
Electrostatic contribution to the bond energy. This is about bringing together the spherical atoms at their positions in the system. The deformation density is not in this term.

Unit
hartree

`Bond energy terms%Hubbard Energy`

Type
float

Description
Hubbard model contribution to the bond energy.

Unit
hartree

`Bond energy terms%Kinetic energy`

Type
float

Description
Kinetic energy contribution to the bond energy.

Unit
hartree

`Bond energy terms%Madelung`

Type
float

Description
Madelung contribution to the bond energy. Is only nonzero when charged atoms are used (never).

Unit
hartree

Bond energy terms%Post SCF correlation

Type
float

Description
?

Unit
hartree

Bond energy terms%RPAxc

Type
float

Description
?

Unit
hartree

Bond energy terms%Solvation

Type
float

Description
Solvation model contribution to the bond energy.

Unit
hartree

Bond energy terms%V(atomic)*rho(def)

Type
float

Description
Contribution to the electrostatic interaction due to the deformation density.

Unit
hartree

Bond energy terms%V(def)*err

Type
float

Description
Fit error correction to the electrostatic interaction.

Unit
hartree

Bond energy terms%V(def)*rho(def)

Type
float

Description
Contribution to the electrostatic interaction due to the deformation density..

Unit
hartree

Bond energy terms%XC

Type
float

Description
XC contribution to the bond energy.

Unit
hartree

BZcell(primitive cell)

Section content: The Brillouin zone of the primitive cell.

BZcell(primitive cell)%boundaries

Type
float_array

Description
Normal vectors for the boundaries.

Shape
[ndim, nboundaries]

BZcell(primitive cell)%distances

Type
float_array

Description
Distance to the boundaries.

Shape
[nboundaries]

BZcell(primitive cell)%idVerticesPerBound

Type
int_array

Description
The indices of the vertices per bound.

Shape
[nvertices, nboundaries]

BZcell(primitive cell)%latticeVectors

Type
float_array

Description
The lattice vectors.

Shape
[3, :]

BZcell(primitive cell)%nboundaries

Type
int

Description
The nr. of boundaries for the cell.

BZcell(primitive cell)%ndim

Type
int

Description
The nr. of lattice vectors spanning the Wigner-Seitz cell.

BZcell(primitive cell)%numVerticesPerBound

Type
int_array

Description
The nr. of vertices per bound.

Shape
[nboundaries]

BZcell(primitive cell)%nvertices

Type
int

Description
The nr. of vertices of the cell.

BZcell(primitive cell)%vertices

Type
float_array

Description
The vertices of the bounds.

Unit
a.u.

Shape
[ndim, nvertices]

COSMO

Section content: COSMO solvation model.

COSMO%amat

Type
float_array

Description
The matrix that defines the cosmo solution.

Shape
[npUnique, npUnique]

COSMO%amatDiag

Type
float_array

Description
Diagonal part of the matrix that defines the cosmo solution.

Shape
[npUnique]

COSMO%cellSurfDistances

Type
float_array

Description
Distance to the cosmo surface: nearest COSMO point to a cell coordinate.

Shape
[ncell]

COSMO%fscal

Type
float

Description
Solvent dependent scaling factor.

COSMO%isAtomSym

Type
int_array

Description
Each point may belong to a maximum of noper atoms (i.e. is shared). This affects the gradients. Most points belong to one atom.

Shape
[Nr. of operators, npSym]

COSMO%ncell

Type
int

Description
Number of cells.

COSMO%nequiv

Type
int_array

Description
Each unique point may stand for several ones.

Shape
[npUnique]

COSMO%npSym

Type
int

Description
Number of symmetric points.

COSMO%npUnique

Type
int

Description
Number of unique symmetric points.

COSMO%Nr. of operators

Type
int

Description
Number of symmetry operators.

COSMO%Number of Segments

Type
int

Description
Number of segments that form the COSMO surface.

COSMO%pointsMap

Type
int_array

Description
Each point maps to another point under symmetry operations.

Shape
[Nr. of operators, npSym]

COSMO%qSym

Type
float_array

Description
COSMO charges in the symmetric points.

Shape
[npSym]

COSMO%qUnique

Type
float_array

Description
COSMO charges in the unique points.

Shape
[npUnique]

COSMO%Segment Charge Density

Type
float_array

Description
COSMO charges divided by the segment surfaces.

Unit
bohr

Shape
[Number of Segments]

`COSMO%Segment Coordinates`

Type
float_array

Description
Coordinates of the segments.

Unit
bohr

Shape
[3, Number of Segments]

`COSMO%uniqueToFullIndex`

Type
int_array

Description
ipSym=uniqueToFullIndex(ipUnique).

Shape
[npUnique]

`COSMO%xyzaSym`

Type
float_array

Description
Symmetric COSMO points coordinates and area, obtained by applying the operators on the original points, and removing duplicates.

Shape
[4, npSym]

`COSMO%xyzaUnique`

Type
float_array

Description
Unique symmetric COSMO points coordinates and area.

Shape
[4, npUnique]

DataForVoronoiGrid
Section content: ?

`DataForVoronoiGrid%alfas`

Type
float_array

Description
?

DataForVoronoiGrid%npowx

Type
int

Description
?

Dependency
Section content: ?

Dependency%minNBasOrth

Type
int

Description
?

Dependency%nBasOrthPerKun

Type
int_array

Description
?

DFTHalf
Section content: Information for the DFTB 1/2 method to improve the band gap. It is about adding something to the SCF potential.

DFTHalf%V_type_*

Type
float_array

Description
Spherical potential for an atom of a certain type (atomic number).

Unit
a.u.

Shape
[:]

DFTHalfPreparation
Section content: Analysis on which atom (types) contribute to the TOVB. These can be activated in a DFT 1/2 calculation.

DFTHalfPreparation%ActiveAtomNames

Type
lchar_string_array

Description
Names of the active atom types.

DFTHalfPreparation%ActiveAtomTypes

Type
int_array

Description
Which atom (types, band style) should be active in the next DFT 1/2 run?

DFTHalfPreparation%ActiveAtomTypesLValue

Type
int_array

Description
For the active atoms 1/2 an electron should be removed from a certain atomic orbital. Not possible to use this suggestion right now (it will be simply the HOMO of that atom).

DFTHalfPreparation%Error message

Type
string

Description
An error message in case something goes wrong.

DFTHalfPreparation%Success

Type
bool

Description
Whether the preparation step was successful.

dos

Section content: Mainly configuration options for the band DOS code.

dos%dfermi

Type
float

Description
Uncertainty in the Fermi energy. Depending on the occupation strategy this can be a large number, for instance if it may be arbitrarily anywhere in a gap.

Unit
hartree

dos%efdirc

Type
float_array

Description
Energies used for fermi energy averaging. (only leading 1:nfdirc part is used)

Unit
hartree

Shape
[nfdirc]

dos%efermi

Type
float

Description
Fermi energy.

Unit
hartree

dos%entropy correction

Type
float

Description
Entropy energy from the finite electronic temperature.

Unit
hartree

dos%ifragdos

Type
int_array

Description
For each dos part the fragment number. Any atom not present in these fragments is handled …. as an atom.

Shape
[nfragdos]

dos%nband_dosplot

Type
int

Description
Number of bands used for the DOS.

dos%nfdirc

Type
int

Description
DOS is sampled at several energies at once, and (weight) averaged over them.

dos%nfragdos

Type
int

Description
Number of parts to be used for the DOS analysis. Normally (nfragdos=0) the atoms are the building blocks but larger fragments can also be used, like the MOs of a molecule. The fragments may cover only a part of the whole system, the rest will be atom based.

dos%SpinDependentFermiEnergies

Type
float_array

Description
Fermi energy per spin component, only when nspin==2.

Unit
hartree

Shape
[2]

dos%T(V+C/D+C)

Type
float_array

Description
Trafo from the full V+C basis to the DOS basis.

Unit
hartree

Shape
[SystType%nbas+SystType%ncores, SystType%nbas+SystType%ncores, 2, kspace%kuniqu]

dos%T(V+C/D+C)-1**

Type
float_array

Description
Inverse trafo from the full V+C basis to the DOS basis. Note: it appears that this variable is created but never filled out with actual data.

Unit
hartree

dos%wfdirc

Type
float_array

Description
Weights used for fermi energy averaging.

Shape
[nfdirc]

DOS

Section content: Info regarding the DOS

DOS%Atom per basis function

Type
int_array

Description
Atom index per basis function.

DOS%COOP per basis pair

Type
float_array

Description
COOP per basis pair.

Shape
[nEnergies, nSpin, :, :]

DOS%DeltaE

Type
float

Description
The energy difference between sampled DOS energies. When there is no DOS at all a certain energy range can be skipped.

Unit
hartree

DOS%DOS per basis function

Type
float_array

Description
DOS contributions per basis function, based on Mulliken analysis.

Shape
[nEnergies, nSpin, :]

DOS%Energies

Type
float_array

Description
The energies at which the DOS is sampled.

Unit
hartree

Shape
[nEnergies]

DOS%Fermi Energy

Type
float

Description
The fermi energy.

Unit
hartree

DOS%IntegrateDeltaE

Type
bool

Description
If enabled it means that the DOS is integrated over intervals of DeltaE. Sharp delta function like peaks cannot be missed this way.

DOS%L-value per basis function

Type
int_array

Description
quantum number l for all basis functions.

DOS%M-value per basis function

Type
int_array

Description
quantum number m for all basis functions.

DOS%nEnergies

Type
int

Description
The nr. of energies to use to sample the DOS.

DOS%nSpin

Type
int

Description
The number of spin components for the DOS.

Possible values
[1, 2]

DOS%Overlap population per basis pai

Type
float_array

Description
? note that the word 'pair' is cut of due to the finite length of the kf variables name…

DOS%Population per basis function

Type
float_array

Description
?

DOS%Symmetry per basis function

Type
int_array

Description
?

DOS%Total DOS

Type
float_array

Description
The total DOS.

Shape
[nEnergies, nSpin]

DOS_Phonons
Section content: Phonon Density of States

DOS_Phonons%DeltaE

Type
float

Description
The energy difference between sampled DOS energies. When there is no DOS at all a certain energy range can be skipped.

Unit
hartree

DOS_Phonons%Energies

Type
float_array

Description
The energies at which the DOS is sampled.

Unit
hartree

Shape
[nEnergies]

DOS_Phonons%Fermi Energy

Type
float

Description
The fermi energy.

Unit
hartree

DOS_Phonons%IntegrateDeltaE

Type
bool

Description
If enabled it means that the DOS is integrated over intervals of DeltaE. Sharp delta function like peaks cannot be missed this way.

DOS_Phonons%nEnergies

Type
int

Description
The nr. of energies to use to sample the DOS.

DOS_Phonons%nSpin

Type
int

Description
The number of spin components for the DOS.

Possible values
[1, 2]

DOS_Phonons%Total DOS

Type
float_array

Description
The total DOS.

Shape
[nEnergies, nSpin]

EffectiveMass

Section content: In the effective mass approximation the curvature of the bands is a measure of the charge mobility. The curvature is obtained by numerical differentiation. The mass is the inverse of the curvature.

`EffectiveMass%EffectiveMasses`

Type
float_array

Description
Inverse curvatures at the extrema. Several bands may be sampled at once.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, MaxNBands, nKPoints, nSpin]

`EffectiveMass%ErrorEffectiveMasses`

Type
float_array

Description
Estimated errors from using two different step sizes for finite difference calculations.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, MaxNBands, nKPoints, nSpin]

`EffectiveMass%iBandHigh`

Type
int_array

Description
See comment for iBandLow.

Shape
[nKPoints]

`EffectiveMass%iBandLow`

Type
int_array

Description
For point k bands iBandLo(k) to iBandHi(k) are considered

Shape
[nKPoints]

`EffectiveMass%kCoordinates`

Type
float_array

Description
The coordinates in k-space of the top of the valence band(s) or bottom of conduction band(s).

Unit
1/bohr

Shape
[Molecule%nLatticeVectors, nKPoints]

`EffectiveMass%MaxNBands`

Type
int

Description
Maximum number of curvatures calculated for all k points.

`EffectiveMass%nKPoints`

Type
int

Description
The number of k points for which the effective mass is calculated. These should always be extrema (minimum or maximum) of the bands.

`EffectiveMass%nSpin`

Type
int

Description
If 1: spin restricted calculation. For unrestricted results it has the value of 2.

Possible values
[1, 2]

eigensystem
Section content: Information about the eigensystem.

`eigensystem%decoulpsi#`

Type
float

Description
Coulomb energy contribution to the bond energy according to psi_0. (or psi_1, etc. for Relief terms).

Unit
hartree

`eigensystem%dekinpsi#`

Type
float

Description
Kinetic energy contribution to the bond energy according to psi_0 (or psi_1, etc. for Relief terms).

Unit
hartree

`eigensystem%ebindpsi#`

Type
float_array

Description
Bond energy according to psi_0 (for 14 hardcoded functionals)). In case of Relief analysis there are not only psi_0, but also psi_1, etc. corresponding to Fock matrices that are partially set to zero.

Unit
hartree

Shape
[14]

eigensystem%eigval

Type
float_array

Description
Eigenvalues for all unique k-points.

Shape
[nband, nspin, kspace%kuniqu]

eigensystem%eMinMax

Type
float_array

Description
Minimum and maximum for all bands.

Shape
[2, nband, nspin]

eigensystem%hubbardOccupations

Type
float_array

Description
Occupations for the hubbard model.

Shape
[SystType%nbas, SystType%nspin]

eigensystem%isHubbardOrb

Type
bool_array

Description
Whether an orbital is active in the Hubbard model.

Shape
[SystType%nbas]

eigensystem%nband

Type
int

Description
Number of stored bands. This is smaller than or equal to the number of valence basis functions.

eigensystem%nband_occ

Type
int

Description
Number of bands with non zero occupations.

eigensystem%nspin

Type
int

Description
Number of spin components

Possible values
[1, 2]

eigensystem%occful

Type
float_array

Description
Occupation numbers for full bands in all k-points. Second component only used when nspin=2

Shape
[kspace%kt, 2]

eigensystem%occup

Type
float_array

Description
Occupation numbers for the bands in all unique k-points.

Shape
[nband_occ, nspin, kspace%kuniqu]

eigensystem%occupationPerBandAndSpin

Type
float_array

Description
Occupations per band and spin.

Shape
[nband, nspin]

eigensystem%PEDAocc

Type
float_array

Description
Occupations to be used.

eigensystem%T(VOC/FOC3)

Type
float_array

Description
Transformation (real/imag) from the VOC to the final fragment basis (FOC3). Does not allow for nspin=2.

Shape
[SystType%nbas, SystType%nbas, 2, kspace%kuniqu]

ElectrostaticEmbeddingType
Section content: Electrostatic embedding.

`ElectrostaticEmbeddingType%eeAttachTo`

Type
int_array

Description
A multipole may be attached to an atom. This influences the energy gradient.

`ElectrostaticEmbeddingType%eeChargeWidth`

Type
float

Description
If charge broadening was used for external charges, this represents the width of the charge distribution.

`ElectrostaticEmbeddingType%eeEField`

Type
float_array

Description
The external homogeneous electric field.

Unit
hartree/(e*bohr)

Shape
[3]

`ElectrostaticEmbeddingType%eeLatticeVectors`

Type
float_array

Description
The lattice vectors used for the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNLatticeVectors]

`ElectrostaticEmbeddingType%eeMulti`

Type
float_array

Description
The values of the external point- or multipole- charges.

Unit
a.u.

Shape
[eeNZlm, eeNMulti]

`ElectrostaticEmbeddingType%eeNLatticeVectors`

Type
int

Description
The number of lattice vectors for the external point- or multipole- charges.

`ElectrostaticEmbeddingType%eeNMulti`

Type
int

Description
The number of external point- or multipole- charges.

`ElectrostaticEmbeddingType%eeNZlm`

Type
int

Description
When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

`ElectrostaticEmbeddingType%eeUseChargeBroadening`

Type
bool

Description
Whether or not the external charges are point-like or broadened.

`ElectrostaticEmbeddingType%eeXYZ`

Type
float_array

Description
The position of the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNMulti]

Energy gradients

Section content: Various terms contributing to the energy gradients.

`Energy gradients%Cosmo`

Type
float_array

Description
COSMO solvation energy contribution to the gradients (at fixed density matrix).

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

Energy gradients%Dispersion

Type
float_array

Description
Empirical dispersion energy contribution to the gradients (at fixed density matrix).

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

Energy gradients%Electric field

Type
float_array

Description
External electric field contribution to the gradients (at fixed density matrix).

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

Energy gradients%Electrostatic energy

Type
float_array

Description
Non-pair electrostatic energy contribution to the gradients (at fixed density matrix).

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

Energy gradients%Kinetic energy

Type
float_array

Description
Kinetic energy contribution to the gradients (at fixed density matrix).

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

Energy gradients%P Matrix

Type
float_array

Description
Density matrix contribution to the gradients (pulay term).

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

`Energy gradients%Pair interactions`

Type
float_array

Description
Electrostatic pair energy contribution to the gradients. Follows from purely spherical pair contributions.

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

`Energy gradients%Total`

Type
float_array

Description
Total energy gradients.

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

`Energy gradients%XC energy`

Type
float_array

Description
XC energy contribution to the gradients (at fixed density matrix).

Unit
hartree/bohr

Shape
[3, Molecule%nAtoms]

Energy stress tensor

Section content: The stress tensor is the energy derivative with respect to strains, divided by the volume. They can and are obtained from energy derivatives along symmetric strain modes.

`Energy stress tensor%a1LatticeStrains`

Type
float_array

Description
Totally symmetric strains.

Unit
bohr

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors, nA1LatticeStrains]

Energy stress tensor%at def

Type
float_array

Description
At-def energy contribution to the stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

Energy stress tensor%at def(modeSD)

Type
float_array

Description
At-def contribution to the symmetric strain mode derivatives.

Unit
a.u.

Shape
[nA1LatticeStrains]

Energy stress tensor%def def

Type
float_array

Description
Def def (electrostatic) energy contribution to the stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

Energy stress tensor%def def(modeSD)

Type
float_array

Description
Def-def contribution to the symmetric strain mode derivatives.

Unit
a.u.

Shape
[nA1LatticeStrains]

Energy stress tensor%Dispersion

Type
float_array

Description
Empirical dispersion energy contribution to the stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

`Energy stress tensor%Dispersion(modeSD)`

Type
float_array

Description
Empirical dispersion contribution to the symmetric strain mode derivatives.

Unit
a.u.

Shape
[nA1LatticeStrains]

`Energy stress tensor%Electrostatic`

Type
float_array

Description
Electrostatic energy contribution to the stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

`Energy stress tensor%Electrostatic pair`

Type
float_array

Description
Electrostatic pair energy contribution to the stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

`Energy stress tensor%Electrostatic pair(modeSD)`

Type
float_array

Description
Electrostatic pair contribution to the symmetric strain mode derivatives.

Unit
a.u.

Shape
[nA1LatticeStrains]

Energy stress tensor%Electrostatic(modeSD)

Type
float_array

Description
Electrostatic contribution to the symmetric strain mode derivatives.

Unit
a.u.

Shape
[nA1LatticeStrains]

Energy stress tensor%Kinetic

Type
float_array

Description
Kinetic energy contribution to the stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

Energy stress tensor%Kinetic(modeSD)

Type
float_array

Description
Kinetic energy contribution to the mode derivative.

Unit
a.u.

Shape
[nA1LatticeStrains]

Energy stress tensor%nA1LatticeStrains

Type
int

Description
Number of symmetric displacements (modes).

Energy stress tensor%P Mat

Type
float_array

Description
Density matrix contribution to the stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

`Energy stress tensor%P Mat(modeSD)`

Type
float_array

Description
Density matrix contribution to the symmetric strain mode derivatives.

Unit
a.u.

Shape
[nA1LatticeStrains]

`Energy stress tensor%Stress Tensor`

Type
float_array

Description
Stress tensor, dE/de per volume/surface/distance (3D/2D/1D).

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

`Energy stress tensor%Total`

Type
float_array

Description
Stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

`Energy stress tensor%Total(modeSD)`

Type
float_array

Description
Symmetric strain mode derivatives.

Unit
a.u.

Shape
[nA1LatticeStrains]

`Energy stress tensor%XC`

Type
float_array

Description
XC energy contribution to the stress tensor.

Unit
a.u.

Shape
[Molecule%nLatticeVectors, Molecule%nLatticeVectors]

`Energy stress tensor%XC(modeSD)`

Type
float_array

Description
XC contribution to the symmetric strain mode derivatives.

Unit
a.u.

Shape
[nA1LatticeStrains]

Energy terms

Section content: Various terms contributing to the energy.

`Energy terms%ecor(atoms)`

Type
float_array

Description
Exchange plus correlation (atomic correction)terms for the sum of spherical atoms. These are hardcoded total XC functionals such as Becke88X+PW91C.

Unit
hartree

Shape
[14]

`Energy terms%ekin(atoms)`

Type
float

Description
Kinetic (valence) energy for the sum of spherical atoms.

Unit
hartree

`Energy terms%elstt`

Type
float

Description
Electrostatic interaction energy between the spherical atoms.

Unit
hartree

`Energy terms%emadel`

Type
float

Description
Madelung energy in case charged spherical atoms are used (by default never).

Unit
hartree

`Energy terms%etot(atoms)`

Type
float_array

Description
Total energy for the sum of spherical atoms for some hardcoded functionals.

Unit
hartree

Shape
[14]

`Energy terms%excterms(atoms)`

Type
float_array

Description
Exchange and correlation terms for the sum of spherical atoms. A term is like Becke88X.

Unit
hartree

Shape
[20]

`Energy terms%qsett`

Type
float_array

Description
Total charge (number of electrons) per atom set.

Shape
[geometry%natstt]

`Energy terms%qsetv`

Type
float_array

Description
Valence charge (number of electrons) per atom set.

Shape
[geometry%natstt]

FermiSurface

Section content: ?

`FermiSurface%nDimK`

Type
int

Description
?

FermiSurface%nSimplices

Type
int

Description
?

FermiSurface%nSpin

Type
int

Description
?

FermiSurface%nVertices

Type
int

Description
?

FermiSurface%nVerticesPerSimplex

Type
int

Description
?

FermiSurface%VerticesCoords

Type
float_array

Description
?

FermiSurface%VerticesIds

Type
int_array

Description
?

fit

Section content: Information for density fitting by STO functions.

fit%fit coefficients

Type
float_array

Description
The deformation density is approximated by these fit coefficients. rho(x) = sum_i c_i f_i(x).

There are nspinr+1 components. First one is the sum and the last two the separate spin components. Finally nspinr=max(nspin,nspino).

Shape
[nsymft, :]

fit%ifitpat

Type
int_array

Description
If you specify the atom number i, as on input, and the fit function on that atom j, the number of the fit function is { t ifitpat(j,i)}.

Shape
[SystType%nfitt, Molecule%nAtoms]

fit%ifitpati

Type
int_array

Description
If you know the fit function k, it was function { t ifitpati(2,k)} on atom { t ifitpati(1,k)}.

Shape
[2, SystType%nfitt]

fit%ilmfit

Type
int_array

Description
Atom index (internal order), l-value, and m-value for all fit functions.

Shape
[3, SystType%nfitt]

fit%ilsymfit

Type
int_array

Description
The atoms set, ilsymft(1,:), and the l value, ilsymft(2,:), for all symmetric functions.

Shape
[2, nsymft]

fit%lambda

Type
float_array

Description
Lagrange multiplier needed to enforce charge neutrality.

fit%method

Type
int

Description
Switch for the method to be used. 0: automatic, 1: inverse, 2: conj-grad, 3: divide-and fit.

Possible values
[0, 1, 2, 3]

`fit%nforth`

Type
int

Description
Number of fit functions after orthonormalization.

`fit%nsymft`

Type
int

Description
Number of symmetric fit function combinations.

`fit%orthonormal_fit`

Type
bool

Description
The fit basis may be transformed to an orthonormal set.

`fit%projFitRhoDef`

Type
float_array

Description
Projection of the deformation density on the fit functions.

Shape
[nsymft]

`fit%qfit`

Type
float_array

Description
Charge (integrals) of the fit functions. Only non-zero for s-functions.

Shape
[nsymft]

fragment

Section content: A system can be built up from fragments, allowing an energy decomposition. The bonding energy will be with respect to the fragments.

`fragment%deltaShift(standard)`

Type
float_array

Description
How the atoms were (lattice) shifted to match the positions in the fragments.

Unit
bohr

Shape
[3, Molecule%nAtoms]

fragment%filenames

Type
lchar_string_array

Description
Fragment kf files.

Shape
[nfrag]

fragment%FragOcc#

Type
float_array

Description
Occupations for a fragment in the final basis.

Shape
[SystType%nbas, SystType%nspin]

fragment%ifrgat

Type
int_array

Description
The atom mapping from fragment to atoms in the final system.

Shape
[Molecule%nAtoms, nfrag]

fragment%kequif

Type
int_array

Description
How k points in the final system map to k points of the fragments.

Shape
[kspace%kuniqu, nfrag]

fragment%lShift

Type
bool

Description
Whether some atoms needed shifting, affecting the Bloch phase factor. Only relevant when the number of k-points is larger than one.

fragment%nfrag

Type
int

Description
Number of fragments.

`fragment%nfragocc`

Type
int_array

Description
Energy ordered orbitals may be divided into three parts 1: occupied, 2: virtual, 3: sea (very high lying orbitals). Normally the sea is empty.

Shape
[3, nfrag, SystType%nspin, kspace%kuniqu]

FuzzyUnitCell

Section content: Becke-style unit cell partition function.

`FuzzyUnitCell%CoordsFuzzyAtoms`

Type
float_array

Description
Coordinates of the atoms inside the region where the fuzzy unit cell is not zero.

Shape
[3, nFuzzyAtoms]

`FuzzyUnitCell%nCells`

Type
int

Description
Number of cells needed for a fuzzy cell summation.

`FuzzyUnitCell%nFuzzyAtoms`

Type
int

Description
Number of atoms inside the region where the fuzzy unit cell is not zero.

`FuzzyUnitCell%PartitionFunctionOnAtoms`

Type
float_array

Description
Value of the partition function at the atomic positions.

Shape
[nFuzzyAtoms]

`FuzzyUnitCell%qFuzzyAtoms`

Type
float_array

Description
Nuclear charges (atom number) of the atoms inside the region where the fuzzy unit cell is not zero.

Shape
[nFuzzyAtoms]

FuzzyUnitCell%UnitCellAtomIndex

Type
int_array

Description
Cell index for all the fuzzy atoms.

Shape
[nFuzzyAtoms]

FuzzyUnitCell%xyzCells

Type
float_array

Description
Coordinates of the cells needed for the fuzzy cell summation.

Shape
[3, nCells]

General

Section content: General information about the BAND calculation.

General%account

Type
string

Description
Name of the account from the license

General%engine input

Type
string

Description
The text input of the engine.

General%engine messages

Type
string

Description
Message from the engine. In case the engine fails to solves, this may contains extra information on why.

General%file-ident

Type
string

Description
The file type identifier, e.g. RKF, RUNKF, TAPE21…

General%jobid

Type
int

Description
Unique identifier for the job.

General%program

Type
string

Description
The name of the program/engine that generated this kf file.

General%release

Type
string

Description
The version of the program that generated this kf file (including svn revision number and date).

General%termination status

Type
string

Description
The termination status. Possible values: 'NORMAL TERMINATION', 'NORMAL TERMINATION with warnings', 'NORMAL TERMINATION with errors', 'ERROR', 'IN PROGRESS'.

General%title

Type
string

Description
Title of the calculation.

General%uid

Type
string

Description
SCM User ID

General%version

Type
int

Description
Version number?

geometry

Section content: Information on the geometry.

geometry%Atom map new order

Type
int_array

Description
From input to internal: internalAtomIndex = array(inputAtomIndex).

Shape
[natomt]

geometry%Atom map old order

Type
int_array

Description
From internal to input: inputAtomIndex = array(internalAtomIndex).

Shape
[natomt]

geometry%atomTypeString

Type
lchar_string_array

Description
?.

Shape
[Molecule%nAtoms]

geometry%distances(atom sets)

Type
float_array

Description
Half matrix containing distances between atom sets.

geometry%input_lattice

Type
float_array

Description
The lattice vectors (if any) as specified on input.

Unit
bohr

Shape
[3, 3]

geometry%input_xyzatm

Type
float_array

Description
The coordinates of the atoms as specified on input.

Unit
bohr

Shape
[3, natomt]

geometry%itypat

Type
int_array

Description
The type in the range [1:ntyp] for all atoms.

Shape
[natomt]

geometry%mdim

Type
int

Description
Dimension of the molecule, i.e. 1 for something linear, 2 for a flat one.

geometry%natom

Type
int_array

Description
For each set the number of atoms.

Shape
[natstt]

geometry%natomt

Type
int

Description
Number of atoms.

geometry%natst

Type
int_array

Description
Number of atom sets for all types.

Shape
[ntyp]

geometry%natstt

Type
int

Description
Number of atom sets. A set consists of symmetry equivalent atoms.

geometry%ntyp

Type
int

Description
Number of types, i.e. atoms with a different basis set, or accuracy setting (or different region).

geometry%Number of selected atoms

Type
int

Description
Number of selected atoms.

geometry%qatm

Type
float_array

Description
Charge of the nuclei (almost always the atomic number).

Shape
[ntyp]

geometry%qelec

Type
float

Description
The number of valence electrons (not including a nett charge of the system).

geometry%qset

Type
float_array

Description
Valence charge per atom set.

Shape
[natstt]

geometry%removeZSymmetry

Type
bool

Description
Remove z symmetry from the symmetry operator set.

geometry%Selected atoms (input order)

Type
int_array

Description
List of selected atoms.

Shape
[Number of selected atoms]

geometry%standard_lattice

Type
float_array

Description
The lattice vectors (if any) in the standard frame.

Unit
bohr

Shape
[3, 3]

geometry%standard_xyzatm

Type
float_array

Description
The coordinates of the atoms after transforming to the standard frame.

Unit
bohr

Shape
[3, natomt]

geometry%xyzatm(atoms_in_new_order)

Type
float_array

Description
The coordinates of the atoms in the internal order.

Unit
bohr

Shape
[3, natomt]

GeomType

Section content: Geometry related info.

GeomType%avec

Type
float_array

Description
The lattice stored as a 3xnLatticeVectors matrix. Only the ndimk,ndimk part has meaning.

Unit
bohr

Shape
[kspace%ndim, kspace%ndim]

GeomType%bvec

Type
float_array

Description
The inverse lattice stored as a 3x3 matrix. Only the ndimk,ndimk part has meaning.

Unit
1/bohr

Shape
[kspace%ndim, kspace%ndim]

GeomType%natomt

Type
int

Description
Number of atoms. (Same as geometry%natomt)

GeomType%natstt

Type
int

Description
Number of atom sets. A set consists of symmetry equivalent atoms. (Same as geometry%natstt)

GeomType%ndim

Type
int

Description
Number of dimensions for the molecule. Water is flat and has two dimensions.

GeomType%ndimk

Type
int

Description
Number of dimensions for k-space integration. Normally this is the number of lattice vectors.

GeomType%noper

Type
int

Description
Number of symmetry operators (real space). (Same as Symmetry%Nr. of operators)

GeomType%noperk

Type
int

Description
Number of symmetry operators (reciprocal space). (Same as Symmetry Nr. of operators (k-space))

GeomType%ntyp

Type
int

Description
Number of types, i.e. atoms with a different basis set, or accuracy setting (or different region). (Same as geometry%ntyp)

GeomType%Serializer::type

Type
string_fixed_length

Description
Information for the Serializer code.

GeomType%stdrot

Type
float_array

Description
Rotation to the standard frame. The point group part (P) of x_standard = P x_input + t.

Shape
[3, 3]

GeomType%stdvec

Type
float_array

Description
Translation to the standard frame. The translation part (t) of x_standard = P x_input + t.

Shape
[3]

GGA bond terms (c)
Section content: XC energy terms according to some hardcoded list of functionals (correlation part).

GGA bond terms (c)%BLYP

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%BOP

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%BP

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%FT97

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%HCTH/120

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%HCTH/147

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%HCTH/407

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%HCTH/93

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%KCIS-modified

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%KCIS-original

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%KT1

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%KT2

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%LAK

Type
float

Description
?

GGA bond terms (c)%LAKc

Type
float

Description
?

GGA bond terms (c)%LAKx

Type
float

Description
?

GGA bond terms (c)%LDA(VWN)

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%M06-L

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%M11-L

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%mPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%mPBEKCIS

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%mPW

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%MS0

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%MS1

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%MS2

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%MVS

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%MVSx

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%OLYP

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%OPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%OPerdew

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%PBE

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%PBEsol

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%PKZB

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%PKZBx-KCIScor

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%PW91

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%revPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%revTPSS

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%RGE2

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%RPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%SCAN

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%SCANc

Type
float

Description
?

GGA bond terms (c)%SCANx

Type
float

Description
?

GGA bond terms (c)%SOGGA

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%SOGGA11

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%SSB-D

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%TASK

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%TASKCC

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%TASKCCALDA

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%TASKLDA

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%TASKSCAN

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%TASKxc

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%tau-HCTH

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%TPSS

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%VS98

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%VS98-x(xc)

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%VS98-x-only

Type
float

Description
.

Unit
hartree

GGA bond terms (c)%XLYP

Type
float

Description
.

Unit
hartree

GGA bond terms (x)
Section content: XC energy terms according to some hardcoded list of functionals (exchange part).

GGA bond terms (x)%BLYP

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%BOP

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%BP

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%FT97

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%HCTH/120

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%HCTH/147

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%HCTH/407

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%HCTH/93

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%KCIS-modified

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%KCIS-original

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%KT1

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%KT2

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%LAK

Type
float

Description
?

GGA bond terms (x)%LAKc

Type
float

Description
?

GGA bond terms (x)%LAKx

Type
float

Description
?

GGA bond terms (x)%LDA(VWN)

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%M06-L

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%M11-L

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%mPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%mPBEKCIS

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%mPW

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%MS0

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%MS1

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%MS2

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%MVS

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%MVSx

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%OLYP

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%OPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%OPerdew

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%PBE

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%PBEsol

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%PKZB

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%PKZBx-KCIScor

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%PW91

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%revPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%revTPSS

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%RGE2

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%RPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%SCAN

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%SCANc

Type
float

Description
?

GGA bond terms (x)%SCANx

Type
float

Description
?

GGA bond terms (x)%SOGGA

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%SOGGA11

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%SSB-D

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%TASK

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%TASKCC

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%TASKCCALDA

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%TASKLDA

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%TASKSCAN

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%TASKxc

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%tau-HCTH

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%TPSS

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%VS98

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%VS98-x(xc)

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%VS98-x-only

Type
float

Description
.

Unit
hartree

GGA bond terms (x)%XLYP

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)
Section content: XC energies according to some hardcoded list of functionals.

GGA bond terms (xc)%BLYP

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%BOP

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%BP

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%FT97

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%HCTH/120

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%HCTH/147

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%HCTH/407

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%HCTH/93

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%KCIS-modified

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%KCIS-original

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%KT1

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%KT2

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%LAK

Type
float

Description
?

GGA bond terms (xc)%LAKc

Type
float

Description
?

GGA bond terms (xc)%LAKx

Type
float

Description
?

GGA bond terms (xc)%LDA(VWN)

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%M06-L

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%M11-L

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%mPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%mPBEKCIS

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%mPW

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%MS0

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%MS1

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%MS2

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%MVS

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%MVSx

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%OLYP

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%OPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%OPerdew

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%PBE

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%PBEsol

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%PKZB

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%PKZBx-KCIScor

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%PW91

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%revPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%revTPSS

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%RGE2

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%RPBE

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%SCAN

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%SCANc

Type
float

Description
?

GGA bond terms (xc)%SCANx

Type
float

Description
?

GGA bond terms (xc)%SOGGA

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%SOGGA11

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%SSB-D

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%TASK

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%TASKCC

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%TASKCCALDA

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%TASKLDA

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%TASKSCAN

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%TASKxc

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%tau-HCTH

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%TPSS

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%VS98

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%VS98-x(xc)

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%VS98-x-only

Type
float

Description
.

Unit
hartree

GGA bond terms (xc)%XLYP

Type
float

Description
.

Unit
hartree

green

Section content: ?

green%NOld

Type
float

Description
?

green%pmatim

Type
float_array

Description
Imaginary part of the density matrix.

Shape
[SystType%nbas, SystType%nbas]

green%pmatre

Type
float_array

Description
Real part of the density matrix.

Shape
[SystType%nbas, SystType%nbas]

green%shift_add

Type
float

Description
Shift added to the potential.

GW

Section content: ?

GW%freqGrid

Type
float_array

Description
?

GW%G0W0_QP_hole_ener

Type
float_array

Description
?

GW%G0W0_QP_hole_ener_dif

Type
float_array

Description
?

GW%G0W0_QP_hole_ener_sp_A

Type
float_array

Description
?

GW%G0W0_QP_hole_ener_sp_A_dif

Type
float_array

Description
?

GW%G0W0_QP_hole_ener_sp_B

Type
float_array

Description
?

GW%G0W0_QP_hole_ener_sp_B_dif

Type
float_array

Description
?

GW%G0W0_QP_hole_energies

Type
float

Description
?

GW%G0W0_QP_hole_energies_diff

Type
float

Description
?

GW%G0W0_QP_part_ener

Type
float_array

Description
?

GW%G0W0_QP_part_ener_dif

Type
float_array

Description
?

GW%G0W0_QP_part_ener_sp_A

Type
float_array

Description
?

GW%G0W0_QP_part_ener_sp_A_dif

Type
float_array

Description
?

GW%G0W0_QP_part_ener_sp_B

Type
float_array

Description
?

GW%G0W0_QP_part_ener_sp_B_dif

Type
float_array

Description
?

GW%G0W0_QP_particle_energies

Type
float

Description
?

GW%G0W0_QP_particle_energies_diff

Type
float

Description
?

GW%G3W2

Type
float_array

Description
?

GW%G3Wp2

Type
float_array

Description
?

GW%G3Wp2 - GWGamma

Type
float_array

Description
?

GW%GLRatio

Type
float_array

Description
?

GW%nBas

Type
int_array

Description
?

GW%nFit

Type
int

Description
?

GW%nFreq

Type
int

Description
?

GW%nFreqTotal

Type
int

Description
?

GW%nInnerLoopIterations

Type
int

Description
?

GW%nInnerLoopIterationsTotal

Type
int

Description
?

GW%nIterations

Type
int

Description
?

GW%normV

Type
float

Description
?

GW%normW0

Type
float

Description
?

GW%nremov

Type
int

Description
?

GW%nStates

Type
int

Description
?

GW%nTime

Type
int

Description
?

GW%nTimeTotal

Type
int

Description
?

GW%QPocc

Type
float_array

Description
?

GW%QPocc_A

Type
float_array

Description
?

GW%QPocc_B

Type
float_array

Description
?

GW%QPun

Type
float_array

Description
?

GW%QPun_A

Type
float_array

Description
?

GW%QPun_B

Type
float_array

Description
?

GW%SCGW_QP_hole_ener

Type
float_array

Description
?

GW%SCGW_QP_hole_ener_dif

Type
float_array

Description
?

GW%SCGW_QP_hole_energies

Type
float_array

Description
?

GW%SCGW_QP_hole_energies_diff

Type
float_array

Description
?

GW%SCGW_QP_part_ener

Type
float_array

Description
?

GW%SCGW_QP_part_ener_dif

Type
float_array

Description
?

GW%SCGW_QP_particle_energies

Type
float_array

Description
?

GW%SCGW_QP_particle_energies_diff

Type
float_array

Description
?

GW%SOSEX

Type
float_array

Description
?

GW%SOSEX2

Type
float_array

Description
?

GW%SOSEXINF

Type
float_array

Description
?

GW%spectral_*

Type
float_array

Description
?

GW%SSOXcorrection

Type
float_array

Description
?

GW%SSOXgreater

Type
float_array

Description
?

GW%SSOXlesser

Type
float_array

Description
?

GW%tMat_A

Type
float_array

Description
?

GW%tMat_B

Type
float_array

Description
?

HartreeFock
Section content: Section for hybrid functionals.

HartreeFock%EnergyContribution

Type
float

Description
Energy contribution from the HF part.

Unit
hartree

HFExchangeEval
Section content: Partial information about the (periodic) RIHartreeFock method to be used for analysis, and not relevant to band itself. The content has a different purpose than the section with the same name on the file RIHartreeFock

HFExchangeEval%InteractingAtoms

Type
bool_array

Description
Which atoms do interact

Shape
[Molecule%nAtoms, Molecule%nAtoms, nCells]

HFExchangeEval%nCells

Type
int

Description
Number of cells.

HFExchangeEval%nCellsForP

Type
int

Description
Number of cells used for the density matrix P(R). The density matrix P(R) has artificial lattice copies, and these should not be used.

K-Matrices

Section content: Storage of matrices depending on k

K-Matrices%Data(#{matrix})

Type
float_array

Description
Contents of the matrix. The outer loop is over unique k-points, followed by the real part of the matrix, and if complex, followd by the imaginary part.

K-Matrices%Dimensions(#{matrix})

Type
int_array

Description
Dimensions of the matrix for each k-point.

K-Matrices%DimensionsX(#{matrix})

Type
int_array

Description
Maximum dimensions of the matrix. Sometimes dimensions can be smaller than the allocated space (for instance when not all eigenvectors are stored). Same length as Dimensions.

K-Matrices%IsKunComplex

Type
bool_array

Description
Are matrices real or complex per unique k-point.

Shape
[kspace%kuniqu]

K-Matrices%Name(#{matrix})

Type
string_fixed_length

Description
Name of the matrix.

K-Matrices%nMatrices

Type
int

Description
Number of matrices stored in this section.

K-Matrices%ReadCount(#{matrix})

Type
int

Description
Number of times that the matrix has been read.

K-Matrices%StorageMode

Type
int

Description
Information may be distributed over nodes. 1) fully distributed, 2) Intra node distributed, 3) Master-only 4) Not distributed. For serial calculations this is all the same.

K-Matrices:AlwaysComplex
Section content: Storage of matrices depending on k. This section is for those matrices that are always stored as complex.

K-Matrices:AlwaysComplex%Data(#{matrix})

Type
float_array

Description
Contents of the matrix. The outer loop is over unique k-points, followed by the real part of the matrix, and if complex, followd by the imaginary part.

K-Matrices:AlwaysComplex%Dimensions(#{matrix})

Type
int_array

Description
Dimensions of the matrix for each k-point.

K-Matrices:AlwaysComplex%DimensionsX(#{matrix})

Type
int_array

Description
Maximum dimensions of the matrix. Sometimes dimensions can be smaller than the allocated space (for instance when not all eigenvectors are stored). Same length as Dimensions.

K-Matrices:AlwaysComplex%IsKunComplex

Type
bool_array

Description
Are matrices real or complex per unique k-point.

Shape
[kspace%kuniqu]

K-Matrices:AlwaysComplex%Name(#{matrix})

Type
string_fixed_length

Description
Name of the matrix.

K-Matrices:AlwaysComplex%nMatrices

Type
int

Description
Number of matrices stored in this section.

K-Matrices:AlwaysComplex%ReadCount(#{matrix})

Type
int

Description
Number of times that the matrix has been read.

K-Matrices:AlwaysComplex%StorageMode

Type
int

Description
Information may be distributed over nodes. 1) fully distributed, 2) Intra node distributed, 3) Master-only 4) Not distributed. For serial calculations this is all the same.

KFDefinitions

Section content: The definitions of the data on this file

KFDefinitions%json

Type
string

Description
The definitions of the data on this file in json.

KPointsConfig

Section content: Configuration information for the k-space integration points. This is all about splitting the space into simplices.

KPointsConfig%commentString

Type
string_fixed_length

Description
Description on how the grid was generated.

KPointsConfig%interpolation

Type
int

Description
How to interpolate the bands over the simplices. 1) linear method, 2) quadratic method.

KPointsConfig%method

Type
int

Description
Method used to divide space into simplices 1) symmetric method, 2) grid. The k-space integration method is the same.

KPointsConfig%ndimk

Type
int

Description
Dimension of reciprocal space.

KPointsConfig%parameters

Type
int_array

Description
For the symmetric method parameters(1) will be the accuracy parameter. For the regular method the parameters are per lattice vector.

Shape
[3]

KPointsConfig%splitCubeInSix

Type
bool

Description
We want to split a cube in tetrahedra. The minimal number needed is five. With six it looks more symmetrical.

KPointsConfig%splitLongest

Type
bool

Description
Sometimes a tetrahedron needs to be split, and there is some arbitrariness in how to do that. With this option the longest edge will be used.

KPointsConfig%splitPermutation

Type
int_array

Description
There is some arbitrariness when splitting a volume. Also the order of the splitting has some effect.

KPointsConfig%symSampling

Type
bool

Description
Map out points from Irr. BZ of the pure lattice (neglecting atoms).

kspace

Section content: Info regarding the k-space integration…

kspace%avec

Type
float_array

Description
The lattice stored as a 3xnLatticeVectors matrix. Only the ndimk,ndimk part has meaning.

Unit
bohr

Shape
[3, :]

kspace%bvec

Type
float_array

Description
The inverse lattice stored as a 3x3 matrix. Only the ndimk,ndimk part has meaning.

Unit
1/bohr

Shape
[ndim, ndim]

kspace%bzvol

Type
float

Description
The volume of the BZ zone. In 2D it is the surface and in 1D it is the length. The unit is bohr raised to the power ndim.

kspace%iDimkEffective

Type
int_array

Description
Which latttice vectors are really used for the k-space integration.

Shape
[nDimkEffective]

kspace%isKunComplex

Type
bool_array

Description
Whether or not the Hamiltonian matrix is complex for a unique k-point.

Shape
[kuniqu]

kspace%kequiv

Type
int_array

Description
When kequiv(i)=i the k-point is unique.

Shape
[kt]

kspace%kequn

Type
int_array

Description
When looping over all k-points, the unique index is kun=kequn(k).

Shape
[kt]

kspace%kinteg

Type
int

Description
In case a symmetric grid is used this is the parameter used to create it.

kspace%klbl

Type
lchar_string_array

Description
labels describing the k-points

Shape
[kt]

kspace%klblun

Type
lchar_string_array

Description
labels describing the unique k-points

Shape
[kuniqu]

kspace%klnear

Type
bool

Description
Whether or not linear k-space integration is used (symmetric method with even kinteg).

kspace%ksimpl

Type
int_array

Description
Index array defining the simplices, referring to the xyzpt array.

Shape
[nvertk, nsimpl]

kspace%kt

Type
int

Description
The total number of k-points used by the k-space to sample the unique wedge of the Brillouin zone.

kspace%ktBoltz

Type
float

Description
band only?.

kspace%kuniqu

Type
int

Description
The number of symmetry unique k-points where an explicit diagonalization is needed. Smaller or equal to kt.

kspace%ndim

Type
int

Description
The nr. of lattice vectors.

kspace%ndimk

Type
int

Description
The nr. of dimensions used in the k-space integration.

kspace%nDimkEffective

Type
int

Description
Normally ndimk is equal to the number of lattice vectors. For very large lattice vectors the k-space dispersion is ignored, leading to a lower dimensional band structure.

kspace%noperk

Type
int

Description
The nr. of operators in k-space. band only?

kspace%nsimpl

Type
int

Description
The number of simplices constructed from the k-points to span the IBZ.

kspace%numBoltz

Type
int

Description
Number of energies to sample around the fermi energy. band only?

kspace%numEquivSimplices

Type
int_array

Description
Simplices may be equivalent due to symmetry operations..

Shape
[nsimpl]

kspace%nvertk

Type
int

Description
The number of vertices that each simplex has.

kspace%operk

Type
float_array

Description
Symmetry operators in k-space. band only?

Unit
bohr

Shape
[ndim, ndim, noperk]

kspace%xyzpt

Type
float_array

Description
The coordinates of the k-points.

Unit
1/bohr

Shape
[ndimk, kt]

kspace(primitive cell)
Section content: should not be here!!!

kspace(primitive cell)%avec

Type
float_array

Description
The lattice stored as a 3xnLatticeVectors matrix. Only the ndimk,ndimk part has meaning.

Unit
bohr

Shape
[3, :]

kspace(primitive cell)%bvec

Type
float_array

Description
The inverse lattice stored as a 3x3 matrix. Only the ndimk,ndimk part has meaning.

Unit
1/bohr

Shape
[ndim, ndim]

kspace(primitive cell)%kt

Type
int

Description
The total number of k-points used by the k-space to sample the unique wedge of the Brillouin zone.

kspace(primitive cell)%kuniqu

Type
int

Description
The number of symmetry unique k-points where an explicit diagonalization is needed. Smaller or equal to kt.

kspace(primitive cell)%ndim

Type
int

Description
The nr. of lattice vectors.

kspace(primitive cell)%ndimk

Type
int

Description
The nr. of dimensions used in the k-space integration.

kspace(primitive cell)%xyzpt

Type
float_array

Description
The coordinates of the k-points.

Unit
1/bohr

Shape
[ndimk, kt]

Low Frequency Correction

Section content: Configuration for the Head-Gordon Dampener-powered Free Rotor Interpolation.

Low Frequency Correction%Alpha

Type
float

Description
Exponent term for the Head-Gordon dampener.

Low Frequency Correction%Frequency

Type
float

Description
Frequency around which interpolation happens, in 1/cm.

Low Frequency Correction%Moment of Inertia

Type
float

Description
Used to make sure frequencies of less than ca. 1 1/cm don't overestimate entropy, in kg m^2.

Magnetic properties

Section content: When applying a finite magnetic field.

Magnetic properties%BField

Type
float_array

Description
Applied b field.

Shape
[3]

Magnetic properties%DipoleAtom

Type
int

Description
If this is present the applied field is a dipole with this atom index.

Magnetic properties%InducedBFieldAtNuclei

Type
float_array

Description
The external BField induces a current, and hence a bfield. Here given at the nuclear positions. This defines the shielding: how much is it changed from the external field at the nuclei.

Shape
[3, Molecule%nAtoms]

Magnetic properties%NmrShieldingAllAtoms(ppm)

Type
float_array

Description
Shielding tensor for all the atoms. This is calculated analytically, not by using a finite magnetic field.

Shape
[3, 3, Molecule%nAtoms]

Magnetic properties%ShieldingRowAtNuclei(ppm)

Type
float_array

Description
A row of the shielding tensor for all nuclei.

Shape
[3, Molecule%nAtoms]

Matrices
Section content: Section that can contain any number of real matrices

`Matrices%Data(#)`

Type
float_array

Description
The array, rank and dimensions as specified by Dimensions.

`Matrices%Dimensions(#)`

Type
int_array

Description
The dimensions of the array

`Matrices%Name(#)`

Type
string

Description
The name of the matrix.

`Matrices%nEntries`

Type
int

Description
The number of matrices

`Matrices%Type(#)`

Type
string

Description
The type such as Real, and perhaps Complex?

metaGGA:atoms
Section content: XC energies for the sum of all reference atoms.

`metaGGA:atoms%ec`

Type
float_array

Description
Correlation energies.

Unit
hartree

Shape
[numXC]

metaGGA:atoms%ex

Type
float_array

Description
Exchange energies.

Unit
hartree

Shape
[numXC]

metaGGA:atoms%exc

Type
float_array

Description
Exchange correlation energies.

Unit
hartree

Shape
[numXC]

metaGGA:atoms%numXC

Type
int

Description
Number of xc energies.

MGGAOEP
Section content: OEP for meta GGAs.

MGGAOEP%nband

Type
int

Description
Number of bands.

MGGAOEP%valKLIIntRes

Type
float_array

Description
?

Shape
[valKLIIntResdim1, valKLIIntResdim2, valKLIIntResdim3]

MGGAOEP%valKLIIntResdim1

Type
int

Description
First dimension.

MGGAOEP%valKLIIntResdim2

Type
int

Description
Second dimension.

MGGAOEP%valKLIIntResdim3

Type
int

Description
Third dimension.

MGGAOEP%valKLITauIntRes

Type
float_array

Description
?

Shape
[valKLITauIntResdim1, valKLITauIntResdim2, valKLITauIntResdim3]

MGGAOEP%valKLITauIntResdim1

Type
int

Description
First dimension.

MGGAOEP%valKLITauIntResdim2

Type
int

Description
Second dimension.

MGGAOEP%valKLITauIntResdim3

Type
int

Description
Third dimension.

Mobile Block Hessian
Section content: Mobile Block Hessian.

Mobile Block Hessian%Coordinates Internal

Type
float_array

Description
?

Mobile Block Hessian%Free Atom Indexes Input

Type
int_array

Description
?

Mobile Block Hessian%Frequencies in atomic units

Type
float_array

Description
?

Mobile Block Hessian%Frequencies in wavenumbers

Type
float_array

Description
?

Mobile Block Hessian%Input Cartesian Normal Modes

Type
float_array

Description
?

Mobile Block Hessian%Input Indexes of Block #

Type
int_array

Description
?

Mobile Block Hessian%Intensities in km/mol

Type
float_array

Description
?

Mobile Block Hessian%MBH Curvatures

Type
float_array

Description
?

Mobile Block Hessian%Number of Blocks

Type
int

Description
Number of blocks.

Mobile Block Hessian%Sizes of Blocks

Type
int_array

Description
Sizes of the blocks.

Shape
[Number of Blocks]

Molecule

Section content: The input molecule of the calculation.

`Molecule%AtomicNumbers`

Type
int_array

Description
Atomic number 'Z' of the atoms in the system

Shape
[nAtoms]

`Molecule%AtomMasses`

Type
float_array

Description
Masses of the atoms

Unit
a.u.

Values range
[0, '\infinity']

Shape
[nAtoms]

`Molecule%AtomSymbols`

Type
string

Description
The atom's symbols (e.g. 'C' for carbon)

Shape
[nAtoms]

`Molecule%bondOrders`

Type
float_array

Description
The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

`Molecule%Charge`

Type
float

Description
Net charge of the system

Unit
e

Molecule%Coords

Type
float_array

Description
Coordinates of the nuclei (x,y,z)

Unit
bohr

Shape
[3, nAtoms]

Molecule%eeAttachTo

Type
int_array

Description
A multipole may be attached to an atom. This influences the energy gradient.

Molecule%eeChargeWidth

Type
float

Description
If charge broadening was used for external charges, this represents the width of the charge distribution.

Molecule%eeEField

Type
float_array

Description
The external homogeneous electric field.

Unit
hartree/(e*bohr)

Shape
[3]

Molecule%eeLatticeVectors

Type
float_array

Description
The lattice vectors used for the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNLatticeVectors]

Molecule%eeMulti

Type
float_array

Description
The values of the external point- or multipole- charges.

Unit
a.u.

Shape
[eeNZlm, eeNMulti]

Molecule%eeNLatticeVectors

Type
int

Description
The number of lattice vectors for the external point- or multipole- charges.

Molecule%eeNMulti

Type
int

Description
The number of external point- or multipole- charges.

Molecule%eeNZlm

Type
int

Description
When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

Molecule%eeUseChargeBroadening

Type
bool

Description
Whether or not the external charges are point-like or broadened.

Molecule%eeXYZ

Type
float_array

Description
The position of the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNMulti]

Molecule%EngineAtomicInfo

Type
string_fixed_length

Description
Atom-wise info possibly used by the engine.

`Molecule%fromAtoms`

Type
int_array

Description
Index of the first atom in a bond. See the bondOrders array

`Molecule%latticeDisplacements`

Type
int_array

Description
The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

`Molecule%LatticeVectors`

Type
float_array

Description
Lattice vectors

Unit
bohr

Shape
[3, nLatticeVectors]

`Molecule%nAtoms`

Type
int

Description
The number of atoms in the system

`Molecule%nAtomsTypes`

Type
int

Description
The number different of atoms types

`Molecule%nLatticeVectors`

Type
int

Description
Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values
[0, 1, 2, 3]

`Molecule%toAtoms`

Type
int_array

Description
Index of the second atom in a bond. See the bondOrders array

MoleculeSuperCell
Section content: The system used for the numerical phonon super cell calculation.

`MoleculeSuperCell%AtomicNumbers`

Type
int_array

Description
Atomic number 'Z' of the atoms in the system

Shape
[nAtoms]

`MoleculeSuperCell%AtomMasses`

Type
float_array

Description
Masses of the atoms

Unit
a.u.

Values range
[0, '\infinity']

Shape
[nAtoms]

`MoleculeSuperCell%AtomSymbols`

Type
string

Description
The atom's symbols (e.g. 'C' for carbon)

Shape
[nAtoms]

`MoleculeSuperCell%bondOrders`

Type
float_array

Description
The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

`MoleculeSuperCell%Charge`

Type
float

Description
Net charge of the system

Unit
e

MoleculeSuperCell%Coords

Type
float_array

Description
Coordinates of the nuclei (x,y,z)

Unit
bohr

Shape
[3, nAtoms]

MoleculeSuperCell%eeAttachTo

Type
int_array

Description
A multipole may be attached to an atom. This influences the energy gradient.

MoleculeSuperCell%eeChargeWidth

Type
float

Description
If charge broadening was used for external charges, this represents the width of the charge distribution.

MoleculeSuperCell%eeEField

Type
float_array

Description
The external homogeneous electric field.

Unit
hartree/(e*bohr)

Shape
[3]

MoleculeSuperCell%eeLatticeVectors

Type
float_array

Description
The lattice vectors used for the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNLatticeVectors]

MoleculeSuperCell%eeMulti

Type
float_array

Description
The values of the external point- or multipole- charges.

Unit
a.u.

Shape
[eeNZlm, eeNMulti]

MoleculeSuperCell%eeNLatticeVectors

Type
int

Description
The number of lattice vectors for the external point- or multipole- charges.

MoleculeSuperCell%eeNMulti

Type
int

Description
The number of external point- or multipole- charges.

MoleculeSuperCell%eeNZlm

Type
int

Description
When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

MoleculeSuperCell%eeUseChargeBroadening

Type
bool

Description
Whether or not the external charges are point-like or broadened.

MoleculeSuperCell%eeXYZ

Type
float_array

Description
The position of the external point- or multipole- charges.

Unit
bohr

Shape
[3, eeNMulti]

MoleculeSuperCell%EngineAtomicInfo

Type
string_fixed_length

Description
Atom-wise info possibly used by the engine.

MoleculeSuperCell%fromAtoms

Type
int_array

Description
Index of the first atom in a bond. See the bondOrders array

MoleculeSuperCell%latticeDisplacements

Type
int_array

Description
The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

MoleculeSuperCell%LatticeVectors

Type
float_array

Description
Lattice vectors

Unit
bohr

Shape
[3, nLatticeVectors]

MoleculeSuperCell%nAtoms

Type
int

Description
The number of atoms in the system

MoleculeSuperCell%nAtomsTypes

Type
int

Description
The number different of atoms types

MoleculeSuperCell%nLatticeVectors

Type
int

Description
Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values
[0, 1, 2, 3]

MoleculeSuperCell%toAtoms

Type
int_array

Description
Index of the second atom in a bond. See the bondOrders array

MP2 energies
Section content: ?

MP2 energies%Contribution to DH energy

Type
float

Description
?

MP2 energies%LT-MP2 energy

Type
float

Description
?

MP2 energies%os LT-MP2 energy

Type
float

Description
?

MP2 energies%os RI-MP2 energy

Type
float

Description
?

MP2 energies%RI-MP2 energy

Type
float

Description
?

MP2 energies%ss LT-MP2 energy

Type
float

Description
?

MP2 energies%ss RI-MP2 energy

Type
float

Description
?

NAOSetCells

Section content: For periodic systems neighboring cells need to be considered. More cells are needed for more diffuse basis sets.

`NAOSetCells%Coords(#{entry})`

Type
float_array

Description
Cell coordinates for a basis set.

Shape
[3, nCells(#{entry})]

`NAOSetCells%Name(#{entry})`

Type
string

Description
The name of the basis set.

`NAOSetCells%nAtoms(#{entry})`

Type
int

Description
Number of atoms for a basis set.

`NAOSetCells%nCells(#{entry})`

Type
int

Description
Number of cells needed for a basis set.

`NAOSetCells%nEntries`

Type
int

Description
The number of entries (basis sets), for basis sets like valence and core, fit, etc..

`NAOSetCells%SkipAtom(#{entry})`

Type
bool_array

Description
Sometimes the functions of an atom do not require a cell at all.

Shape
[nAtoms(#{entry}), nCells(#{entry})]

NEGF

Section content: NEGF models the electron transport through a device.

`NEGF%ContactShift`

Type
float

Description
?

NEGF%current

Type
float_array

Description
Current from one contact to the other.

Shape
[nSpin]

NEGF%DeltaPhi0

Type
float

Description
To do with the alignment.

NEGF%DeltaPhi1

Type
float

Description
To do with the alignment of the potential to the bulk potential.

NEGF%dos

Type
float_array

Description
Density of states.

Shape
[nEnergies, nSpin]

NEGF%energyGrid

Type
float_array

Description
Energies for the NEGF results.

Shape
[nEnergies]

NEGF%nEnergies

Type
int

Description
Number of energies

NEGF%nSpin

Type
int

Description
Number of spin components.

NEGF%OffsetShift

Type
float

Description
?

NEGF%transmission

Type
float_array

Description
Transmission.

Shape
[nEnergies, nSpin]

NeutralizingDensity
Section content: For charged (periodic) cells a neutralizing density is used.

NeutralizingDensity%neutralizingFactor

Type
float

Description
The fixed neutralizing density needs to be multiplied with this factor to make the system neutral (after adding it).

Num Int Params
Section content: Parameters needed for the Voronoi grid. It is tried to integrate a set of functions up to a certain accuracy.

Num Int Params%accint

Type
float

Description
Desired accuracy thought of as 10**(-accint).

Num Int Params%accout

Type
float

Description
Setting of the accint parameter for the outer region.

Num Int Params%accpyr

Type
float

Description
Setting of the accint parameter for the pyramids.

Num Int Params%accsph

Type
float

Description
Setting of the accint parameter for the spheres.

Num Int Params%alfas

Type
float_array

Description
Exponents to use for the test functions. Per (radial) order a minimum and maximum alpha is given.

Shape
[npowx, 2, ntyps]

Num Int Params%ldim

Type
int

Description
Number of lattice vectors.

Num Int Params%linteg all

Type
int_array

Description
Per element the l-value to integrate.

Shape
[ntyps]

Num Int Params%lintgx

Type
int

Description
Maximum l-value to integrate.

Num Int Params%nnucs

Type
int

Description
Number of atoms, possibly including point charges.

Num Int Params%noper

Type
int

Description
Number of symmetry operators.

Num Int Params%npowx

Type
int

Description
The radial part of a STO test function will be tried up to this power?

Num Int Params%nratst1

Type
int_array

Description
Index array. nratst1(ityp) will be the first atom of that type.

Shape
[ntyps+1]

Num Int Params%ntyps

Type
int

Description
Number of non equivalent atoms (either by atomic number or basis set).

Num Int Params%oper

Type
float_array

Description
Point group part of the symmetry operators.

Shape
[3, 3, noper]

Num Int Params%qatm

Type
float_array

Description
Atomic numbers.

Shape
[ntyps]

Num Int Params%sphgrid

Type
bool_array

Description
For point charges in xyzatm some may not need a spherical grid when far away.

Shape
[nnucs]

Num Int Params%transl

Type
float_array

Description
Translational part of the symmetry operators (for periodic systems).

Shape
[3, noper]

Num Int Params%vlatt

Type
float_array

Description
Lattice vectors.

Unit
bohr

Shape
[3, ldim]

Num Int Params%xyzatm

Type
float_array

Description
Atomic coordinates.

Unit
bohr

Shape
[3, nnucs]

NumericalBasisSets

Section content: Specification of numerical atomic basis sets, consisting of a numerical radial table and a spherical harmonic: R_{nl} Y_{lm}.

NumericalBasisSets%BasisType(#{set},#{type})

Type
string

Description
Something like valence or core for (type,set). Will not depend on type.

NumericalBasisSets%bField for GIAO(#{set},#{type})

Type
float_array

Description
Band only. Finite magnetic field strength for GIAOs.

Shape
[3]

NumericalBasisSets%d2RadialFuncs(#{set},#{type})

Type
float_array

Description
The second derivative of the radial functions (for a type,set).

Shape
[NumRad(#{type}), nRadialFuncs(#{set},#{type})]

NumericalBasisSets%dRadialFuncs(#{set},#{type})

Type
float_array

Description
The derivative of the radial functions (for a type,set).

Shape
[NumRad(#{type}), nRadialFuncs(#{set},#{type})]

NumericalBasisSets%Element(#{type})

Type
string

Description
The chemical element (H,He,Li) for a type.

NumericalBasisSets%GridType(#{type})

Type
string

Description
What kind of radial grid is used. Currently this is always logarithmic.

NumericalBasisSets%ljValues(#{set},#{type})

Type
int_array

Description
Normally for each radial function the l value. In case of spin-orbit there is also a j value (for a type,set).

Shape
[2, nRadialFuncs(#{set},#{type})]

NumericalBasisSets%MaxRad(#{type})

Type
float

Description
Maximum value of the radial grid (for a type).

NumericalBasisSets%MinRad(#{type})

Type
float

Description
Minimum value of the radial grid (for a type).

NumericalBasisSets%nRadialFuncs(#{set},#{type})

Type
int

Description
The number of radial functions (for a type,set).

NumericalBasisSets%nSets

Type
int

Description
The number of basis sets stored for each type. For instance if you store core and the valence basis sets it is two.

NumericalBasisSets%nTypes

Type
int

Description
The number of types: elements with a different basis set. Normally this is just the number of distinct elements in the system.

NumericalBasisSets%NumRad(#{type})

Type
int

Description
The number of radial points (for a type).

NumericalBasisSets%RadialFuncs(#{set},#{type})

Type
float_array

Description
The radial functions (for a type,set).

Shape
[NumRad(#{type}), nRadialFuncs(#{set},#{type})]

NumericalBasisSets%RadialMetaInfo(#{set},#{type})

Type
float_array

Description
Info about the radial functions. Whether it is a NAO or STO. For instance for an STO the alpha value. All encoded in a real array of fixed size.

Shape
[:, nRadialFuncs(#{set},#{type})]

NumericalBasisSets%SpherHarmonicType(#{set},#{type})

Type
string

Description
Either zlm or spinor (type,set). Will not depend on type.

NumiType
Section content: Information related to numerical integration.

NumiType%accint

Type
float

Description
Accuracy parameter for the (obsolete) Voronoi grid.

NumiType%bzvol

Type
float

Description
Volume of the Brillouin zone. In lower than 3D this is a surface (2D) or length (1D). For molecules it is set to one.

NumiType%kgrp

Type
int

Description
Number of k-points processed together.

NumiType%kinteg

Type
int

Description
K-space parameter used in case of the symmetric grid.

NumiType%kmesh

Type
int

Description
Parameter for the hybrid method. The quadratic bands are integrated linearly on a grid that is finer by a factor kmesh. (seems obsolete)

NumiType%kt

Type
int

Description
Number of k-points.

NumiType%kuniqu

Type
int

Description
Number of unique k-points.

NumiType%nblock

Type
int

Description
Number of integration blocks.

NumiType%npx

Type
int

Description
Block size.

NumiType%npxsym

Type
int

Description
Obsolete.

NumiType%nrx

Type
int

Description
Maximum number of radial points (see the radial section).

NumiType%nsimpl

Type
int

Description
Number of simplices used for the irreducible Brillouin zone.

NumiType%nuelst

Type
int

Description
Parameter for elliptical grid used to calculate the electrostatic interaction between spherical densities.

NumiType%nvelst

Type
int

Description
Parameter for elliptical grid used to calculate the electrostatic interaction between spherical densities.

NumiType%nvertk

Type
int

Description
Number of vertices per simplex (k-space integration).

NumiType%Serializer::type

Type
string_fixed_length

Description
Information for the Serializer code.

NumiType%volnum

Type
float

Description
Sum of the weight of the real space grid. This only converges with better grids for 3d systems. Otherwise is simply grows with grid size.

OccuType

Section content: Occupation related info.

`OccuType%chbnds`

Type
bool

Description
Technical option about skipping bands.

`OccuType%dfermi`

Type
float

Description
Uncertainty in the fermi energy.

Unit
hartree

`OccuType%edegen`

Type
float

Description
When orbitals are degenerate (within edegen) their occupations are made equal.

Unit
hartree

`OccuType%edosmx`

Type
float

Description
Maximum energy to be used for the dos.

Unit
hartree

`OccuType%efermi`

Type
float

Description
Fermi energy.

Unit
hartree

`OccuType%esprd`

Type
float

Description
ktBoltz: electronic temperature to be used.

Unit
hartree

OccuType%iopdos

Type
int

Description
Whether to use volume (iopdos=1) or a surface integral (iopdos=0).

OccuType%klnear

Type
bool

Description
Technical option.

OccuType%lfntmp

Type
bool

Description
Technical option.

OccuType%nedos

Type
int

Description
Number of energies at which the DOS is sampled.

OccuType%nfdirc

Type
int

Description
Fermi is sampled at several energies at once, and (weight) averaged over them. The weights are in the DOS section.

OccuType%nfdrcx

Type
int

Description
Maximum for nfdirc .

OccuType%nordr

Type
int

Description
Order of the interpolation of the bands: no interpolation(nordr=0), linear(nordr=1), quadratic(nordr=2).

OccuType%Serializer::type

Type
string_fixed_length

Description
Information for the Serializer code.

OccuType%tboltz

Type
float

Description
Temperature Used for the weights efdirc to average the occupations over some energies near the fermi energy.

Unit
hartree

PEDA

Section content: Periodic energy decomposition analysis.

PEDA%AdditionalZEROXC

Type
float

Description
Extra term needed for meta gga's, otherwise XC terms would go wrong.

Unit
hartree

PEDA%EpartPre

Type
float_array

Description
Contributions to the preparation energy.1: kinetic, 2: coulomb, 3: xc.

Unit
hartree

Shape
[3]

PEDA%EpartZero

Type
float_array

Description
Contributions to the energy of psi_0. 1: kinetic, 2: coulomb, 3: xc.

Unit
hartree

Shape
[3]

PEDA%FragmentBondEnergy

Type
float_array

Description
Bond energy of the fragments.

Unit
hartree

Shape
[fragment%nfrag]

PEDA%FragmentDispersion

Type
float_array

Description
Empirical dispersion energy of the fragments.

Unit
hartree

Shape
[fragment%nfrag]

PEDA%FragmentElstat

Type
float_array

Description
Electrostatic energy of the fragments.

Unit
hartree

Shape
[fragment%nfrag]

PEDA%FragmentKinetic

Type
float_array

Description
Kinetic energy of the fragments.

Unit
hartree

Shape
[fragment%nfrag]

PEDA%FragmentMadelung

Type
float_array

Description
Madelung energy of the fragments (almost always zero).

Unit
hartree

Shape
[fragment%nfrag]

PEDA%FragmentXC

Type
float_array

Description
XC energy of the fragments.

Unit
hartree

Shape
[fragment%nfrag]

`PEDA%IsGrimme`

Type
bool_array

Description
Whether empirical dispersion is used in the fragments and in the final system.

Shape
[fragment%nfrag+1]

PEDA bond energy terms
Section content: PEDA bond energy terms.

`PEDA bond energy terms%Dispersion`

Type
float

Description
Dispersion contribution to the interaction energy.

Unit
hartree

`PEDA bond energy terms%E^0`

Type
float

Description
The energy of psi_0. This is the sum of the PauliRepulsion and Electrostatic terms, and also T^0+Elst^0+XC^0.

Unit
hartree

`PEDA bond energy terms%E_orb`

Type
float

Description
The orbital interaction energy. It is written as E_orb=T_orb+Elst_orb+XC_orb

Unit
hartree

`PEDA bond energy terms%Electrostatic`

Type
float

Description
Electrostatic contribution to the interaction energy.

Unit
hartree

PEDA bond energy terms%Elst^0

Type
float

Description
Electrostatic energy contribution to E^0.

Unit
hartree

PEDA bond energy terms%Elst_orb

Type
float

Description
Electrostatic energy contribution to E_orb.

Unit
hartree

PEDA bond energy terms%OrbitalInteraction

Type
float

Description
The orbital interaction contribution to the interaction energy.

Unit
hartree

PEDA bond energy terms%PauliRepulsion

Type
float

Description
Pauli contribution to the interaction energy.

Unit
hartree

PEDA bond energy terms%T^0

Type
float

Description
Kinetic energy contribution to E^0.

Unit
hartree

PEDA bond energy terms%T_orb

Type
float

Description
Kinetic energy contribution to E_orb.

Unit
hartree

PEDA bond energy terms%TotalInteraction

Type
float

Description
The total energy with respect to the fragments. It is the sum of PauliRepulsion+Electrostatic+OrbitalInteraction, and possibly Dispersion.

Unit
hartree

PEDA bond energy terms%XC^0

Type
float

Description
XC energy contribution to E^0.

Unit
hartree

PEDA bond energy terms%XC_orb

Type
float

Description
XC energy contribution to E_orb.

Unit
hartree

PEDANOCV

Section content: ?

PEDANOCV%EigNOCV

Type
float_array

Description
The NOCV eigen vectors.

Shape
[SystType%nbas, SystType%nspin, kspace%kuniqu]

PEDANOCV%ENOCV

Type
float_array

Description
?

Shape
[SystType%nbas+1, SystType%nspin, kspace%kuniqu]

PEDANOCV%nNOCV

Type
int_array

Description
Index of the NOCVs.

Shape
[SystType%nspin, kspace%kuniqu]

PEDANOCV%TNOCV

Type
float_array

Description
?

Shape
[SystType%nbas, SystType%nspin, kspace%kuniqu]

PEDANOPR

Section content: ?

PEDANOPR%EigNOCV

Type
float_array

Description
The NOCV eigen vectors.

Shape
[SystType%nbas, SystType%nspin, kspace%kuniqu]

PEDANOPR%ENOCV

Type
float_array

Description
?

Shape
[SystType%nbas+1, SystType%nspin, kspace%kuniqu]

PEDANOPR%nNOCV

Type
int_array

Description
Index of the NOCVs.

Shape
[SystType%nspin, kspace%kuniqu]

PEDANOPR%TNOCV

Type
float_array

Description
?

Shape
[SystType%nbas, SystType%nspin, kspace%kuniqu]

PeriodicZlmFit
Section content: ZlmFit info related to periodic systems.

`PeriodicZlmFit%aVec`

Type
float_array

Description
The lattice stored as a 3xnLatticeVectors matrix. Only the ndimk,ndimk part has meaning.

Unit
bohr

Shape
[kspace%ndim, kspace%ndim]

`PeriodicZlmFit%cellCenter`

Type
float_array

Description
Origin of the cell.

Shape
[3]

`PeriodicZlmFit%cellMultipoles`

Type
float_array

Description
Multipole coefficients for the cells. Dimension: self%zlmFit%lMax+1)**2

Shape
[:]

`PeriodicZlmFit%fGaussianW`

Type
float

Description
Width for the gaussians around atom centers. (Only 3D)

`PeriodicZlmFit%fGridSpacing`

Type
float

Description
Spacing for the fourier grid. Only used for 3D periodic systems.

`PeriodicZlmFit%firstTopoCell`

Type
int

Description
First cell from whereon the fitting of the topological extrapolation is started.

PeriodicZlmFit%fKSpaceCutoff

Type
float

Description
Cutoff criterion used in k-space. (Only 3D)

PeriodicZlmFit%fMultipoleCoeff

Type
float_array

Description
Atomic multipole cofs.

Shape
[nAtoms, nMultipolesFourier]

PeriodicZlmFit%lastTopoCell

Type
int

Description
Last cell from whereon the fitting of topological extrapolation is started. For cells>lastTopoCell the extrapolation is used.

PeriodicZlmFit%nAtoms

Type
int

Description
Number of atoms.

PeriodicZlmFit%nDim

Type
int

Description
Numer of lattice vectors. (Same as geometry%ndim)

PeriodicZlmFit%neutralizingCharges

Type
float_array

Description
?

PeriodicZlmFit%nMultipolesFourier

Type
int

Description
Number of multipoles used for the Fourier expansion. Only used for 3D periodic systems.

PeriodicZlmFit%nNeutralizingCharges

Type
int

Description
?

PeriodicZlmFit%orderTopoCells

Type
int

Description
Order used to generate topological cells.

PeriodicZlmFit%orderTopoTrick

Type
int

Description
Order for the topological extrapolation.

PeriodicZlmFit%Ren.ChargeMethod

Type
int

Description
Method used to enforce charge neutrality. Either 1 or 2.

PeriodicZlmFit%TotalCharge

Type
float

Description
?

PeriodicZlmFit%xyzNeutralizingCharges

Type
float_array

Description
?

PeriodicZlmFit(pot)
Section content: ZlmFit info related to periodic systems.

PeriodicZlmFit(pot)%aVec

Type
float_array

Description
The lattice stored as a 3xnLatticeVectors matrix. Only the ndimk,ndimk part has meaning.

Unit
bohr

Shape
[kspace%ndim, kspace%ndim]

PeriodicZlmFit(pot)%cellCenter

Type
float_array

Description
Origin of the cell.

Shape
[3]

PeriodicZlmFit(pot)%cellMultipoles

Type
float_array

Description
Multipole coefficients for the cells. Dimension: self%zlmFit%lMax+1)**2

Shape
[:]

PeriodicZlmFit(pot)%fGaussianW

Type
float

Description
Width for the gaussians around atom centers. (Only 3D)

PeriodicZlmFit(pot)%fGridSpacing

Type
float

Description
Spacing for the fourier grid. Only used for 3D periodic systems.

PeriodicZlmFit(pot)%firstTopoCell

Type
int

Description
First cell from whereon the fitting of the topological extrapolation is started.

PeriodicZlmFit(pot)%fKSpaceCutoff

Type
float

Description
Cutoff criterion used in k-space. (Only 3D)

PeriodicZlmFit(pot)%fMultipoleCoeff

Type
float_array

Description
Atomic multipole cofs.

Shape
[nAtoms, nMultipolesFourier]

PeriodicZlmFit(pot)%lastTopoCell

Type
int

Description
Last cell from whereon the fitting of topological extrapolation is started. For cells>lastTopoCell the extrapolation is used.

PeriodicZlmFit(pot)%nAtoms

Type
int

Description
Number of atoms.

PeriodicZlmFit(pot)%nDim

Type
int

Description
Numer of lattice vectors. (Same as geometry%ndim)

PeriodicZlmFit(pot)%neutralizingCharges

Type
float_array

Description
?

PeriodicZlmFit(pot)%nMultipolesFourier

Type
int

Description
Number of multipoles used for the Fourier expansion. Only used for 3D periodic systems.

PeriodicZlmFit(pot)%nNeutralizingCharges

Type
int

Description
?

PeriodicZlmFit(pot)%orderTopoCells

Type
int

Description
Order used to generate topological cells.

PeriodicZlmFit(pot)%orderTopoTrick

Type
int

Description
Order for the topological extrapolation.

PeriodicZlmFit(pot)%Ren.ChargeMethod

Type
int

Description
Method used to enforce charge neutrality. Either 1 or 2.

PeriodicZlmFit(pot)%TotalCharge

Type
float

Description
?

PeriodicZlmFit(pot)%xyzNeutralizingCharges

Type
float_array

Description
?

PeriodicZlmFit(pot)-ZlmFit

Section content: General zlm fit info.

PeriodicZlmFit(pot)-ZlmFit%densityThresh

Type
float_array

Description
Threshold for the density.

Shape
[nAtoms]

PeriodicZlmFit(pot)-ZlmFit%lMax

Type
int

Description
Number of atoms.

PeriodicZlmFit(pot)-ZlmFit%lMaxExpansion

Type
int_array

Description
Maximum l-value for the fit functions per atom.

Shape
[nAtoms]

PeriodicZlmFit(pot)-ZlmFit%maxNPointsRadGrid

Type
int

Description
?.

PeriodicZlmFit(pot)-ZlmFit%nAtoms

Type
int

Description
Number of atoms.

PeriodicZlmFit(pot)-ZlmFit%nRadialPoints

Type
int_array

Description
Number of radial points per atom.

Shape
[nAtoms]

PeriodicZlmFit(pot)-ZlmFit%nSpin

Type
int

Description
Number of spin components.

Possible values
[1, 2]

PeriodicZlmFit(pot)-ZlmFit%potentialThresh

Type
float_array

Description
Threshold for the potential.

Unit
a.u.

Shape
[nAtoms]

PeriodicZlmFit(pot)-ZlmFit%projCoeff

Type
float_array

Description
Projection coefficients.

Shape
[sizeProjCoeff]

PeriodicZlmFit(pot)-ZlmFit%pruning

Type
bool

Description
Whether or not to prune.

PeriodicZlmFit(pot)-ZlmFit%pruningL

Type
int

Description
?.

PeriodicZlmFit(pot)-ZlmFit%pruningThreshDist

Type
float

Description
Distance threshold for pruning.

Unit
bohr

PeriodicZlmFit(pot)-ZlmFit%radialGrid

Type
float_array

Description
Radial grids per atom.

Shape
[nAtoms, maxNPointsRadGrid]

PeriodicZlmFit(pot)-ZlmFit%sizeProjCoeff

Type
int

Description
?.

PeriodicZlmFit(pot)-ZlmFit%xyzAtoms

Type
float_array

Description
Atom coordinates.

Unit
bohr

Shape
[3, nAtoms]

PeriodicZlmFit-ZlmFit
Section content: General zlm fit info.

PeriodicZlmFit-ZlmFit%densityThresh

Type
float_array

Description
Threshold for the density.

Shape
[nAtoms]

PeriodicZlmFit-ZlmFit%lMax

Type
int

Description
Number of atoms.

PeriodicZlmFit-ZlmFit%lMaxExpansion

Type
int_array

Description
Maximum l-value for the fit functions per atom.

Shape
[nAtoms]

PeriodicZlmFit-ZlmFit%maxNPointsRadGrid

Type
int

Description
?.

PeriodicZlmFit-ZlmFit%nAtoms

Type
int

Description
Number of atoms.

PeriodicZlmFit-ZlmFit%nRadialPoints

Type
int_array

Description
Number of radial points per atom.

Shape
[nAtoms]

PeriodicZlmFit-ZlmFit%nSpin

Type
int

Description
Number of spin components.

Possible values
[1, 2]

PeriodicZlmFit-ZlmFit%potentialThresh

Type
float_array

Description
Threshold for the potential.

Unit
a.u.

Shape
[nAtoms]

PeriodicZlmFit-ZlmFit%projCoeff

Type
float_array

Description
Projection coefficients.

Shape
[sizeProjCoeff]

PeriodicZlmFit-ZlmFit%pruning

Type
bool

Description
Whether or not to prune.

PeriodicZlmFit-ZlmFit%pruningL

Type
int

Description
?.

PeriodicZlmFit-ZlmFit%pruningThreshDist

Type
float

Description
Distance threshold for pruning.

Unit
bohr

PeriodicZlmFit-ZlmFit%radialGrid

Type
float_array

Description
Radial grids per atom.

Shape
[nAtoms, maxNPointsRadGrid]

PeriodicZlmFit-ZlmFit%sizeProjCoeff

Type
int

Description
?.

PeriodicZlmFit-ZlmFit%xyzAtoms

Type
float_array

Description
Atom coordinates.

Unit
bohr

Shape
[3, nAtoms]

phonon_curves
Section content: Phonon dispersion curves.

phonon_curves%brav_type

Type
string

Description
Type of the lattice.

phonon_curves%Edge_#_bands

Type
float_array

Description
The band energies

Shape
[nBands, nSpin, :]

phonon_curves%Edge_#_direction

Type
float_array

Description
Direction vector.

Shape
[nDimK]

phonon_curves%Edge_#_kPoints

Type
float_array

Description
Coordinates for points along the edge.

Shape
[nDimK, :]

phonon_curves%Edge_#_labels

Type
lchar_string_array

Description
Labels for begin and end point of the edge.

Shape
[2]

phonon_curves%Edge_#_lGamma

Type
bool

Description
Is gamma point?

phonon_curves%Edge_#_nKPoints

Type
int

Description
The nr. of k points along the edge.

phonon_curves%Edge_#_vertices

Type
float_array

Description
Begin and end point of the edge.

Shape
[nDimK, 2]

phonon_curves%Edge_#_xFor1DPlotting

Type
float_array

Description
x Coordinate for points along the edge.

Shape
[:]

phonon_curves%indexLowestBand

Type
int

Description
?

phonon_curves%nBands

Type
int

Description
Number of bands.

phonon_curves%nBas

Type
int

Description
Number of basis functions.

phonon_curves%nDimK

Type
int

Description
Dimension of the reciprocal space.

phonon_curves%nEdges

Type
int

Description
The number of edges. An edge is a line-segment through k-space. It has a begin and end point and possibly points in between.

phonon_curves%nEdgesInPath

Type
int

Description
A path is built up from a number of edges.

phonon_curves%nSpin

Type
int

Description
Number of spin components.

Possible values
[1, 2]

phonon_curves%path

Type
int_array

Description
If the (edge) index is negative it means that the vertices of the edge abs(index) are swapped e.g. path = (1,2,3,0,-3,-2,-1) goes though edges 1,2,3, then there's a jump, and then it goes back.

Shape
[nEdgesInPath]

phonon_curves%path_source

Type
string

Description
Source or program used to generate the path.

Possible values
['input', 'kpath', 'seekpath']

phonon_curves%path_type

Type
string

Description
?

Phonons

Section content: Information on the numerical phonons (super cell) setup. NB: the reciprocal cell of the super cell is smaller than the reciprocal primitive cell.

Phonons%Modes

Type
float_array

Description
The normal modes with the translational symmetry of the super cell.

Shape
[3, nAtoms, 3, NumAtomsPrim, nK]

Phonons%nAtoms

Type
int

Description
Number of atoms in the super cell.

Phonons%nK

Type
int

Description
Number of gamma-points (of the super cell) that fit into the primitive reciprocal cell.

Phonons%NumAtomsPrim

Type
int

Description
Number of atoms in the primitive cell.

Phonons%xyzKSuper

Type
float_array

Description
The coordinates of the gamma points that fit into the primitive reciprocal cell.

Shape
[3, nK]

Plot

Section content: Generic section to store x-y plots.

Plot%numPlots

Type
int

Description
Number of plots.

Plot%NumPoints(#)

Type
int

Description
Number of x points for plot #.

Plot%NumYSeries(#)

Type
int

Description
Number of y series for plot #.

Plot%Title(#)

Type
string

Description
Title of plot #

`Plot%XLabel(#)`

Type
string

Description
X label for plot #.

`Plot%XUnit(#)`

Type
string

Description
X unit for plot #.

`Plot%XValues(#)`

Type
float_array

Description
X values for plot #.

Shape
[:]

`Plot%YLabel(#)`

Type
string

Description
Y label for plot #.

`Plot%YUnit(#)`

Type
string

Description
Y unit for plot #.

`Plot%YValues(#)`

Type
float_array

Description
Y values for plot #. Array has extra column NumYSeries.

PrecType

Section content: Precision related info.

`PrecType%cutoff`

Type
float

Description
cutoff criterion.

PrecType%dmadel

Type
float

Description
Decay width parameter for the Madelung screening function.

PrecType%fermfc

Type
float

Description
Another parameter for the Madelung screening.

PrecType%ldscrm

Type
bool

Description
Use directional screening. Only relevant when Madelung screening is used: old STO fit and COSMO.

PrecType%ncel

Type
int

Description
Number of cells.

PrecType%rcelx

Type
float

Description
Largest distance of all the cells considered.

PrecType%rfar

Type
float

Description
Maximum extension of radial functions. Not used.

PrecType%rmadel

Type
float

Description
Distance parameter for the Madelung screening function.

PrecType%scrcor

Type
float

Description
Dependency%Core input parameter. (core/core overlap)

PrecType%scrcv

Type
float

Description
Dependency%CoreValence input parameter.

PrecType%scrfit

Type
float

Description
Dependency%Fit input parameter.

PrecType%scrval

Type
float

Description
Dependency%Basis input parameter.

PrecType%Serializer::type

Type
string_fixed_length

Description
Information for the Serializer code.

Properties

Section content: Property section.

Properties%BP atoms

Type
int_array

Description
?

Properties%BP number of

Type
int

Description
Number of bond paths (QTAIM).

Properties%BP shift

Type
float_array

Description
(lattice) shifts for start and end atoms

Shape
[3, BP number of, 2]

Properties%BP step number

Type
int_array

Description
Number of steps in the path (QTAIM).

Shape
[BP number of]

Properties%BPs and their properties

Type
float_array

Description
?

Shape
[13, :, BP number of]

Properties%CP code number for (Rank,Signatu

Type
float_array

Description
Characterization of the critical points. 1: (3,-3) atom critical point. 2: (3,+3) cage critical point. 3: (3,-1) bond critical point. 4: (3,+1) ring critical point.

Shape
[CP number of]

Properties%CP coordinates

Type
float_array

Description
Coordinates of the critical points (QTAIM).

Shape
[3, CP number of]

Properties%CP density at

Type
float_array

Description
Density at the critical points (QTAIM).

Shape
[CP number of]

Properties%CP density gradient at

Type
float_array

Description
Gradient of the density at the critical points (QTAIM).

Shape
[3, CP number of]

Properties%CP density Hessian at

Type
float_array

Description
Hessian of the density at the critical points (QTAIM). Compressed symmetric storage.

Shape
[6, CP number of]

Properties%CP number of

Type
int

Description
Number of critical points for the density (QTAIM).

Properties%nEntries

Type
int

Description
Number of properties.

Properties%Subtype(#)

Type
string_fixed_length

Description
Extra detail about the property. For a charge property this could be Mulliken.

Properties%Type(#)

Type
string

Description
Type of the property, like energy, gradients, charges, etc.

Properties%Value(#)

Type
float_array

Description
The value(s) of the property.

radial

Section content: A system can be built up from fragments, allowing an energy decomposition. The bonding energy will be with respect to the fragments.

radial%cutoff

Type
float

Description
Cutt off criterion for the radial tables.

radial%ncelforscreening

Type
int

Description
Number of cells needed for screening (obsolete).

radial%ncore

Type
int_array

Description
Number of radial core functions per type.

Shape
[geometry%ntyp]

radial%nfit

Type
int_array

Description
Number of radial core functions per type.

Shape
[geometry%ntyp]

radial%nr

Type
int_array

Description
Number of radial points per type.

Shape
[geometry%ntyp]

radial%nrx

Type
int

Description
Maximum number of radial points for any type.

radial%rad

Type
float_array

Description
Maximum number of radial points for any type.

Shape
[nrx, geometry%ntyp]

radial%xyzcelforscreening

Type
float_array

Description
Cell coordinates needed for screening (obsolete).

Shape
[3, ncelforscreening]

radial tables type#
Section content: Information about the eigensystem.

`radial tables type#%alj`

Type
string

Description
Labels for the valence orbitals with each label having a length of six characters.

`radial tables type#%d2rho`

Type
float_array

Description
Second derivative of the density.

Shape
[nRadialPoints]

`radial tables type#%dfxc`

Type
float_array

Description
Derivative of the XC energy density.

Shape
[nRadialPoints]

`radial tables type#%drho`

Type
float_array

Description
Radial derivative of density.

Shape
[nRadialPoints]

`radial tables type#%drho(valence)`

Type
float_array

Description
Radial derivative of valence density.

Shape
[nRadialPoints]

`radial tables type#%dvcoul`

Type
float_array

Description
Derivative of the Coulomb potential.

Shape
[nRadialPoints]

radial tables type#%ecor

Type
float_array

Description
Some hardcoded X+C energies, like Becke88X+Perdew86c.

Unit
hartree

Shape
[14]

radial tables type#%ekin(valence)

Type
float

Description
Kinetic energy due to the sum of valence NAOs.

Unit
hartree

radial tables type#%eorb

Type
float_array

Description
Valence orbital energies. The number of spin components is usually 1, unless one uses unrestricted reference.

Shape
[nao, :]

radial tables type#%etotal

Type
float_array

Description
Total energy for 14 hardcoded functionals.

Unit
hartree

Shape
[14]

radial tables type#%excterms

Type
float_array

Description
Some hardcoded xc terms, like Becke88X.

Unit
hartree

Shape
[20]

radial tables type#%fxc

Type
float_array

Description
XC energy density.

Shape
[nRadialPoints]

radial tables type#%gradPot

Type
float_array

Description
Derivative of the KS potential.

Shape
[nRadialPoints]

radial tables type#%gradRho(core)

Type
float_array

Description
Radial derivative of the core density.

Shape
[nRadialPoints]

radial tables type#%gradRho(valence)

Type
float_array

Description
Radial derivative of the valence density.

Shape
[nRadialPoints]

radial tables type#%gradTau(core)

Type
float_array

Description
Gradient of core contribution to tau.

Shape
[nRadialPoints]

radial tables type#%nao

Type
int

Description
Number of numerical atomic orbitals (solutions for a spherical atom).

radial tables type#%ncell

Type
int

Description
Number of cells.

radial tables type#%ncore

Type
int

Description
Number of core orbitals.

radial tables type#%neutralizing density

Type
float_array

Description
May be used for atomic based neutralizing densities (charged cells).

Shape
[nRadialPoints]

radial tables type#%nfit

Type
int

Description
Number of fit functions.

radial tables type#%nRadialPoints

Type
int

Description
Number of radial points.

radial tables type#%nspna

Type
int

Description
Spin multiplicity for spherical atoms, in practice this is always 1.

radial tables type#%qeff

Type
float

Description
Nett charge of the atom, usually zero.

radial tables type#%qnao

Type
float_array

Description
Occupations of the valence NAOs.

Shape
[nao]

radial tables type#%qnuclr

Type
float

Description
Charge of the nucleus (almost always the atomic number).

radial tables type#%qval

Type
float

Description
Total valence occupation.

radial tables type#%radius of most diffuse NAO

Type
float

Description
Radius of the most diffuse NAO.

Unit
bohr

radial tables type#%rho

Type
float_array

Description
Density.

Shape
[nRadialPoints]

radial tables type#%rho(valence)

Type
float_array

Description
Valence density.

Shape
[nRadialPoints]

radial tables type#%secDerRho(core)

Type
float_array

Description
Second derivative of the core density.

Shape
[nRadialPoints]

radial tables type#%secDerRho(valence)

Type
float_array

Description
Second derivative of the valence density.

Shape
[nRadialPoints]

radial tables type#%tau

Type
float_array

Description
Tau =0.5*(dpsi)**2.

Shape
[nRadialPoints]

radial tables type#%tau(core)

Type
float_array

Description
Core contribution to tau.

Shape
[nRadialPoints]

radial tables type#%tau(valence)

Type
float_array

Description
Valence contribution to tau.

Shape
[nRadialPoints]

radial tables type#%tauAsymVal

Type
float_array

Description
Asymmetric valence tau (=0.5 psi d**2 psi).

Shape
[nRadialPoints]

radial tables type#%totalPotential

Type
float_array

Description
Total KS potential.

Shape
[nRadialPoints]

radial tables type#%valkin

Type
float_array

Description
Valence kinetic energy density.

Shape
[nRadialPoints]

`radial tables type#%vcoul`

Type
float_array

Description
Coulomb potential.

Shape
[nRadialPoints]

`radial tables type#%vxc`

Type
float_array

Description
XC potential.

Shape
[nRadialPoints]

RadialAtomicFunctions
Section content: Info regarding spherical atom centered functions.

`RadialAtomicFunctions%d2RadialFunc(#{func},#{type})`

Type
float_array

Description
Second derivative of the radial function.

Shape
[NumericalBasisSets%NumRad(#{type})]

`RadialAtomicFunctions%dRadialFunc(#{func},#{type})`

Type
float_array

Description
Derivative of the radial function.

Shape
[NumericalBasisSets%NumRad(#{type})]

`RadialAtomicFunctions%FunctionType(#{func},#{type})`

Type
string

Description
FunctionType(a,b) gives the name of function a for type b. It could have a value like core density.

RadialAtomicFunctions%nFunctions

Type
int

Description
The number of radial functions stored for each type. For instance if you store the core and the valence density it is two.

RadialAtomicFunctions%nTypes

Type
int

Description
The number of types: elements with a different basis set. Normally this is just the number of distinct elements in the system.

RadialAtomicFunctions%RadialFunc(#{func},#{type})

Type
float_array

Description
RadialFunc(a,b) gives the radial table for function a for type b

Shape
[NumericalBasisSets%NumRad(#{type})]

response
Section content: Old response.

response%nfrad

Type
int_array

Description
Number of radial functions per type.

Shape
[geometry%ntyp]

response%xyzcel

Type
float_array

Description
Cell coordinates.

Shape
[3, PrecType%ncel]

Response TD-CDFT
Section content: Results of te response calculation.

Response TD-CDFT%ActiveXYZ

Type
bool_array

Description
Whether the cartesian components are active.

Shape
[3]

Response TD-CDFT%Converged

Type
bool_array

Description
Whether the response-SCF converged for all frequencies.

Shape
[Number of Frequencies]

Response TD-CDFT%DielecFunc - Imag Part

Type
float_array

Description
Imaginary part of the dielectric function.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%DielecFunc - Real Part

Type
float_array

Description
Real part of dielectric function.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%DielecFunc-boots - Imag Part

Type
float_array

Description
Imaginary part of the dielectric function.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%DielecFunc-boots - Real Part

Type
float_array

Description
Real part of dielectric function.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%Frequencies in au

Type
float_array

Description
Frequencies.

Shape
[Number of Frequencies]

Response TD-CDFT%Number of Frequencies

Type
int

Description
Number of frequencies.

Response TD-CDFT%Polarisabi - Imag Part

Type
float_array

Description
Imaginary part of the Polarizability.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%Polarisabi - Real Part

Type
float_array

Description
Real part of Polarizability.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%Polarisabi-boots - Imag Part

Type
float_array

Description
Imaginary part of the Polarizability.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%Polarisabi-boots - Real Part

Type
float_array

Description
Real part of Polarizability.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%RefraIndex - Imag Part

Type
float_array

Description
Imaginary part of the refractive index.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%RefraIndex - Real Part

Type
float_array

Description
Real part of refractive index.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%RefraIndex-boots - Imag Part

Type
float_array

Description
Imaginary part of the refractive index.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%RefraIndex-boots - Real Part

Type
float_array

Description
Real part of refractive index.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%Susceptibi - Imag Part

Type
float_array

Description
Imaginary part of the susceptibility.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%Susceptibi - Real Part

Type
float_array

Description
Real part of the susceptibility.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%Susceptibi-boots - Imag Part

Type
float_array

Description
Imaginary part of the susceptibility.

Shape
[3, 3, Number of Frequencies]

Response TD-CDFT%Susceptibi-boots - Real Part

Type
float_array

Description
Real part of the susceptibility.

Shape
[3, 3, Number of Frequencies]

ResponseEigenSystem
Section content: Store eignesystem in another way.

ResponseEigenSystem%Data

Type
float_array

Description
Outer loop over kun, then spin. eigenvalues are written and real part of eigsys followed by the imag part.

ResponseEigenSystem%EigenValues

Type
float_array

Description
Just the eigenvalues. Like in data but without the eigenvectors.

ResponseRestartInfo
Section content: Info for restarting new response.

ResponseRestartInfo%IsActiveXYZ

Type
bool_array

Description
Whether the cartesian components are active.

Shape
[3]

ResponseRestartInfo%LowFreqAlgo

Type
bool

Description
Whether to use the low frequency algorithm.

ResponseRestartInfo%nBand

Type
int

Description
Number of bands.

ResponseRestartInfo%nBas

Type
int

Description
Number of basis functions.

ResponseRestartInfo%nFreq

Type
int

Description
Number of frequencies.

ResponseRestartInfo%nFul

Type
int

Description
Number of fully occupied bands.

ResponseRestartInfo%nOcc

Type
int

Description
Number of bands with an occupation > 0.

ResponseRestartInfo%nPar

Type
int

Description
Number of partially occupied bands.

ResponseRestartInfo%nSpin

Type
int

Description
Number of spin components.

Possible values
[1, 2]

ResponseRestartInfo%nVir

Type
int

Description
Number of virtual (unoccupied) bands.

RPA energies

Section content: ?

RPA energies%Direct RPA correlation

Type
float

Description
?

RPA energies%GM Delta P correlation

Type
float_array

Description
?

`RPA energies%RPA correlation`

Type
float

Description
?

`RPA energies%RPA exchange`

Type
float

Description
?

`RPA energies%RPA xc`

Type
float_array

Description
?

`RPA energies%SOS-MP2 correlation`

Type
float

Description
?

`RPA energies%SOX`

Type
float

Description
?

Scalar Atomic Properties

Section content: Scalar numbers per atom.

`Scalar Atomic Properties%*`

Type
float_array

Description
.

Shape
[Molecule%nAtoms]

`Scalar Atomic Properties%Number of properties`

Type
int

Description
Number of properties.

`Scalar Atomic Properties%Property names`

Type
lchar_string_array

Description
names of the properties.

Shape
[Number of properties]

SCCLogger
Section content: Information on the progress of the SCF procedure.

`SCCLogger%coefficients(#)`

Type
float_array

Description
?

`SCCLogger%Converged`

Type
bool

Description
?

`SCCLogger%Criterion`

Type
float

Description
?

`SCCLogger%currentEntryOpen`

Type
bool

Description
Information might be incomplete when this is True.

`SCCLogger%error(#{cycle})`

Type
float

Description
Self Consistent error in the density.

`SCCLogger%ItemName(#)`

Type
string

Description
Allowed names per cycle in this section.

Possible values
['mix', 'error', 'nvctr', 'method', 'nVectors', 'iterationNumbers', 'coefficients']

SCCLogger%iterationNumbers(#)

Type
int_array

Description
?

SCCLogger%LastError

Type
float

Description
?

SCCLogger%method(#{cycle})

Type
string

Description
A letter indicating which algorithm was used. m: mixing, d: diis.

SCCLogger%mix(#{cycle})

Type
float

Description
Mixing parameter used.

SCCLogger%nEntries

Type
int

Description
Number of SCF cycles done.

SCCLogger%nFailedSCF

Type
int

Description
?

SCCLogger%nIterations

Type
int

Description
?

SCCLogger%nIterationsGlobal

Type
int

Description
?

SCCLogger%nSCF

Type
int

Description
?

SCCLogger%nvctr(#{cycle})

Type
int

Description
Number of older densities used to guess the next density.

SCCLogger%nVectors(#)

Type
int

Description
?

scf

Section content: Information about the self consistent procedure. Mostly outdated.

scf%degenerate

Type
bool

Description
Degenerate option to make weights for degenerate bands equal.

scf%parmin

Type
float

Description
Minimum for the mixing parameter.

scf%parmix

Type
float

Description
Mixing parameter.

scf%scfrtx

Type
float

Description
.

Symmetry

Section content: Info regarding the symmetry of the system.

Symmetry%iatopr

Type
int_array

Description
Each operator maps an atom to another atom: jAtom=iatopr(iAtom,iOper).

Shape
[Molecule%nAtoms, Nr. of operators]

Symmetry%Inverse Operator Index

Type
int_array

Description
Which operator is the inverse operator?

Shape
[Nr. of operators]

Symmetry%lxsum

Type
int

Description
If lmax is the max l value occurring this is sum_l^lmax (2*l+1)**2: the total nr. of spherical harmonics with l smaller or equal than lmax.

Symmetry%Nr. of operators

Type
int

Description
The number of symmetry operations.

Symmetry%Nr. of operators (k-space)

Type
int

Description
The number of symmetry operations.

Symmetry%operatorAtomShift

Type
float_array

Description
Each operator can move an atom to another cell. Will be a lattice translation.

Shape
[3, Molecule%nAtoms, Nr. of operators]

Symmetry%Operators

Type
float_array

Description
The point group part of the space group operators.

Shape
[3, 3, Nr. of operators]

Symmetry%Operators (k-space)

Type
float_array

Description
The symmetry operators in reciprocal space.

Shape
[:, :, Nr. of operators (k-space)]

Symmetry%oprzlm

Type
float_array

Description
Zlm representation of the operators. An operator working on l,m, makes it into a linear combi of the same l.

Shape
[lxsum, Nr. of operators]

Symmetry%Translations

Type
float_array

Description
The translation part of the space group operators (partial fractional lattice displacements).

Shape
[3, Nr. of operators]

SystType

Section content: Basis set and other aspects of the calculation.

SystType%ioptxc

Type
int

Description
Obscure XC option. 0: x-alpha, 1: vosko-wilk-nusair, 2: vosko-wilk-nusair + stoll correction.

SystType%lcorex

Type
int

Description
Maximum l-value for frozen core functions.

SystType%lfitx

Type
int

Description
Maximum l-value for STO fit functions (if any).

SystType%lvalx

Type
int

Description
Maximum l-value for valence functions.

SystType%lxsum

Type
int

Description
Space required to describe mapping of Zlm's under a symmetry operation. The maximum l is max(lcorex,lfitx,lvalx).

SystType%nband

Type
int

Description
Number of bands to be stored or printed. Smaller or equal to nbas.

SystType%nbas

Type
int

Description
Number of (valence) basis functions.

SystType%ncores

Type
int

Description
Number of frozen core functions.

SystType%ncorex

Type
int

Description
Obsolete, not used. Maximum nr. of radial core functions per type.

SystType%ncorez

Type
int

Description
Number of slater type core functions (obsolete, not used).

SystType%nfitt

Type
int

Description
Number of STO fit functions.

SystType%nfitx

Type
int

Description
Maximum number of radial STO fit functions per type.

SystType%nfrag

Type
int

Description
Number of fragments to be used for the calculation.

SystType%ngross

Type
int

Description
Number of user requested gross populations to print.

SystType%noverl

Type
int

Description
Number of user requested overlap populations to print.

SystType%nspin

Type
int

Description
Number of spin components for the eigenvectors. 1) Spin restricted, 2) Spin unrestricted: eigenvectors can be symmetry labeled up or down.

Possible values
[1, 2]

SystType%nspino

Type
int

Description
Number of spin components for the basis set. 1) Scalar or non relativistic, 2) Spin orbit. With nspino=2 nspin=1, as eigenvectors do not have a up or down symmetry label. The density has max(nspin,nspin) components. In case of noncollinear magnetization it even is a 2x2 complex Hermitian matrix.

Possible values
[1, 2]

SystType%nsymft

Type
int

Description
Number of symmetric STO fit function combinations.

SystType%nvalx

Type
int

Description
Maximum number of radial valence functions per type.

SystType%qelec

Type
float

Description
The number of valence electrons (not including a nett charge of the system).

SystType%Serializer::type

Type
string_fixed_length

Description
Information for the Serializer code.

SystType%vsplit

Type
float

Description
Add vsplit to the spin up potential at the first SCF cycle to break the initial exact spin symmetry.

SystType%xcpar

Type
float

Description
x-alpha parameter.

SystType%zora

Type
bool

Description
Whether to use the relativistic ZORA approximation (zora=yes) or the nonrelativistic one (zora=no).

Thermodynamics

Section content: Thermodynamic properties computed from normal modes.

Thermodynamics%Enthalpy

Type
float_array

Description
Enthalpy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Entropy rotational

Type
float_array

Description
Rotational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Entropy total

Type
float_array

Description
Total entropy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Entropy translational

Type
float_array

Description
Translational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Entropy vibrational

Type
float_array

Description
Vibrational contribution to the entropy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Gibbs free Energy

Type
float_array

Description
Gibbs free energy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Heat Capacity rotational

Type
float_array

Description
Rotational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Heat Capacity total

Type
float_array

Description
Total heat capacity.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Heat Capacity translational

Type
float_array

Description
Translational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Heat Capacity vibrational

Type
float_array

Description
Vibrational contribution to the heat capacity.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Inertia direction vectors

Type
float_array

Description
Inertia direction vectors.

Shape
[3, 3]

Thermodynamics%Internal Energy rotational

Type
float_array

Description
Rotational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Internal Energy total

Type
float_array

Description
Total internal energy.

Unit
a.u.

Thermodynamics%Internal Energy translational

Type
float_array

Description
Translational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%Internal Energy vibrational

Type
float_array

Description
Vibrational contribution to the internal energy.

Unit
a.u.

Shape
[nTemperatures]

Thermodynamics%lowFreqEntropy

Type
float_array

Description
Entropy contributions from low frequencies (see 'lowFrequencies').

Unit
a.u.

Shape
[nLowFrequencies]

Thermodynamics%lowFreqHeatCapacity

Type
float_array

Description
Heat capacity contributions from low frequencies (see 'lowFrequencies').

Unit
a.u.

Shape
[nLowFrequencies]

Thermodynamics%lowFreqInternalEnergy

Type
float_array

Description
Internal energy contributions from low frequencies (see 'lowFrequencies').

Unit
a.u.

Shape
[nLowFrequencies]

Thermodynamics%lowFrequencies

Type
float_array

Description
Frequencies below 20 cm^-1 (contributions from frequencies below 20 cm^-1 are not included in vibrational sums, and are saved separately to 'lowFreqEntropy', 'lowFreqInternalEnergy' and 'lowFreqInternalEnergy'). Note: this does not apply to RRHO-corrected quantities.

Unit
cm^-1

Shape
[nLowFrequencies]

Thermodynamics%Moments of inertia

Type
float_array

Description
Moments of inertia.

Unit
a.u.

Shape
[3]

Thermodynamics%nLowFrequencies

Type
int

Description
Number of elements in the array lowFrequencies.

`Thermodynamics%nTemperatures`

Type
int

Description
Number of temperatures.

`Thermodynamics%Pressure`

Type
float

Description
Pressure used.

Unit
atm

`Thermodynamics%RRHOCorrectedHeatCapacity`

Type
float_array

Description
Heat capacity T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%RRHOCorrectedInternalEnergy`

Type
float_array

Description
Internal energy T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%RRHOCorrectedTS`

Type
float_array

Description
T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%Temperature`

Type
float_array

Description
List of temperatures at which properties are calculated.

Unit
a.u.

Shape
[nTemperatures]

`Thermodynamics%TS`

Type
float_array

Description
T*S, i.e. temperature times entropy.

Unit
a.u.

Shape
[nTemperatures]

TmpGrad

Section content: Energy gradient information in the band internal logic: rotation and atom reordering.

`TmpGrad%energy`

Type
float

Description
Energy.

`TmpGrad%gradients`

Type
float_array

Description
Energy gradients according to the band internal logic: rotated to standard frame and atom reordering. If not calculated it has zero length.

UnitCell(real space)

Section content: Information on the Wigner-Seitz cell.

`UnitCell(real space)%boundaries`

Type
float_array

Description
Normal vectors for the boundaries.

Shape
[ndim, nboundaries]

UnitCell(real space)%distances

Type
float_array

Description
Distance to the boundaries.

Shape
[nboundaries]

UnitCell(real space)%idVerticesPerBound

Type
int_array

Description
The indices of the vertices per bound.

Shape
[nvertices, nboundaries]

UnitCell(real space)%latticeVectors

Type
float_array

Description
The lattice vectors.

Shape
[3, :]

UnitCell(real space)%nboundaries

Type
int

Description
The nr. of boundaries for the cell.

UnitCell(real space)%ndim

Type
int

Description
The nr. of lattice vectors spanning the Wigner-Seitz cell.

UnitCell(real space)%numVerticesPerBound

Type
int_array

Description
The nr. of vertices per bound.

Shape
[nboundaries]

UnitCell(real space)%nvertices

Type
int

Description
The nr. of vertices of the cell.

`UnitCell(real space)%vertices`

Type
float_array

Description
The vertices of the bounds.

Unit
a.u.

Shape
[ndim, nvertices]

Vibrations

Section content: Information related to molecular vibrations.

`Vibrations%ExcitedStateLifetime`

Type
float

Description
Raman excited state lifetime.

Unit
hartree

`Vibrations%ForceConstants`

Type
float_array

Description
The force constants of the vibrations.

Unit
hartree/bohr^2

Shape
[nNormalModes]

`Vibrations%Frequencies[cm-1]`

Type
float_array

Description
The vibrational frequencies of the normal modes.

Unit
cm^-1

Shape
[nNormalModes]

`Vibrations%Intensities[km/mol]`

Type
float_array

Description
The intensity of the normal modes.

Unit
km/mol

Shape
[nNormalModes]

`Vibrations%IrReps`

Type
lchar_string_array

Description
Symmetry symbol of the normal mode.

Shape
[nNormalModes]

`Vibrations%ModesNorm2`

Type
float_array

Description
Norms of the rigid motions.

Shape
[nNormalModes+nRigidModes]

`Vibrations%ModesNorm2*`

Type
float_array

Description
Norms of the rigid motions (for a given irrep…?).

Shape
[nNormalModes+nRigidModes]

`Vibrations%nNormalModes`

Type
int

Description
Number of normal modes.

`Vibrations%NoWeightNormalMode(#)`

Type
float_array

Description
?.

Shape
[3, Molecule%nAtoms]

`Vibrations%NoWeightRigidMode(#)`

Type
float_array

Description
?

Shape
[3, Molecule%nAtoms]

Vibrations%nRigidModes

Type
int

Description
Number of rigid modes.

Vibrations%nSemiRigidModes

Type
int

Description
Number of semi-rigid modes.

Vibrations%PVDOS

Type
float_array

Description
Partial vibrational density of states.

Values range
[0.0, 1.0]

Shape
[nNormalModes, Molecule%nAtoms]

Vibrations%RamanDepolRatioLin

Type
float_array

Description
Raman depol ratio (lin).

Shape
[nNormalModes]

Vibrations%RamanDepolRatioNat

Type
float_array

Description
Raman depol ratio (nat).

Shape
[nNormalModes]

Vibrations%RamanIncidentFreq

Type
float

Description
Raman incident light frequency.

Unit
hartree

Vibrations%RamanIntens[A^4/amu]

Type
float_array

Description
Raman intensities

Unit
A^4/amu

Shape
[nNormalModes]

Vibrations%ReducedMasses

Type
float_array

Description
The reduced masses of the normal modes.

Unit
a.u.

Values range
[0, '\infinity']

Shape
[nNormalModes]

Vibrations%RotationalStrength

Type
float_array

Description
The rotational strength of the normal modes.

Shape
[nNormalModes]

Vibrations%TransformationMatrix

Type
float_array

Description
?

Shape
[3, Molecule%nAtoms, nNormalModes]

Vibrations%VROACIDBackward

Type
float_array

Description
VROA Circular Intensity Differential: Backward scattering.

Unit
10^-3

Shape
[nNormalModes]

Vibrations%VROACIDDePolarized

Type
float_array

Description
VROA Circular Intensity Differential: Depolarized scattering.

Unit
10^-3

Shape
[nNormalModes]

Vibrations%VROACIDForward

Type
float_array

Description
VROA Circular Intensity Differential: Forward scattering.

Unit
10^-3

Shape
[nNormalModes]

Vibrations%VROACIDPolarized

Type
float_array

Description
VROA Circular Intensity Differential: Polarized scattering.

Unit
10^-3

Shape
[nNormalModes]

Vibrations%VROADeltaBackward

Type
float_array

Description
VROA Intensity: Backward scattering.

Unit
10^-3 A^4/amu

Shape
[nNormalModes]

Vibrations%VROADeltaDePolarized

Type
float_array

Description
VROA Intensity: Depolarized scattering.

Unit
10^-3 A^4/amu

Shape
[nNormalModes]

`Vibrations%VROADeltaForward`

Type
float_array

Description
VROA Intensity: Forward scattering.

Unit
10^-3 A^4/amu

Shape
[nNormalModes]

`Vibrations%VROADeltaPolarized`

Type
float_array

Description
VROA Intensity: Polarized scattering.

Unit
10^-3 A^4/amu

Shape
[nNormalModes]

`Vibrations%ZeroPointEnergy`

Type
float

Description
Vibrational zero-point energy.

Unit
hartree

WScell(real_space)
Section content: The Wigner Seitz cell.

`WScell(real_space)%boundaries`

Type
float_array

Description
Normal vectors for the boundaries.

Shape
[ndim, nboundaries]

WScell(real_space)%distances

Type
float_array

Description
Distance to the boundaries.

Shape
[nboundaries]

WScell(real_space)%idVerticesPerBound

Type
int_array

Description
The indices of the vertices per bound.

Shape
[nvertices, nboundaries]

WScell(real_space)%latticeVectors

Type
float_array

Description
The lattice vectors.

Shape
[3, :]

WScell(real_space)%nboundaries

Type
int

Description
The nr. of boundaries for the cell.

WScell(real_space)%ndim

Type
int

Description
The nr. of lattice vectors spanning the Wigner-Seitz cell.

WScell(real_space)%numVerticesPerBound

Type
int_array

Description
The nr. of vertices per bound.

Shape
[nboundaries]

WScell(real_space)%nvertices

Type
int

Description
The nr. of vertices of the cell.

WScell(real_space)%vertices

Type
float_array

Description
The vertices of the bounds.

Unit
a.u.

Shape
[ndim, nvertices]

WScell(reciprocal_space)

Section content: The Wigner Seitz cell of reciprocal space, i.e. the Brillouin zone.

WScell(reciprocal_space)%boundaries

Type
float_array

Description
Normal vectors for the boundaries.

Shape
[ndim, nboundaries]

WScell(reciprocal_space)%distances

Type
float_array

Description
Distance to the boundaries.

Shape
[nboundaries]

WScell(reciprocal_space)%idVerticesPerBound

Type
int_array

Description
The indices of the vertices per bound.

Shape
[nvertices, nboundaries]

WScell(reciprocal_space)%latticeVectors

Type
float_array

Description
The lattice vectors.

Shape
[3, :]

WScell(reciprocal_space)%nboundaries

Type
int

Description
The nr. of boundaries for the cell.

`WScell(reciprocal_space)%ndim`

Type
int

Description
The nr. of lattice vectors spanning the Wigner-Seitz cell.

`WScell(reciprocal_space)%numVerticesPerBound`

Type
int_array

Description
The nr. of vertices per bound.

Shape
[nboundaries]

`WScell(reciprocal_space)%nvertices`

Type
int

Description
The nr. of vertices of the cell.

`WScell(reciprocal_space)%vertices`

Type
float_array

Description
The vertices of the bounds.

Unit
a.u.

Shape
[ndim, nvertices]

ZlmFitConfig

Section content: Configuration options for the Zlm density fit.

`ZlmFitConfig%densityThresh`

Type
float_array

Description
Threshold for the density.

Shape
[nAtoms]

`ZlmFitConfig%gridAngOrder`

Type
int_array

Description
Angular order (Lebedev grid) per atom.

Shape
[nAtoms]

ZlmFitConfig%lMaxExpansion

Type
int_array

Description
Maximum l-value for the fit functions per atom.

Shape
[nAtoms]

ZlmFitConfig%nAtoms

Type
int

Description
Number of atoms.

ZlmFitConfig%nRadialPoints

Type
int_array

Description
Number of radial points per atom.

Shape
[nAtoms]

ZlmFitConfig%partitionFunThresh

Type
float_array

Description
Threshold for the partition function.

Shape
[nAtoms]

ZlmFitConfig%partitionSizeAdjustment

Type
bool

Description
Atom dependent partition size?

ZlmFitConfig%potentialThresh

Type
float_array

Description
Threshold for the potential.

Unit
a.u.

Shape
[nAtoms]

ZlmFitConfig%pruning

Type
bool

Description
Whether or not to prune.

ZlmFitConfig%pruningGridAngOrder

Type
int

Description
?.

ZlmFitConfig%pruningL

Type
int

Description
?.

ZlmFitConfig%pruningThreshDist

Type
float

Description
Distance threshold for pruning.

Unit
bohr

CHAPTER

FOURTEEN

FAQ

14.1 Where can I find the total energy in the BAND output?

BAND does not calculate the total energy. Instead, it calculates the formation energy with respect to the spherically symmetric spin-restricted atoms. The reason is that this number is easier to determine accurately using numerical integration then the total energy.

14.2 What to do with this PEDA ERROR? ERROR DETECTED: Fragments cannot be assigned by a simple translation!

The problem is that BAND is (internally) centralizing the atom positions with respect to the geometrical center of the structure. During this centralization an atom can be positioned on one side of the unit cell or the other. (the result for this individual calculation will be the same in both cases, but for a PEDA calculation that is not true) In your case, the centralization of the fragments and of the PEDA calculation gave results which cannot be transformed into one another by a simple translation.

The solution is to pre-centralize the whole system with respect to the geometrical center of the "bigger" fragment. (since this fragment usually gives this problem) E.g. if you study a surface-adsorbate interaction, you centralize the whole structure w.r.t. the geometrical center of the surface. In some cases the adsorbate fragment might be the reason of the problem. If so, we recommend redefining the unit cell so that the adsorbate is closer to the center of the unit cell than to any of the borders of the unit cell.

If this approach does not solve your problem please send us (support@scm.com) your input and output, so we can look into it.

14.3 What are the units in the Density Of States (DOS) plot?

The dimension of the "density of states per unit cell" (DOS)

$$D(E) = \int_k dk \sum_n \delta(E - \epsilon_n(k))$$

is [1/(energy)].

As the number of states at precisely one energy can be a very wild function an average is plotted (not affecting the unit)

$$D^{\text{smoothed}}(E) = \frac{1}{\Delta_E} \int_E^{E+\Delta_E} D(e)\, de$$

The bin-width Δ_E is controlled by the input option DOS%DeltaE. To obtain the exact non-smoothed dos, set the Dos%IntegrateDeltaE option to "false". As stated before: the non-smoothed DOS will be a very wild function, and for instance bands with little dispersion may be completely missed.

14.4 Why do I get negative partial Density Of States (partial DOS)?

The negative values in the partial DOS are artifacts due to the fact that in a non-orthogonal basis set the definition of a "partial DOS" is somewhat arbitrary. The partial DOS can be a useful tool for understanding the physics/chemistry of your system, but it's strictly speaking not a "physical quantity".

14.5 Why is the DOS zero in an interval with bands?

Quite often there is missing DOS: in an energy interval where there are clearly bands there is no DOS. This is an artifact of using insufficient k-sampling. The simple solution is to redo the whole calculation with a finer k-grid. However, it turns out that the finer grid is only really needed for the DOS, and *restarting the DOS* (page 189) with a finer k-grid is usually sufficient to solve such a problem.

14.6 Is the absolute value of the Fermi energy physically meaningful?

Absolute values of orbital energies are physically meaningful for non-periodic systems, 1D periodic systems and 2D periodic systems. For these cases, E=0 corresponds to the energy of a free electron and the absolute value of the Fermi energy is equal (to a first approximation) to the work function.

In 3D periodic systems the orbital energies (and Fermi energy) are defined up to a constant, ergo the absolute value of energy bands does not have a clear physical meaning (we nonetheless use the convention of setting V_{k=0} = 0, like most other programs).

14.7 Why does the band structure of Graphene look wrong?

There can be some issues. The first problem can be that a super cell is used. The smallest possible (primitive) cell has only two atoms in the unit cell. Only when using the primitive cell, the crossing of two bands at the fermi energy in the point with symmetry label "K" can be seen.

Even when using the primitive cell, the fermi energy may be off: it is not exactly where the two bands cross at "K". This can happen when the grid used during the SCF (or during the SCC of DFTB) does not contain the point "K". The simplest way to solve this is to switch to the symmetric grid and use for instance kInteg=5.

```
KSpace
    Type Symmetric
    Symmetric
        KInteg 5   # Should be an odd number bigger than 1. This value is good for␣
↪Graphene
    End
end
```

You can also achieve this via amsinput. Go to Details -> K-Space integration. Set K-space grid type to "Symmetric" and enter 5 in "Accuracy".

INDEX

U

V

X

Y

Z