

# The first calculation with DFTB+

Bálint Aradi

21st September 2007

## Abstract

This document should serve as a tutorial guiding you through your first calculation with DFTB+. As an example, the equilibrium geometry of a water molecule is calculated. Please note, that this tutorial only covers a fraction of the capabilities of DFTB+, and even those features used are not discussed in any detail. You should consult the current manual for a complete and detailed description of all the DFTB+ features.

## Contents

<b>1</b>	<b>Providing the input</b>	<b>1</b>
1.1	Geometry . . . . .	2
1.2	Driver . . . . .	3
1.3	Hamiltonian . . . . .	4
1.4	Options . . . . .	6
1.5	ParserOptions . . . . .	6
<b>2</b>	<b>Running DFTB+</b>	<b>7</b>
<b>3</b>	<b>Examining the output</b>	<b>9</b>
3.1	Standard output . . . . .	9
3.2	dftb_pin.hsd . . . . .	12
3.3	detailed.out . . . . .	13
3.4	band.out . . . . .	15
3.5	results.tag . . . . .	16
3.6	Other output files . . . . .	16

# 1 Providing the input

At first, you have to create the input for the code. DFTB+ accepts either Human-readable Structured Data (HSD) or eXtended Markup Language (XML) input. In this tutorial HSD will be used, and in this case the input file must be called `dftb_in.hsd`. The input file used in this howto looks as follows: (You can obtain this file by downloading the provided tar.gz archive for this howto from the howto overview page.)

```
Geometry = GenFormat {
3 C
O H
  1  1  0.00000000000E+00 -0.10000000000E+01  0.00000000000E+00
  2  2  0.00000000000E+00  0.00000000000E+00  0.78306400000E+00
  3  2  0.00000000000E+00  0.00000000000E+00 -0.78306400000E+00
}

Driver = ConjugateGradient {
  MovedAtoms = Range { 1 -1 }
  MaxForceComponent = 1.0e-4
  MaxSteps = 100
  OutputPrefix = "geom.out"
}

Hamiltonian = DFTB {
  SCC = Yes
  SCCTolerance = 1.0e-5
  MaxSCCIterations = 1000
  Mixer = Broyden {
    MixingParameter = 0.2
  }
  SlaterKosterFiles = {
    O-O = "O-O.skf"
    O-H = "O-H.skf"
    H-O = "O-H.skf"
    H-H = "H-H.skf"
  }
  MaxAngularMomentum = {
    O = "p"
    H = "s"
  }
  Charge = 0.0
  SpinPolarisation = {}
  Filling = Fermi {
    Temperature [Kelvin] = 0.0
  }
}
```

```
Options = {}
```

```
ParserOptions = {  
  ParserVersion = 3  
}
```

The order of the specified properties in the HSD input is arbitrary. You are free to capitalise the keywords and physical units as you like, since they are case-insensitive. This is not valid, however, for string values, especially if they are specifying file names.

## 1.1 Geometry

The Geometry property contains the types and coordinates of the atoms in your system. The geometry of the system in the sample input file is provided in the so called "gen" format, which was the traditional geometry input format of the DFTB method. The formal description of this format can be found in the DFTB+ manual. The current example

```
Geometry = GenFormat {  
  3 C # Cluster with 3 atoms  
  O H # Two elements, 1 - O, 2 - H  
  # Index Type Coordinates  
  1 1 0.0000000000E+00 -0.1000000000E+01 0.0000000000E+00  
  2 2 0.0000000000E+00 0.0000000000E+00 0.7830640000E+00  
  3 2 0.0000000000E+00 0.0000000000E+00 -0.7830640000E+00  
}
```

specifies a cluster system with 3 atoms of the type O and H. The coordinates of the atoms in the "gen" format are given in Angstroms. The first column of integers contains the sequential number of the atoms in the system (ignored by the parser). The second column contains the type of each atom, given as the position of the appropriate element in the element list in the second line. The GenFormat{} is not the only method to specify the geometry, you should check the manual for other methods.

As demonstrated above, it is possible to put arbitrary comments in the HSD input after a hash-mark (#) character. Everything between the hash and the end of the line is ignored by the parser.

Very often, the geometry is stored in an external file different from dftb\_in.hsd. To save you the copying and pasting from that file into the input file, you can use the file inclusion feature of the HSD format:

```
Geometry = GenFormat {  
  <<< "input_geometry.gen"  
}
```

The <<< operator includes the specified file as raw data. (The file is not checked for any HSD constructs.) In the example above, the file input\_geometry.gen must be in gen format of course.

## 1.2 Driver

After having specified the geometry of your system, you should decide, what to do with that geometry. The Driver property determines, how the geometry should be changed (if at all) during the run. If you only would like to make a static calculation, you must set it to an empty value like

```
Driver = {} # Empty value for the driver
```

In the current example, however,

```
# Do conjugate gradient optimisation
Driver = ConjugateGradient {
  MovedAtoms = Range { 1 -1 } # Move all atoms in the system
  MaxForceComponent = 1.0e-4 # Stop if maximal force below 1.0e-4
  MaxSteps = 100 # Stop after maximal 100 steps
  OutputPrefix = "geom.out" # Final geometry in geom.out.{xyz,gen}
}
```

the molecule is relaxed using the conjugate gradient method. The entire range of atoms from the first (1) until and including the last one (-1) is allowed to move. Instead of Range { 1 -1 } you could also have written

```
MovedAtoms = Range { 1 3 } # Atoms from the 1st until the 3rd
```

or

```
MovedAtoms = { 1 2 3 } # Explicitly listing the atoms to move
```

In our case the geometry optimisation continues as long as the maximal force component is bigger than  $1e-4$  atomic units (Hartree/Bohr). Numeric values are per default interpreted in atomic units. The HSD format offers, however, the possibility to use alternative units by specifying the unit (modifier) of the value in square brackets before the equal sign. For example instead of the original specification, you could have used

```
MaxForceComponent [eV/AA] = 5.14e-3 # Force in Electronvolt/Angstrom
```

or

```
MaxForceComponent [Electronvolt/Angstrom] = 5.14e-3
```

The MaxSteps property specifies, after how many geometry optimisation steps the program should stop, even if the specified tolerance for the maximal force component could not be reached.

Finally, the OutputPrefix property specifies the name of the file containing the actual geometry during the optimisation (and the final geometry at the end of the calculation). The geometry is written in gen and xyz formats to the files obtained by appending ".gen" and ".xyz" suffixes to the specified name (geom.out.gen and geom.out.xyz in our case).

### 1.3 Hamiltonian

In order to calculate the various properties of your system, you have to decide upon the way how the Hamiltonian for your system should be built. At the moment DFTB+ eases the decision quite a lot, since it only supports a Density Functional based Tight Binding Hamiltonian (with some extensions). In our example, the chosen self-consistent DFTB Hamiltonian has the following properties:

```
Hamiltonian = DFTB {           # DFTB Hamiltonian
  SCC = Yes                    # Use self consistent charges
  SCCTolerance = 1.0e-5       # Tolerance for charge consistence
  MaxSCCIterations = 1000     # Nr. of maximal SCC iterations
  Mixer = Broyden {           # Broyden mixer for charge mixing
    MixingParameter = 0.2     # Mixing parameter
  }
  SlaterKosterFiles = {       # Specifying Slater-Koster files
    O-O = "O-O.skf"
    O-H = "O-H.skf"
    H-O = "O-H.skf"
    H-H = "H-H.skf"
  }
  MaxAngularMomentum = {     # Maximal l-value of the various species
    O = "p"
    H = "s"
  }
  Charge = 0.0                # System neutral
  SpinPolarisation = {}       # No spin polarisation
  Filling = Fermi {           # No temperature
    Temperature [Kelvin] = 0.0
  }
}
```

In this example the SCC-DFTB method is used for building up the Hamiltonian (and calculating the total energy, forces, etc.). The charges are considered to be converged if the difference between the charges using to build the Hamiltonian and the charges obtained after the diagonalisation of the Hamiltonian is below  $1e-5$ . If the convergence is not reached within the specified number of cycles (MaxSCCIterations), the code calculates the total energy using the charges obtained so far. (Appropriate warning messages are printed out.)

In order to damp charge oscillations and to speed up convergence, the charges of the subsequent iterations are mixed together. The Mixer property specifies the type of mixer to use. The mixers usually requires further properties, like the mixing factor for the Broyden mixer above.

The integral tables (together with other atomic and diatomic parameters) necessary for the build up of the Hamiltonian are stored in the so called Slater-Koster files. Those files always describe the interaction between atom pairs. Therefore, you have to specify for every atom pair the corresponding Slater-Koster file.

```
SlaterKosterFiles = {         # Specifying Slater-Koster files
```

```

O-O = "O-O.skf"
O-H = "O-H.skf"
H-O = "O-H.skf"
H-H = "H-H.skf"
}

```

If you are using a certain file name convention, you can avoid typing all the file names by specifying only the generating pattern. The input

```

SlaterKosterFiles = Type2FileNames { # File names with two atom type names
  Prefix = "" # No prefix before first type name
  Separator = "-" # Dash between type names
  Suffix = ".skf" # Suffix after second type name
}

```

would generate exactly the same file names as in the example above. If the Slater-Koster files are in a different directory as the input file `dftb_in.hsd`, you can append the path as prefix:

```

SlaterKosterFiles = Type2FileNames { # File names from two atom type names
  Prefix = "/home/aradi/slako/mio-0-1" # Path as prefix
  Separator = "-" # Dash between type names
  Suffix = ".skf" # Suffix after second type name
}

```

The historical Slater-Koster file format does not contain any information about the orbitals, which were considered when generating the interaction tables. Therefore, you must provide the highest angular momentum for every element as s, p, d or f. This information can be obtained from the documentation of the Slater-Koster files. In the distributed standardised sets (as distributed on [dftb.org](http://dftb.org)) this information is contained in the documentation appended to the end of each SK-file.

The charge of the system (Charge) can be specified as real number. It's the real total charge of your system, so that negative numbers mean excess electrons in the system.

For the spin polarisation you have to specify the type of the spin polarisation to use. If you don't need spin polarisation, you should assign the empty value to it:

```
SpinPolarisation = {}
```

If you wanted to make a spin polarised calculation with colinear spins, you should write something like

```

SpinPolarisation = Colinear {
  UnpairedElectrons = 2.0 # Magnitude of the spin polarisation
}

```

In that latter case you would also have to specify the spin coupling constants in the SpinConstants property. (See the manual for details.)

Finally, the Filling property describes the method to use for filling up the one electron levels with electrons. The filling functions usually require further parameters (e.g the temperature).

## 1.4 Options

The Options property contains a few global settings for the code. In the current example, no options are specified, so that the empty value is assigned to that property:

```
Options = {}
```

You could even leave out this line, since the default value for the Options property is the empty value anyway.

## 1.5 ParserOptions

This block contains options which are interpreted by the parser itself and are not passed to the main program. The most important of those options is the ParserVersion option, which tells the parser, for which parser the current input file was created for. If this is not the current parser but an older one, this allows backwards compatibility.

The version number of the parser in the current DFTB+ code is always printed out at the program start. It is a good habit to set this value in your input files explicitly, like in our case

```
ParserVersion = 3
```

It allows you namely to use your input file with the future versions of DFTB+ without adapting it by hand, if the input format changed in the new versions.

## 2 Running DFTB+

After creating the main input file, you should make sure that all the other needed files (Slater-Koster files, targets of eventual file inclusions in the HSD input) are at the right place. In our case, only the Slater-Koster files needs to be present, and since we specified them without path, they must be in the same directory as dftb\_in.hsd itself. This howto uses Slater-Koster files from the mio-0-1 SK-set. (The Slater-Koster files are not part of the downloadable tarball for this howto.)

In order to make the calculation, you should invoke DFTB+ without arguments in the directory containing dftb\_in.hsd:

```
/tmp/first_calc> dftb+
```

Assuming dftb+ lies in your path, you should obtain output starting with

```
=====
==
== Density Functional based Tight Binding with a lot of extensions (DFTB+)
==
== Release: 1.0 (p0)
```

==  
==  
==

(ParserVersion = 3)

=====

\*\*\*\*\*  
\*\* Parsing and initializing  
\*\*\*\*\*

Interpreting input file 'dftb\_in.hsd'

-----

Processed input in HSD format written to 'dftb\_pin.hsd'

Starting initialization...

-----

Mode: Conjugate gradient relaxation  
Self consistent charges: Yes  
SCC-tolerance: 0.100000E-04  
Max. scc iterations: 1000  
Spin polarisation: No  
Nr. of up electrons: 4.000000  
Nr. of down electrons: 4.000000  
Periodic boundaries: No  
Diagonalizer: Divide and Conquer  
Mixer: Broyden mixer  
Mixing parameter: 0.200000  
Maximal SCC-cycles: 1000  
Nr. of chrg. vec. in memory: 1000  
Nr. of moved atoms: 3  
Max. nr. of geometry steps: 100  
Force tolerance: 0.100000E-03  
Electronic temperature: 0.100000E-07  
Initial charges: Set to neutral  
Maximal angular momentum: O: P  
H: S

Extra options:

-----

\*\*\*\*\*  
\*\* Geometry step: 0  
\*\*\*\*\*

iSCC Total electronic    Diff electronic    SCC error

```

1 -0.39511797E+01  0.00000000E+00  0.88081627E+00
2 -0.39705438E+01 -0.19364070E-01  0.55742893E+00
3 -0.39841371E+01 -0.13593374E-01  0.32497352E-01
4 -0.39841854E+01 -0.48242063E-04  0.19288772E-02
5 -0.39841856E+01 -0.17020682E-06  0.87062163E-05
>> Charges saved for restart in charges.bin

```

```

Total Energy:          -3.979879
Total Mermin free energy: -3.979879
Maximal force component:  0.187090

```

```

*****
** Geometry step:  1
*****

```

```

iSCC Total electronic  Diff electronic  SCC error
1 -0.40495557E+01  0.00000000E+00  0.92334379E-01
.
.
.

```

If this is the case, you managed to run DFTB+ for the first time. Congratulations!

### 3 Examining the output

DFTB+ communicates through two channels with the world: by printing information on the standard output and by writing information to various files. In the following, the most important of those should be introduced and analysed.

#### 3.1 Standard output

The first thing appearing on standard output after the start of DFTB+ is the program header:

```

=====
==
== Density Functional based Tight Binding with a lot of extensions (DFTB+)
==
== Release: 1.0 (p0)
==
== (ParserVersion = 3)
==
=====

```

This tells you which release (1.0) and with patchlevel (p0) of DFTB+ you are running. Then, you get the version of the parser used in this DFTB+ release:

```
(ParserVersion = 3)
```

As already discussed by the input file, it is a good habit to indicate this version number explicitly in your input in the ParserOptions block, like that:

```
ParserOptions = {  
  ParserVersion = 3  
}
```

Then the parser starts to interpret your input:

```
*****  
** Parsing and initializing  
*****
```

```
Interpreting input file 'dftb_in.hsd'
```

```
-----  
Processed input in HSD format written to 'dftb_pin.hsd'
```

You don't have to explicitly set all the possible options for DFTB+ in the input, for most of them there is a default value set by the parser, if the according specification is missing in the input. If you want to know what default values had been set for those missing specifications, you should look at the processed input file `dftb_pin.hsd`, which contains the values for all the possible input settings. (See next subsection.)

After that, the DFTB+ code is initialised, and for the most important quantities the provided values are printed out:

```
Starting initialization...
```

```
-----  
Mode:                Conjugate gradient relaxation  
Self consistent charges:  Yes  
SCC-tolerance:       0.100000E-04  
Max. scc iterations: 1000  
Spin polarisation:   No  
Nr. of up electrons: 4.000000  
Nr. of down electrons: 4.000000  
Periodic boundaries: No  
Diagonalizer:        Divide and Conquer  
Mixer:                Broyden mixer  
Mixing parameter:    0.200000  
Maximal SCC-cycles: 1000  
Nr. of chrg. vec. in memory: 1000  
Nr. of moved atoms: 3  
Max. nr. of geometry steps: 100
```

```

Force tolerance:      0.100000E-03
Electronic temperature: 0.100000E-07
Initial charges:      Set to neutral
Maximal angular momentum: O: P
                      H: S

```

Extra options:

-----

As you can see, all quantities (e.g. force tolerance, electronic temperature) are converted to the internal units of DFTB+, namely atomic units (with Hartree as the energy unit).

Then the program starts:

```

*****
** Geometry step:  0
*****

```

	iSCC Total electronic	Diff electronic	SCC error
1	-0.39511797E+01	0.00000000E+00	0.88081627E+00
2	-0.39705438E+01	-0.19364070E-01	0.55742893E+00
3	-0.39841371E+01	-0.13593374E-01	0.32497352E-01
4	-0.39841854E+01	-0.48242063E-04	0.19288772E-02
5	-0.39841856E+01	-0.17020682E-06	0.87062163E-05

>> Charges saved for restart in charges.bin

```

Total Energy:          -3.979879
Total Mermin free energy: -3.979879
Maximal force component: 0.187090

```

Since this calculation is an SCC calculation, DFTB+ has to iterate the charges until the specified convergence criteria is fulfilled. In every cycle, you get information about the value of the electronic energy, the difference to the value in the previous SCC cycle, and the error in the charges for which the tolerance value had been specified in the input.

If the SCC cycle converged, the total energy including SCC and repulsive contributions is calculated, and similarly the total free energy. Additionally the biggest force component in the system is indicated.

Then the driver changes the geometry of the system, and the same calculation as before is executed for the new geometry. This is done, as long as the geometry does not converge.

```

*****
** Geometry step: 12
*****

```

	iSCC Total electronic	Diff electronic	SCC error
1	-0.41505783E+01	0.00000000E+00	0.20093314E-02
2	-0.41505783E+01	-0.21634569E-07	0.14891915E-02
3	-0.41505784E+01	-0.26364940E-07	0.27059002E-07

>> Charges saved for restart in charges.bin

```
Total Energy:          -4.077938
Total Mermin free energy: -4.077938
Maximal force component: 0.000003
```

Geometry converged

If the geometry does not converge before the number of maximal geometry steps is reached, you will get an appropriate warning. Assuming the MaxSteps property had been set to 6 in the input, you would obtain:

```
*****
** Geometry step: 6
*****
```

```
iSCC Total electronic  Diff electronic  SCC error
  1 -0.41414787E+01  0.00000000E+00  0.12689108E-01
  2 -0.41414797E+01  -0.96455052E-06  0.93470428E-02
  3 -0.41414808E+01  -0.11439438E-05  0.17369905E-05
```

>> Charges saved for restart in charges.bin

```
Total Energy:          -4.077410
Total Mermin free energy: -4.077410
Maximal force component: 0.020793
```

WARNING!

-> !!! Geometry did NOT converge!

!!! Geometry did NOT converge!

### 3.2 dftb\_pin.hsd

As already mentioned the processed input file dftb\_pin.hsd is an input file generated from your input in dftb\_in.hsd by setting the default values for all unspecified options and by converting all numerical values to atomic units. For example, in our case for the Constraints property of the ConjugateGradient{} method the empty value {} had been set as default (no constraints):

```
Driver = ConjugateGradient {
  MovedAtoms = Range {
1 -1
  }
  MaxForceComponent = 1.0e-4
  MaxSteps = 100
  OutputPrefix = "geom.out"
  AppendGeometries = No
  Constraints = {}
}
```

Similarly, in the DFTB{} method the switch for the orbital resolved SCC, for example, had been set to the default value of No:

```
OrbitalResolvedSCC = No
```

The options, which had been explicitly set in the input, are unchanged but the numerical values are converted to atomic units. The file `dftb_pin.hsd` is a valid HSD input file, you can use it as input (after renaming it to `dftb_in.hsd`). It has always the format suitable for the current parser, even if the input `dftb_in.hsd` was in an older format (indicated by the `ParserVersion` property in it). Therefore, the `ParserVersion` property is always set to the version of the current parser in the processed input file `dftb_pin.hsd`.

### 3.3 detailed.out

This file contains detailed informations about the properties of your system, as calculated during the last SCC cycle. It is updated continuously during the run. All the numerical values are given in atomic units unless explicitly specified.

It contains the coordinates of the moved atoms

Coordinates of moved atoms:

1	0.00000000	-1.39142483	0.00000000
2	0.00000000	-0.24915058	1.46137216
3	0.00000000	-0.24915058	-1.46137216

Then the eigenvalues in both Hartree and electronvolt and the occupations of the individual levels for all the possible spin states. For spin unpolarised calculations (like this one) you get only one value, since the eigenlevels are twofold degenerated. In a spin polarised calculation you would obtain separate values for spin up (SPIN = 1) and spin down (SPIN = 2).

```
SPIN =          1
```

Eigenvalues /H

```
-0.84748357  
-0.40904849  
-0.31712254  
-0.25902674  
0.36019547  
0.52423607
```

Eigenvalues /eV

```
-23.06120138  
-11.13077560  
-8.62934348  
-7.04847615  
9.80141738  
14.26518921
```

```

Fillings
  2.00000
  2.00000
  2.00000
  2.00000
  0.00000
  0.00000

```

Then you obtain the number of total electrons in the system, and the number of electrons on each atom, each l-shell (s, p, d, etc.) and each orbital (s, px, py, pz, ...) as calculated by Mulliken-analysis:

```

Charge (spin: 1):      8.00000000
Atom populations (spin: 1)
Atom  1      6.58494725
Atom  2      0.70752638
Atom  3      0.70752638

```

```

l-shell populations (spin: 1)
Atom l      Charge
  1  0      1.74495507
  1  1      4.83999217
  2  0      0.70752638
  3  0      0.70752638

```

```

Orbital populations (spin: 1)
Atom l m      Charge
  1  0  0      1.74495507
  1  1 -1      1.65653919
  1  1  0      1.18345299
  1  1  1      2.00000000
  2  0  0      0.70752638
  3  0  0      0.70752638

```

In our case due to the electronegativity difference the hydrogen atoms are positively polarised having only 0.707 electrons, while the oxygen is negatively polarised having 6.58 electrons (instead of the neutral state of 6 electrons).

After that you find in detailed.out the Fermi energy, the different energy contributions to the total energy and the total energy in Hartree and electronvolt units. If you are calculating at finite temperature, you should consider the Mermin free energy instead of the total energy.

```

Fermi energy:          -0.1616237529 H      -4.398006 eV
Band energy:           -3.6653626807 H     -99.739593 eV
TS:                    0.0000000000 H      0.000000 eV
Band free energy (E-TS): -3.6653626807 H     -99.739593 eV

```

Extrapolated E(0K):	-3.6653626807 H	-99.739593 eV
Input/Output charge (spin: 1):	8.00000000	8.00000000
Energy H0:	-4.1601031215 H	-113.202166 eV
Energy SCC:	0.0186222798 H	0.506738 eV
Total Electronic energy:	-4.1414808416 H	-112.695428 eV
Repulsive energy:	0.0640707607 H	1.743454 eV
Total energy:	-4.0774100809 H	-110.951973 eV
Total Mermin free energy:	-4.0774100809 H	-110.951973 eV

Between the two energy blocks, the input and output charges of the last Hamiltonian diagonalisation are shown, so that you can check, that no charges got lost during the calculation.

Then you find the number of the geometry step in which the data above was calculated, and a confirmation, that the SCC convergence had been reached in this geometry step. (You should always make sure that the values were calculated by using convergent charges. Values obtained by using non convergent charges are usually meaningless.)

Geometry optimization step: 12

SCC converged

Finally you get the electronic, the repulsive and the total forces on the atoms in your system. (the latter is the sum of the former two.) You get also the maximal force component occurring in your system and the maximal force occurring among the moved atoms. Next the confirmation that the geometry optimisation has reached convergence, so that all force components on the moved atoms are below the specified tolerance.

Full geometry written in geom.out.{xyz|gen}

Electronic Forces

7.669669336132177E-029 0.198329781854174 1.369737656631287E-014  
-5.154642551667670E-017 -9.916489092709796E-002 -0.134494278455826  
5.154642551660000E-017 -9.916489092707595E-002 0.134494278455812

Repulsive Forces

9.040777382140338E-029 -0.198329709564107 -1.204591981718295E-013  
2.307327498763467E-017 9.916485478210176E-002 0.134491462500833  
-2.307327498772508E-017 9.916485478200551E-002 -0.134491462500713

Total Forces

1.671044671827252E-028 7.229006665498972E-008 -1.067618216055166E-013  
-2.847315052904203E-017 -3.614499620441247E-008 -2.815954992990699E-006  
2.847315052887492E-017 -3.614507043669946E-008 2.815955099738643E-006

Maximal force component: 2.815955099738643E-006  
Maximal force component for moved atoms: 2.815955099738643E-006

Geometry converged

The final geometries can be found in xyz and gen formats in the indicated output files (geom.out.xyz and geom.out.gen in our case).

### 3.4 band.out

For big systems, and especially for periodic systems with many k-points, it can become quite difficult to get a good overview over the one electron levels and their occupations in detailed.out. Therefore, the extra file band.out is created, which contains this information in a more human readable format. The eigenenergies are in atomic units.

KPT	1	SPIN	1
-23.10207	2.00000		
-11.27463	2.00000		
-8.53775	2.00000		
-7.05253	2.00000		
10.86408	0.00000		
15.19413	0.00000		

Despite its name, this file is created also for non-periodic systems, containing the eigenenergies and occupation numbers for the calculated molecule. (You should ignore the K-point index in the first line for that case.)

### 3.5 results.tag

If you want to process the results of DFTB+ with another program, you should not extract the informations from the standard output or the human readable output files (detailed.out, band.out), since their format could change significantly between subsequent releases of DFTB+. By setting the WriteResultsTag to Yes in the Options block

```
Options = {  
  WriteResultsTag = Yes  
}
```

you obtain the file results.tag at the end of your calculation containing some of the most important data in a format easily parsed by a script or a program. (We'll make our Python parser for this format freely available after tidying it up a little bit.) The file contains entries like:

```
forces          :real:2:3,3  
-0.277549616145533E-028  0.722900585226061E-007 -0.427713420236842E-013
```

-0.109618479408687E-015 -0.361450127051022E-007 -0.281595502166221E-005  
0.109618479408715E-015 -0.361450458036261E-007 0.281595506446131E-005

In the first line the name of the quantity is given, followed by its type (real, integer, logical). Then the rank of the quantity is given (0 – scalar, 1 – vector, 2 – rank 2 matrix, etc.), followed by the size of each dimension. In the following the value of the given quantity is dumped in a free format.

Unfortunately not all properties found in detailed.out are also represented in results.tag yet, but beginning with release 1.1, the same information should be present in both files.

### 3.6 Other output files

There are also other output files not discussed in detail here. They are only created, if the appropriate option in the Options block is set. Those files and their switches in the Options block are the following:

File name(s)	Switching Option	Short description
eigenvec.out	WriteEigenvectors	Eigenvectors in human readable format
eigenvec.bin	WriteEigenvectors	Eigenvectors in binary format
autotest.tag	WriteAutotestTag	Output file for the autotest system
detailed.xml	WriteDetailedXML	A fraction of detailed.out in XML format.

For a detailed description please consult the manual.