SCM
Software for
Chemistry &
Materials

MCFF - Reparametrization

NSCCS ADF/ReaxFF Workshop

Ole Carstensen
Fedor Goumans
Anna Shchygol

Imperial College London
27th -28th September 2016

Intro

This tutorial will teach you how to get started on reparametrizing ReaxFF using tools form the ADF Modeling Suite. Some things should be said, before starting:

What this tutorial is *not* offering:

- The final answer to the reparametrization of ReaxFF.
- A production-ready force field. We will use a minimal amount of training data and the DFT reference is of low quality to speed things up.
- A blackbox.

What this tutorial *should* offer:

- An introduction to using the MCFF optimizer, including
  - one way to create a reasonable initial guess for the force field
  - instructions on how to create the required inputfiles
  - instructions on how to set up the simulated annealing program
  - instructions on how to check the quality of the force field against reference data

The MCFF optimizer is one of the many approaches to a complex global optimization, which will typically outperform hand-based optimization with less manual effort.
Some of the approaches outlined in this tutorial are empiric. They might work for other systems, but there is no guarantee whatsoever. You should be willing to experiment, simply because - in the end - whoever creates the best force field is right. Don't get discouraged, reparametrizing ReaxFF is tough, but it's possible...

MCFF – General

Inputfiles

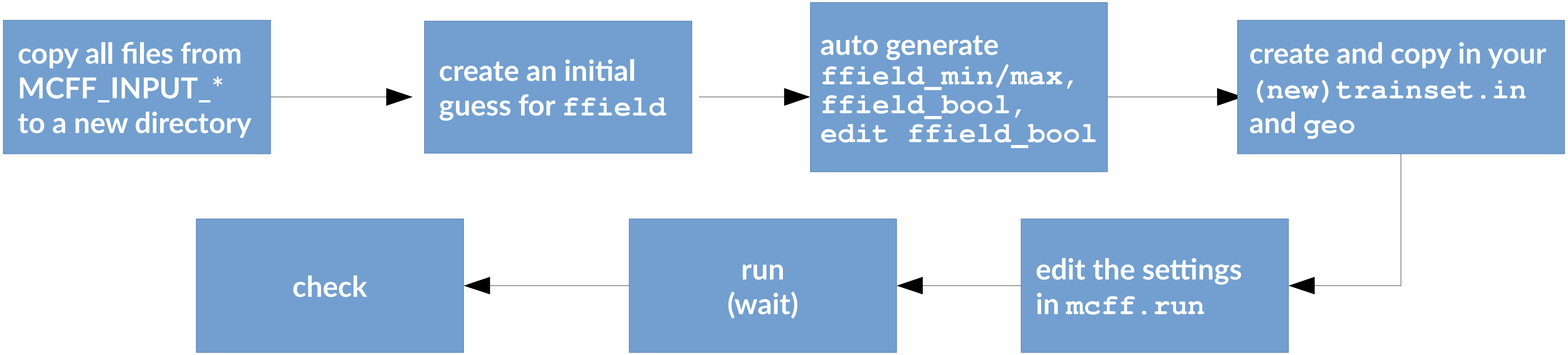
MCFF

List of input files

In order to run the MCFF parameter optimization the following input files need to present:

- `mcff.run` – the executable run script that contains the general ReaxFF settings as well as the MCFF settings and the number of cores for parallel execution
- `ffield` – the initial guess of the forcefield to be optimized
- `ffield_min` – the lower ranges for all parameters
- `ffield_max` – the upper ranges for all parameters
- `ffield_bool` – a forcefield file containing boolean (1/0) values for each parameter. Parameters flagged as 0, i.e. *off*, will be kept constant and not optimized.
- `trainset.in` – holds the (QM) training data and the weights for the Errorfunction
- `geo` – holds the coordinates of the structures in the training set and assigns their IDs

How to prepare and run an MCFF optimization:



copy all files from MCFF_INPUT_* to a new directory

create an initial guess for `ffield`

auto generate `ffield_min/max`, `ffield_bool`, `edit ffield_bool`

create and copy in your `(new)trainset.in` and `geo`

edit the settings in `mcff.run`

run (wait)

check

MCFF: Input files

`ffield, ffield_min, ffield_max, ffield_bool`

The `ffield*` files define the initial forcefield, ranges and active parameters.

The format is the one used in the original ReaxFF code (see here , p. 15ff), from the start of the ffield file to towards the end the parameters are organized as follows:

- (up to) 39 general parameters
- (up to) 32 atomic parameters / atom
- (up to) x interatomic parameters
  - bonds (16/bond) , off-diagonal terms (6/atom pair), angles (7/angle), torsions (7/torsion), hydrogen bonds (4/atom pair)

IMPORTANT: The format of these files is *very strict and* parsing errors are hard to track down. Therefore, after editing <u>any</u> of these files by hand Hans van Schoot's cleanup script should be called to correct the formatting for you:

```
startpython cleanreaxffforcefield.py -i [input ffield] -o [output ffield]
```

The files ffield_min, ffield_max and ffield_bool can be generated from an `ffield` file via `adfreport`:

```
adfreport ffield -ffield-min  > ffield_min
adfreport ffield -ffield-max  > ffield_max
adfreport ffield -ffield-bool > ffield_bool
```

Note:
This will set default ranges taken from existing force fields, Initially all parameters in ffield_bool will be 0, i.e. fixed and have to be edited by hand.

MCFF: Input files

`mcff.run`

The file `mcff.run` contains the settings for the actual MCFF run as well as some general ReaxFF settings. A detailed description of the MCFF settings can be found online.

```
[..snip..]

# ======================================
#       MCFF - Settings
# ======================================

  0.00010 mcbeta    initial MC beta parameter
  [..snip..]

# ======================================
#        Nr. of cores
# ======================================

export NSCM=4

[..snip..]
```

Jump to the end of the file to find the relevant settings for the MCFF optimizer and the number of cores for parallel execution...

For this tutorial the following settings are (somewhat) most important:

- `mcbeta` - the initial β, i.e. the inverse 'Temperature' for the simulated annealing → $\beta = 1/(k_b T)$
- `mcdbet` - the rate at which β changes [increment/step]
- `mcbsca` - divide beta by this value at every step
- `mcffit` - number of MC iterations

MCFF: Input files
`trainset.in`

→ hands-on (geo opt)
→ hands-on (lin transit)

```
CHARGE
#Iden Weight Atom Lit
chexane 0.1 1 -0.15
ENDCHARGE

HEATFO
#Iden Weight Lit
methane 2.00 -17.80 !Heat of formation
ENDHEATFO

GEOMETRY
#Iden
Weight At1 At2 At3 At4 Lit
chexane 0.01 1 2      1.54  !bond
chexane 1.00 1 2 3  111.0   !valence angle
chexane 1.00 1 2 3 4 56.0   !torsion angle
chexane 1.00 0.01           !RMSG
ENDGEOMETRY

CELL PARAMETERS
#Iden Weight Type Lit
chex_cryst 0.01 a 11.20
END CELL PARAMETERS

ENERGY
#Weigh op1 Ide1 n1 op2 Ide2 n2 Lit
#alfa vs. beta vs. gamma cleavage in butylbenzene
1.5 + butbenz/1 - butbenz_a/1 -90.00
1.5 + butbenz/1 - butbenz_b/1 -71.00
1.5 + butbenz/1 - butbenz_c/1 -78.00
#cyclohexane heat of vaporization
1.0 + chex_cryst/16 - chexane/1 -11.83
ENDENERGY
```

The file trainset.in contains the actual training data as well as the weights used to construct the Error function. It's in the same format that is described in the original ReaxFF manual (p. 28ff).

The following properties can be defined:

- Charges
- Heat of formations [kcal/mol]
- Cell parameters [Å or °]
- Geometries [Å or °]
- Root mean square forces, RMSG [kcal/molÅ]
- Relative energies [kcal/mol]

The format supports basic math operators (/,+,-) as shown in the ENERGY section on the left.

The structures are referenced via IDs defined in the BGF files (see geo)

MCFF: Input files

geo

→ hands-on (geo opt)
→ hands-on (lin transit)

```
BIOGRF 200
DESCRP CH3Br_C-Br_2
REMARK Created by adfreport
REMARK DataFrom MCFF/TRAININGSET/CH3Br_C-Br/CH3Br_C-Br.t21
RUTYPE SINGLE POINT
MOLCHARGE 1 5 0
HETATM     1 C            -0.83040  -1.78118   0.05708    C    1 1  -0.27800
HETATM     2 Br            0.59750  -1.07009  -0.09679   Br    1 1  -0.57600
HETATM     3 H            -0.91574  -2.78935  -0.39267    H    1 1   0.28500
HETATM     4 H            -1.68988  -1.24587  -0.39160    H    1 1   0.28500
HETATM     5 H            -1.17306  -1.95235   1.09603    H    1 1   0.28500
UNIT ENERGY   kcalmol
ENERGY -423.178020295
END

BIOGRF 200
[..snip..]
```

The ID *`CH3Br_C-Br_2`* is used to reference this structure in `trainset.in`. `It must be unique.`

`RUTYPE` (sic!) determines what job ReaxFF will perform.
No *`RUTYPE`* or *`RUTYPE NORMAL`* will run the default specified in the `control` file. In this tutorial the default is a geometry optimization.

The file `geo` contains the coordinates for the structures in your training set.
It also sets the ID referred to in the `trainset.in` file within the field `DESCR`.

Suitable BGF files can be created via `adfreport` or the ADF version of `openbabel`:

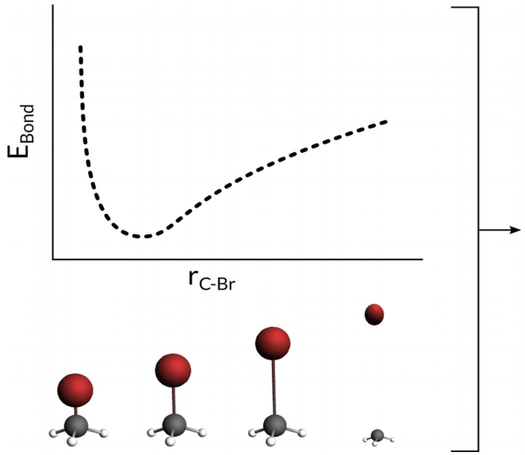
```
run_babel -ixyz some_geometry.xyz -obgf ...
```

MCFF: Input files

Generating `trainset.in` and `geo`

With the ADF Modeling Suite ReaxFF training data can be automatically generated in several ways:

- from *linear transits*, e.g. relaxed bond scans, via the PYTHON script `LT_to_trainset.py`
  - setup a scan with the help of the GUI (see hands on, here)
  - run: `startpython LT_to_trainset.py [filename].t21`



```
ENERGY
 1.0 + CH3Br_C-Br_1/1 - CH3Br_C-Br_6/1 104.6834
 1.0 + CH3Br_C-Br_2/1 - CH3Br_C-Br_6/1 56.9133
 1.0 + CH3Br_C-Br_3/1 - CH3Br_C-Br_6/1 28.0859
 1.0 + CH3Br_C-Br_4/1 - CH3Br_C-Br_6/1 11.5885
 1.0 + CH3Br_C-Br_5/1 - CH3Br_C-Br_6/1 3.1519
[..snip..]
ENDENERGY

CHARGE
  CH3Br_C-Br_1 0.10 -1.20
 CH3Br_C-Br_1 0.10 0.24
 CH3Br_C-Br_1 0.10 0.32
[..snip...]
ENDCHARGE
```

```
BIOGRF 200
DESCRP CH3Br_C-Br_1
REMARK Created by adfreport
REMARK DataFrom ./MCFF/TRAININGSET/CH3Br_C-Br/CH3Br_C-Br.t21
MOLCHARGE 1 5 0
HETATM     1 C    -0.75252  -1.74225   0.04859    C   1 1  -0.27800
HETATM     2 Br    0.58406  -1.07679  -0.09533   Br   1 1  -0.57600
HETATM     3 H    -0.87800  -2.75778  -0.39020    H   1 1   0.28500
HETATM     4 H    -1.64380  -1.23624  -0.38892    H   1 1   0.28500
HETATM     5 H    -1.13188  -1.93212   1.07814    H   1 1   0.28500
UNIT ENERGY   kcalmol
ENERGY -375.407858952
END

BIOGRF 200
DESCRP CH3Br_C-Br_2
[..snip...]
```

`CH3Br_C-Br.t21` → **`trainingdata_CH3Br_C-Br.t21`** + **`geos_CH3Br_C-Br.t21`**

- via `adfreport` on geometry optimizations and singlepoints

  run:

  ```
  adfreport [filename].t21 BGF > geometry
  adfreport geometry -rxtrainset > trainingdata
  ```

Hint:
use ">>" instead of ">"
to append to an existing file...

The files `geometry` and `trainingdata` now contain your geometry in BGF format and charges with geometry data respectively

MCFF – Hands on

creating parameters for a bromination reaction

MCFF: Hands-on

The bromination reaction

The reaction we will be optimizing a forcefield for is the radical bromination of methane:

chain propagation

1. Br-Br → 2 Br·
2. $CH_4 + Br\cdot \rightarrow CH_3\cdot + Hbr$
3. $CH_3\cdot + Br_2 \rightarrow CH_3Br + Br\cdot$

chain termination

1. Br· + Br· → Br2
2. $CH_3\cdot + Br\cdot \rightarrow CH_3Br$
3. $CH_3\cdot + CH_3\cdot \rightarrow C_2H_6$

species

Br_2, Br·, HBr

CH_4, $CH_3\cdot$

CH_3Br, C_2H_6

Why?
+ rather simple reaction → a good start
+ small molecules, DFT calculations are fast
+ there are no Br-parameters available yet
+ the concepts should be transferable to more complex systems

Learn how to:

- come up with an initial guess for your force field parameters
- create and extend a FF training set
- run the MCFF optimizer
- check for overfitting

MCFF – Hands on

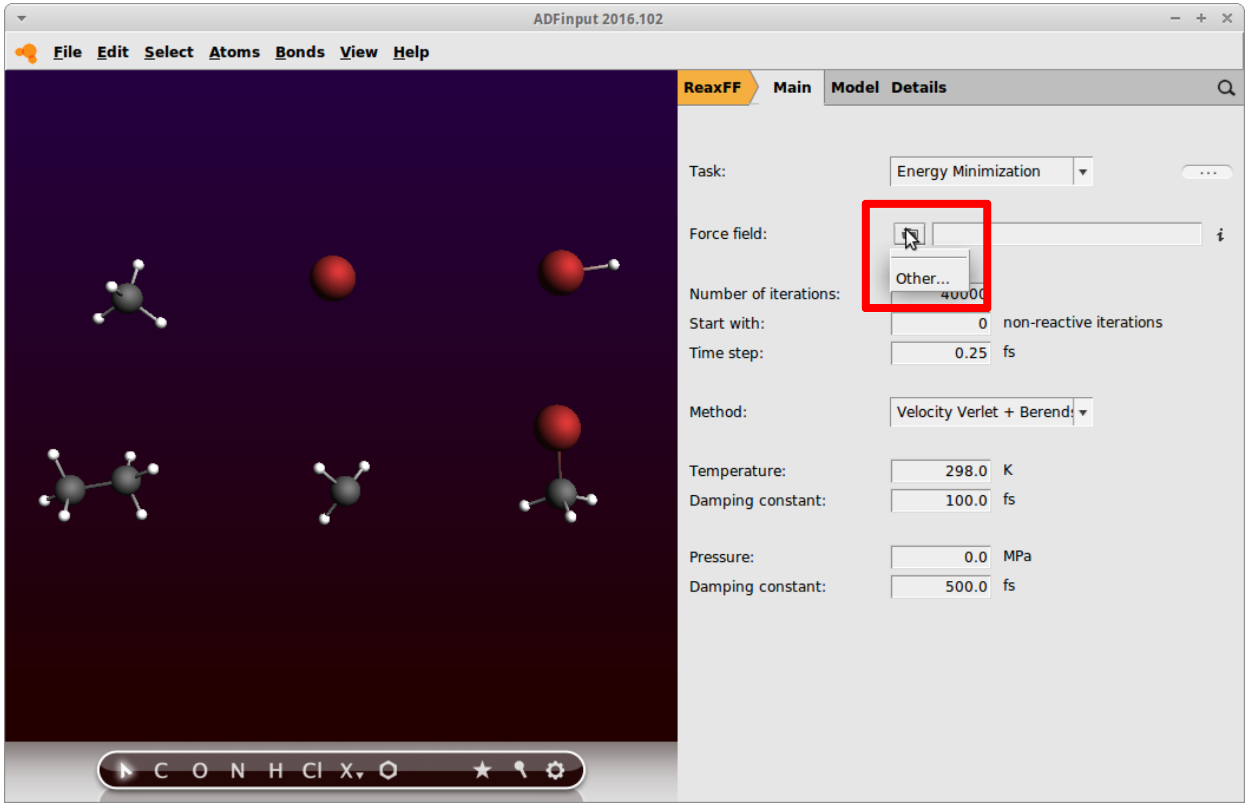
creating an initial guess
for the force field

MCFF: Hands-on

creating an initial guess for the force field, where to start?

In order to create an initial guess for the new force field, we inspect if we can utilize an existing force field for our purposes.

The easiest way to check is by starting up ADFinput drawing the molecules we are interested in and try to select a force field for an energy minimization:



Task: Energy Minimization

Force field: Other...

Number of iterations: 4000

Start with: 0 non-reactive iterations

Time step: 0.25 fs

Method: Velocity Verlet + Berendsen

Temperature: 298.0 K

Damping constant: 100.0 fs

Pressure: 0.0 MPa

Damping constant: 500.0 fs

Depending on your system, the outcome might be:

- No FF available
  - see next slides
- Choose a FF
  - select the one that gives the most reasonable geometries. Use a copy of this FF. *
- All FF files are located inside the ADFHOME directory under: `atomicdata/ForceFields/ReaxFF`

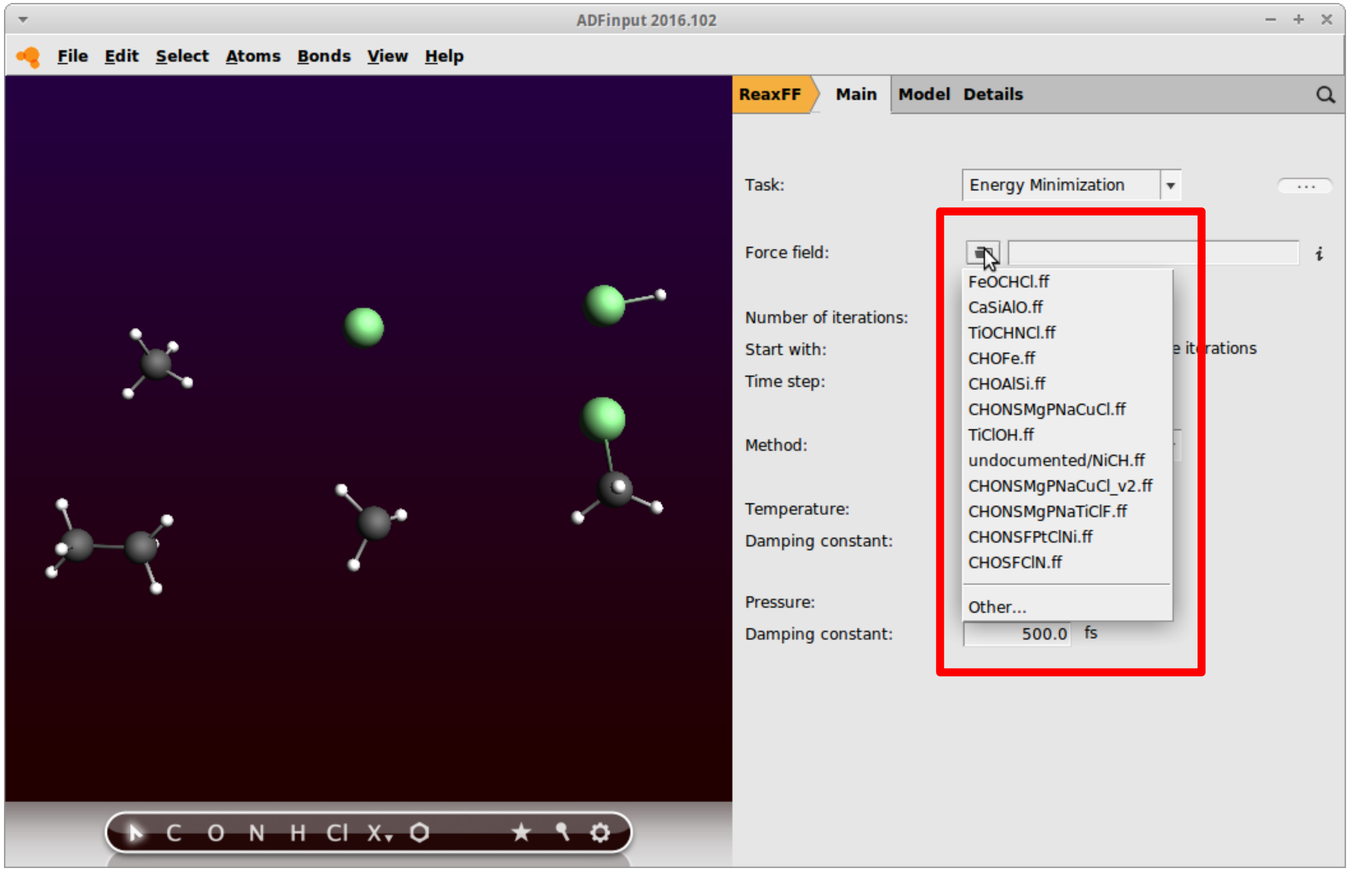
---

* when still in doubt: check the online references which conditions it was made for, e.g. ambient vs. combustion

MCFF: Hands-on

creating an initial guess for the force field, where to start?

Since there are no parameters available for Br (yet), we substitute Br with what we consider to be an element somewhat similar to Br: Chlorine.



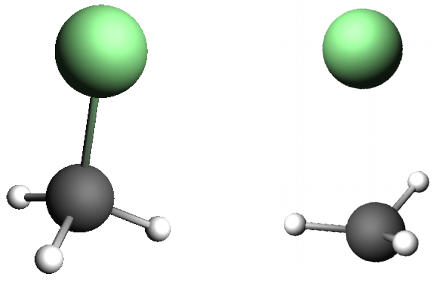
Task: Energy Minimization

Damping constant:

Damping constant: 500.0 fs

Run an Energy Minimization on CH_3Cl using these params:

- check which FFs do have all required parameters
- roughly check the geometries (compare against, e.g. MOPAC)



Which one is the best?

MCFF: Hands-on

creating an initial guess for the force field, where to start?

The best geometries came from the parameters in `CHOSFClN.ff`, even though it was originally made for explosives not even containing Cl. Note: Since this will only be the *initial guess* for the MCFF optimizer we just ignore our own concerns at this stage...

1. Copy the file `CHOSFClN.ff` from `atomicdata/ForceFields/ReaxFF` into a new directory and rename it to `CHBr.ff`

2. Edit it:

- Change the name in the first line, e.g. into 'Reactive MD-force field: C/H/Br - MCFF tutorial'
- Change the Element Symbol of Cl into Br
- Look at the elements listed under the atomic parameters:

```
 7    ! Nr of atoms; atomID;ro(sigma); Val;atom mass;Rvdw;Dij;gamma
            alfa;gamma(w);Val(angle);p(ovun5);n.u.;chiEEM;etaEEM;n.u.
            ro(pipi);p(lp2);Heat increment;p(boc4);p(boc3);p(boc5),n.u.;n.u.
            p(ovun2);p(val3);n.u.;Val(boc);p(val5);n.u.;n.u.;n.u.
C     1.3825   4.0000  12.0000   1.9133   0.1853   0.9000   1.1359   4.0000
      ...
H     0.7853   1.0000   1.0080   1.5904   0.0419   1.0206  -0.1000   1.0000
      ...
O     1.2477   2.0000  15.9990   1.9236   0.0904   1.0503   1.0863   6.0000
      ...
S     1.8328   2.0000  32.0600   1.8815   0.3236   0.7530   1.6468   6.0000
      ...
F     1.1846   1.0000  18.9984   1.7922   0.1267   0.5000  -0.1000   7.0000
      ...
Br    1.8600   1.0000  35.4500   1.9532   0.1081   0.5000  -1.0000   7.0000
      ...
N     1.6157   3.0000  14.0000   1.9376   0.1203   1.0000   1.2558   5.0000
```

3. Identify the numbers of the according elements:

here:

C ↔ 1
H ↔ 2
Br ↔ 6

e.g. H-Br bond:
2 6 ...[parameters]...

MCFF: Hands-on

creating an initial guess for the force field, atom parameters

Very few of the parameters are obvious and can be set directly. Some can be fairly guessed but most are starting with the according Cl-values.

1. Edit the atomic parameters of Br as follows:

```
          ro(sigma;   Val;  atom mass;  Rvdw;     Dij;      gamma;   ro(pi);   Val(e);
          Alfa;     gamma(w);Val(angle);p(ovun5);n.u.;    chiEEM;  etaEEM;    n.u.;
          ro(pipi;   p(lp2); Heat incr.;p(boc4);p(boc3);  p(boc5);  n.u.;     n.u.;
          p(ovun2;   p(val3;   n.u.;    Val(boc; p(val5;   n.u.;    n.u.;     n.u.;

       Br    1.7478    1.0000   79.9040    2.1280    0.1081    0.5000   -1.0000    7.0000
            10.4813   10.1330    1.0000    5.0000    0.0000    8.7500   12.0000    0.0000
            -1.0000    3.5750   28.0000    6.2043    5.2294    0.1542    0.8563    0.0000
           -10.2080    2.9867    1.0338    6.2998    2.5791    0.0000    0.0000    0.0000
```

Guess:

- `ro(sigma)` : sigma bond, covalent radius.
  → guess ~ *10%* shorter than tabulated.
- `Rvdw` : van der Waal radius :
  → guess ~ 12% larger than tabulated.
- `ro(pi)`: pi bond, covalent radius.
  → as for Br, there are none.
- `ro(pipi)`: double pi bond, covalent radius.
  → as for Br, there are none.

Known:

- **`Val`** : valence
- **`atom mass`** : atomic mass in u.
- **`Val(e)`** : number of valence electrons

Note:
The guesses are just ballpark figures derived from inspecting existing FFs and -well- sort of guessing. They should be worth trying, but they don't necessarily work.
Feel free to come up with a better guess...

MCFF: Hands-on

creating an initial guess for the force field, bonds

The parameters within the bonds section will mostly be kept at the values from the initial FF.

1. Edit the bonds containing Br and/or C,H in the bonds section as follows:

```
At1; at2;   De(sigma); De(pi);De(pipi);p(be1); p(bo5); 13corr;  n.u.; p(bo6)
             p(ovun1); p(be2); p(bo3); p(bo4);  n.u.;  p(bo1);  p(bo2 ; n.u.

 1     6    78.0000     0.0000   0.0000 -0.5450 -0.5000   1.0000   0.0000   0.7833
             5.4257    -0.2500  15.0000 1.0000 -0.0827   5.9023   0.0000   0.0000

 2     6    104.0000    0.0000   0.0000 -0.3793 -0.2000   0.0000  16.0000   1.2000
              1.8230   -0.2000  15.0000 1.0000 -0.1240   8.5000   0.0000   0.0000

 6     6    122.0000    0.0000    0.0000 0.6302 -0.3500   1.0000  25.0000   1.2000
             -0.2447   -0.2500  15.0000 1.0000 -0.0711   7.6505   0.0000   0.0000
```

Guess:

The parameters `De(sigma),De(pi),De(pipi)` are the bond dissociation energies. A tempting parameter to set, but less intuitive than one would think. The values given here (and in other FFs) are (in part significantly) bigger than those tabulated.

However, taking the reference value and adding 30% to `De(sigma)` does – at least in our case - yield a significantly better energy description:

```
De(sigma) C-Br  (Reference: ~ 60 kcal/mol) *  1.3 →  78.0000
De(sigma) Br-Br (Reference: ~ 94 kcal/mol) * 1.3  → 122.0000
De(sigma) H-Br  (Reference: ~ 80 kcal/mol) * 1.3 →  104.0000
```

MCFF: Hands-on

creating an initial guess for the force field, off-diagonal

The section off-diagonal contains bond and van der Waals parameters for atom pairs of different types.

IMPORTANT: The parameters put here have a *higher priority* than those given in the atomic section and will overwrite their according values.

```
 at1; at2;   Dij;    RvdW;    alfa;  ro(sigma); ro(pi); ro(pipi);

  1    6   0.1388  1.7970  11.9405   1.9900    -1.0000  -1.0000
  2    6   0.0583  1.6543  10.2027   1.4640    -1.0000  -1.0000
```

Known:

- **`ro(sigma)`**: sigma bond, covalent radius.
  → take the exact reference value
- **`ro(pi)`** : pi bond, covalent radius.
  → no pi bonds in our case
- **`ro(pipi)`** : double pi bond, covalent radius.
  → no double pi bonds in our case

Note:
Setting these will have visible impact on your force field, which you can easily verify by running a geometry optimization using your edited FF…
(reminder: run cleanreaxffforcefield.py first)

MCFF: Hands-on

creating an initial guess for the force field, angles

In the angles section we already know the equilibrium angle:

1. Edit the angles containing Br and/or C,H as follows:

```
at1;at2;at3;    Thetao,    o;      p(val1); p(val2); p(coa1);

2   1    6     71.4000   9.5717  1.1622     0.0000  0.1000  0.0000  1.0000
1   1    6     67.4000  24.7983  1.5638     0.0000  0.3494  0.0000  1.2226
```

Known:

- **Thetao:** 180° - equilibrium angle
  → set to 180 – [measured via GUI]
  for example: 108.6° measured via GUI <=> Thetao = 71,400 = 180 - 108.6

At this stage we have finished setting up the initial guess. For now we skip the sections torsions and hydrogen bonds, because:

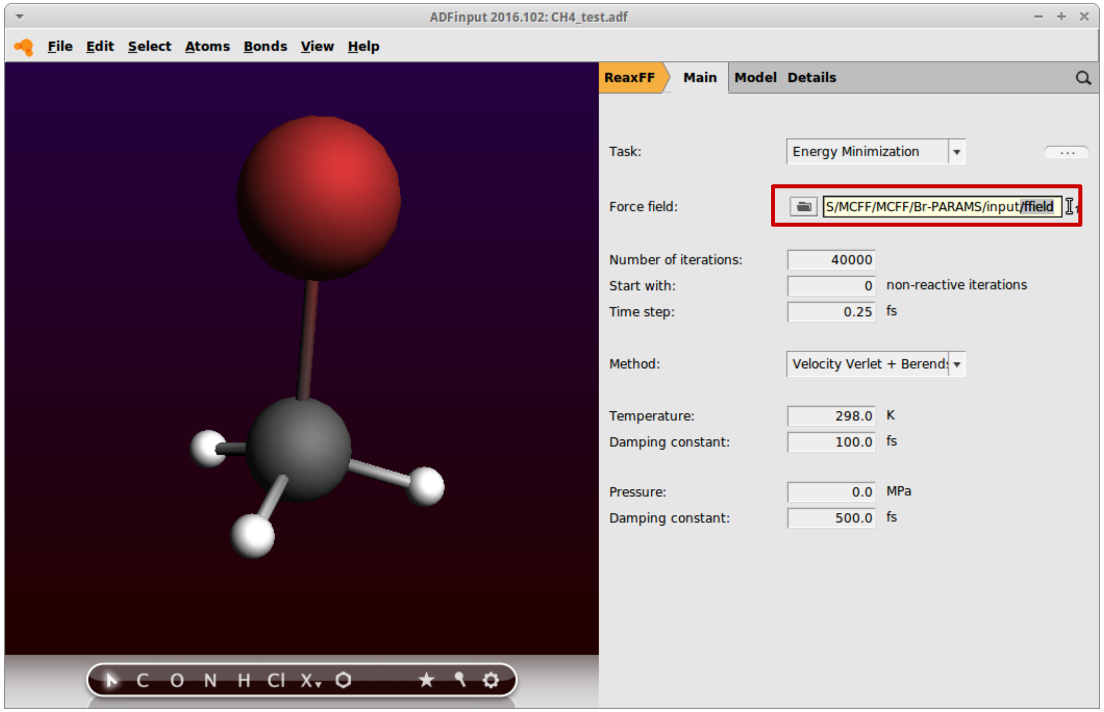
- The only torsion we will actually need for our reaction is the HCCH-torsion for ethane, but this should be covered by the existing parameters (more or less).

MCFF: Hands-on

creating an initial guess for the force field, first test

Let's test the new FF with CH_3Br:

- save your changes to CHBr.ff
- run the automatic formatting to create a correctly formatted `ffield`:
  `startpython cleanreaxffforcefield.py -i CHBr.ff -o ffield`
- open ADFinput
- select ReaxFF, draw CH_3Br
- click on the folder button next to 'Force field', navigate to your `ffield` and select it
- run an Energy minimization (and/or some dynamics if you want to)


ADFinput 2016.102: CH4_test.adf
File Edit Select Atoms Bonds View Help
ReaxFF
Main
Model
Details
C O N H Cl X

Task: Energy Minimization

Force field: S/MCFF/MCFF/Br-PARAMS/input/ffield

Number of iterations: 40000

Start with: 0 non-reactive iterations

Time step: 0.25 fs

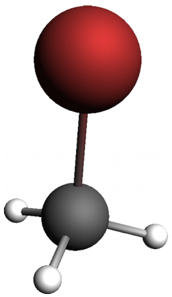
Method: Velocity Verlet + Berend:

Temperature: 298.0 K

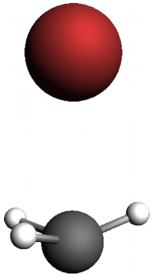
Damping constant: 100.0 fs

Pressure: 0.0 MPa

Damping constant: 500.0 fs



good guess!
→ continue



bad guess...
→ check FF

MCFF: Hands-on

creating the MCFF input from your initial guess

1. Create the parameter ranges, i.e. the ranges in which the MCFF optimizer will operate.

```
adfreport ffield -ffield-min > ffield_min
adfreport ffield -ffield-max > ffield_max
```

2. Create ffield_bool, that defines which parameters will be kept constant:

```
adfreport ffield -ffield-bool > ffield_bool
```

3. Edit only the Br-parameters in the `ffield_boolean` file:

   - set all known parameters and **n.u.** parameters (see above) to *fixed*, i.e. 0.000
   - set all others, including guessed parameters to *active*, i.e. 1.000

MCFF

or just use the ffield files in the folder "MCFF_INPUT_AND_SCRIPTS"

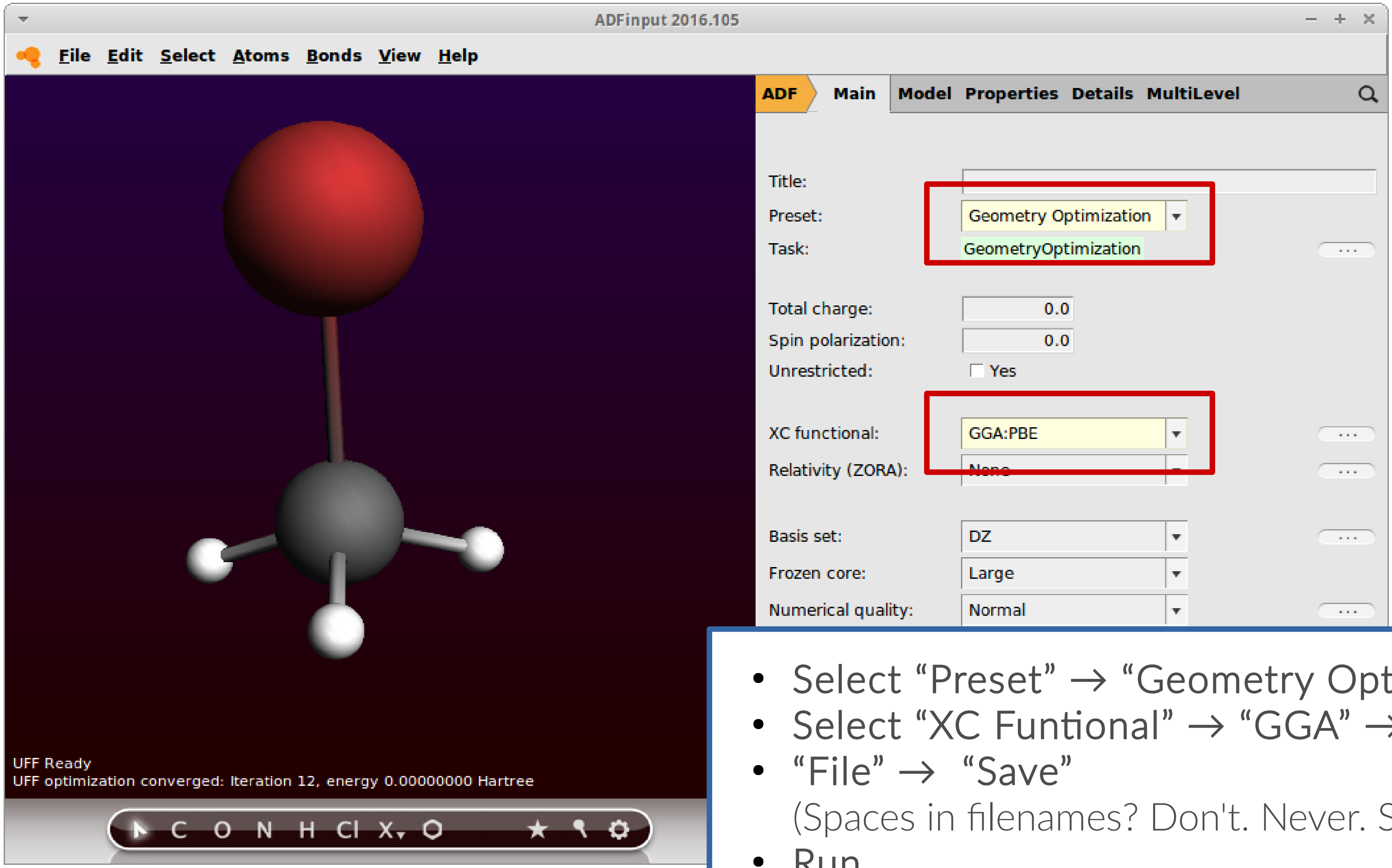
MCFF – Hands on

creating training data

MCFF: Hands-on

creating training data from geometry optimizations & single points

In the following we create a (rather minimal) training set. We focus on the species containing Br.



Title:
Preset: Geometry Optimization
Task: GeometryOptimization
Total charge: 0.0
Spin polarization: 0.0
Unrestricted: ☐ Yes
XC functional: GGA:PBE
Relativity (ZORA): None
Basis set: DZ
Frozen core: Large
Numerical quality: Normal

```
UFF Ready
UFF optimization converged: Iteration 12, energy 0.00000000 Hartree
```

- Select “Preset” → “Geometry Optimization”
- Select “XC Funtional” → “GGA” → “PBE”
- “File” → “Save”
  (Spaces in filenames? Don't. Never. Seriously.)
- Run
- Repeat for Br_2, Hbr

Using scalar ZORA would have created better references at no more expense. I (Ole) simply forgot.
For consistency within the supplied trainingdata, you should stick to the setting explained above.

MCFF: Hands-on

creating training data from geometry optimizations & single points

The training data can be extracted from ADF's TAPE21 files with adfreport

1. using the shell, navigate into the folder you ran the geometry optimizations in and type

```
adfreport [filename].t21 BGF > geometry
adfreport  geometry -rxtrainset > trainingset
```

2. repeat for every geometry optimization you ran (CH_3Br, Br_2, HBr)

3. use a text editor to merge the data into one `geo` and one `trainset.in` file:
   - the geometries can just be appended into the `geo` file with one blank line separating them
   - the entries in the `trainset.in` file should be grouped according to their type, e.g. collect all charges in one block:

```
CHARGE
 ... charges ...
ENDCHARGE
```

see format → trainset.in → geo

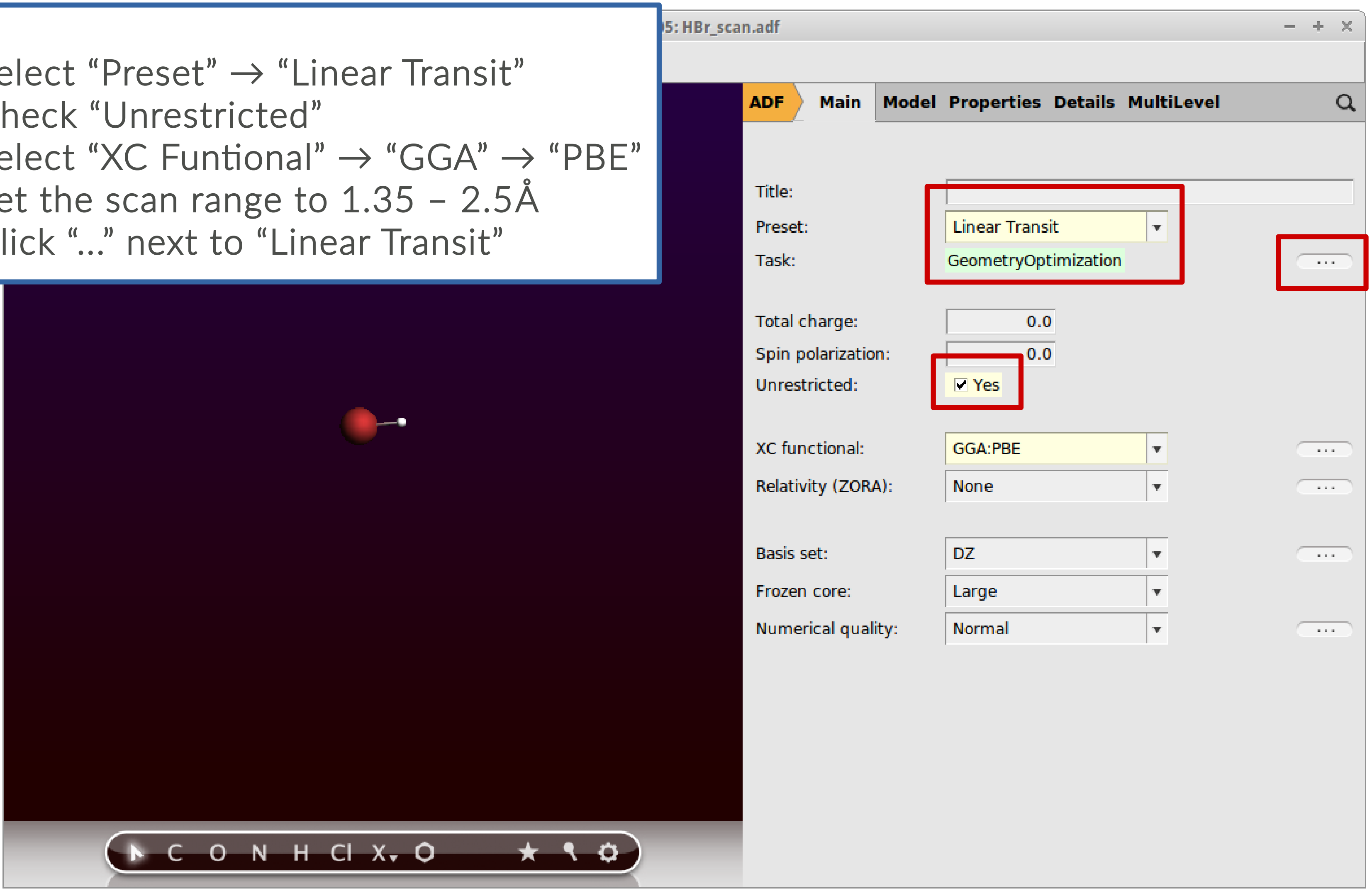
MCFF: Hands-on

creating training data from linear transits

The training data can also be extracted from linear transits (like bond/angle/torsion scans) :

1. use ADFinput to scan the bond length in HBr (PBE/DP, unrestricted)

- Select “Preset” → “Linear Transit”
- Check “Unrestricted”
- Select “XC Funtional” → “GGA” → “PBE”
- Set the scan range to 1.35 – 2.5Å
- Click “…” next to “Linear Transit”



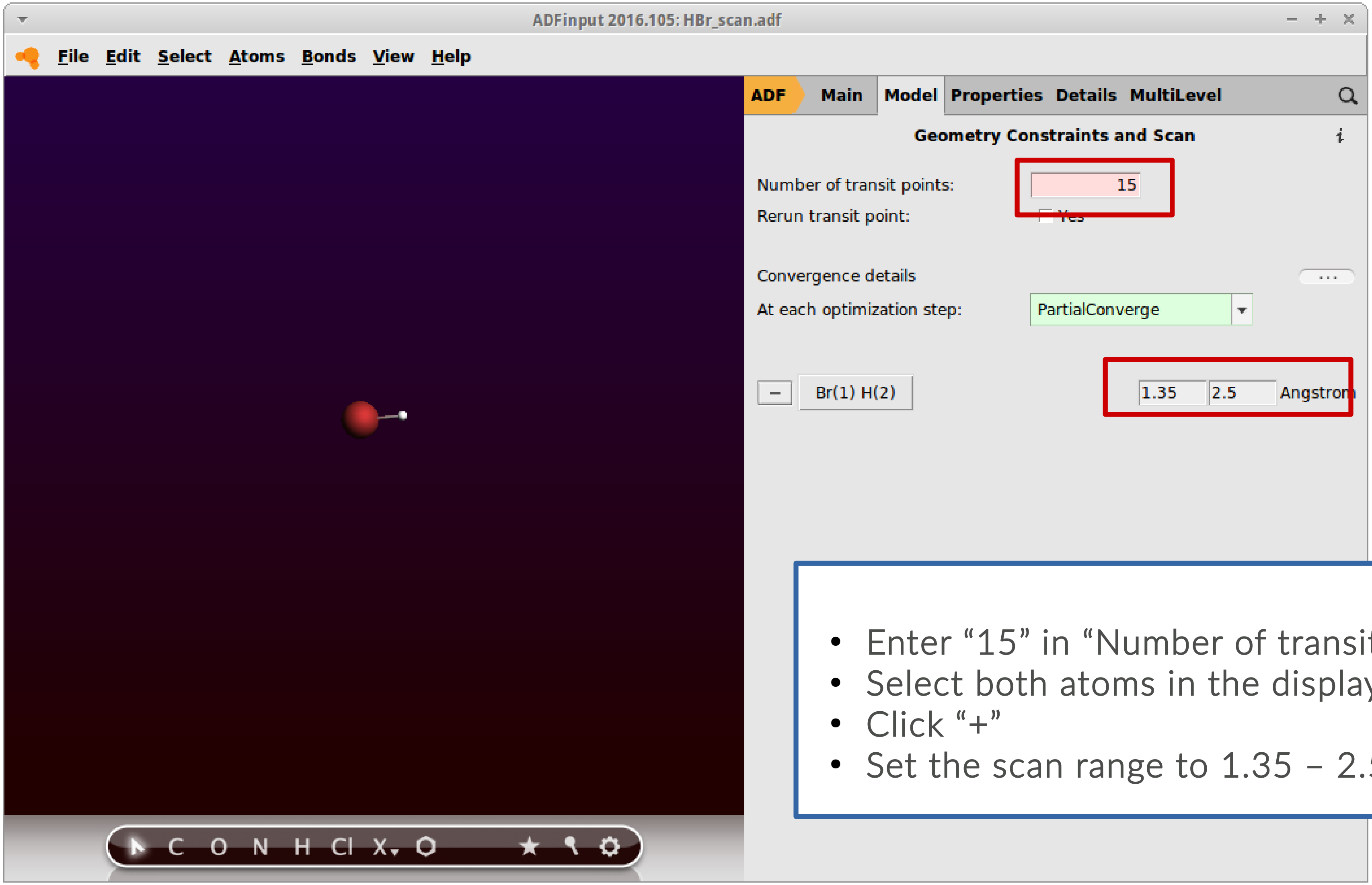
Title:
Preset: Linear Transit
Task: GeometryOptimization
Total charge: 0.0
Spin polarization: 0.0
Unrestricted: ☑ Yes
XC functional: GGA:PBE
Relativity (ZORA): None
Basis set: DZ
Frozen core: Large
Numerical quality: Normal

Note: A linear scan might not be the best approach to scan a diatomic, since every step is calculated twice. Here it serves the purpose of illustrating how to extract data from linear transits.

MCFF: Hands-on

creating training data from linear transits

Use ADFinput to set the scan range in the linear transit calculation:


ADFinput 2016.105: HBr_scan.adf
File Edit Select Atoms Bonds View Help
ADF Main Model Properties Details MultiLevel
Geometry Constraints and Scan
Convergence details
Br(1) H(2)
C O N H Cl X

Number of transit points: 15

Rerun transit point: ☐ Yes

At each optimization step: PartialConverge

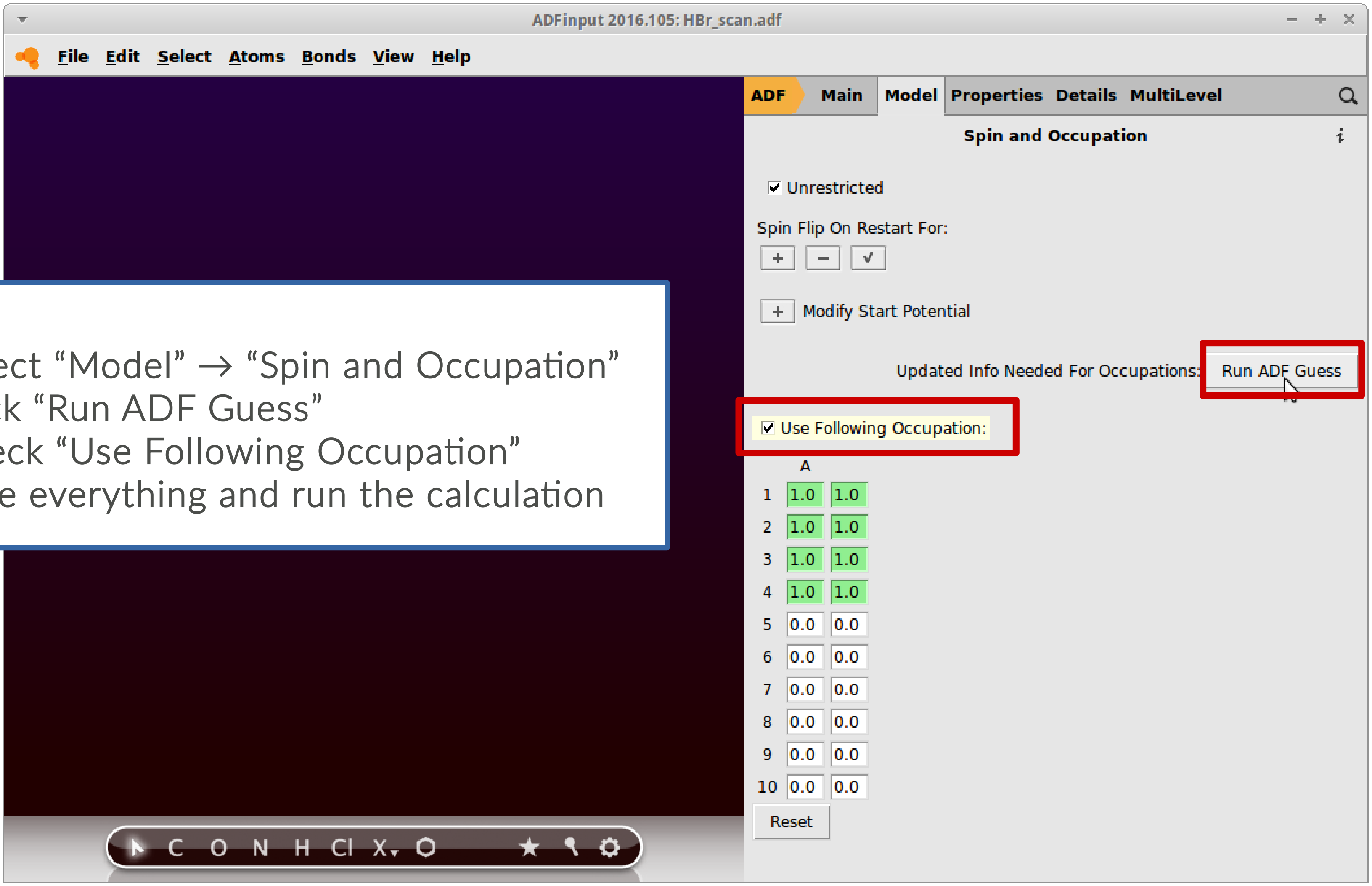
Br(1) H(2) 1.35 2.5 Angstrom

- Enter “15” in “Number of transit points”
- Select both atoms in the display window
- Click “+”
- Set the scan range to 1.35 – 2.5Å

MCFF: Hands-on

creating training data from linear transits

Use ADFinput and ADF to guess the initial Spin and Occupation...


ADFinput 2016.105: HBr_scan.adf
File Edit Select Atoms Bonds View Help
ADF Main Model Properties Details MultiLevel
Spin and Occupation
Spin Flip On Restart For:
+ − √
+ Modify Start Potential
Run ADF Guess
Reset
C O N H Cl X

☑ Unrestricted

Updated Info Needed For Occupations: Run ADF Guess

☑ Use Following Occupation:

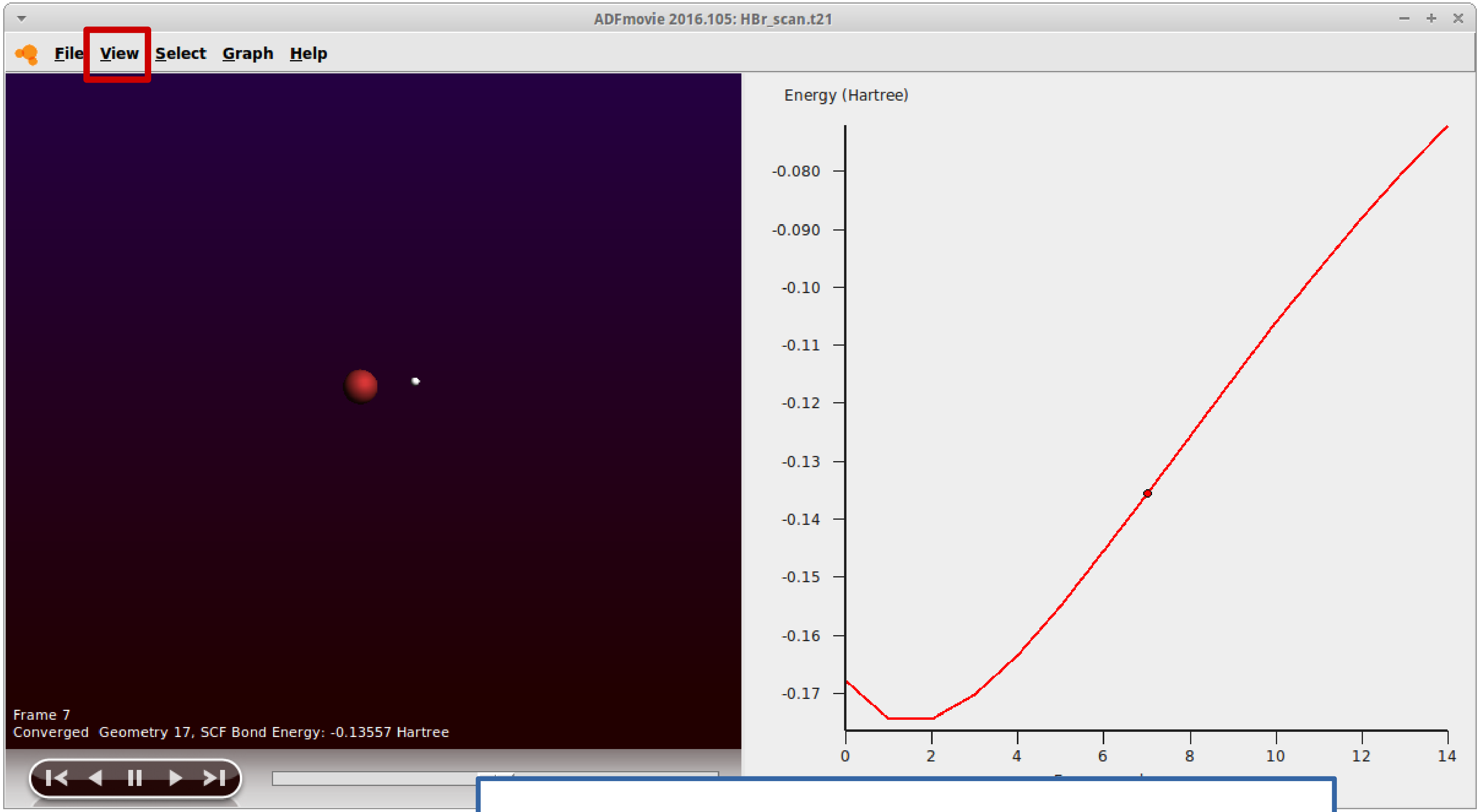
	A	
1	1.0	1.0
2	1.0	1.0
3	1.0	1.0
4	1.0	1.0
5	0.0	0.0
6	0.0	0.0
7	0.0	0.0
8	0.0	0.0
9	0.0	0.0
10	0.0	0.0

- Select “Model” → “Spin and Occupation”
- Click “Run ADF Guess”
- Check “Use Following Occupation”
- Save everything and run the calculation

MCFF: Hands-on

creating training data from linear transits

The energies and structures can be visualized with ADFmovie while the calculation is running



Frame 7
Converged Geometry 17, SCF Bond Energy: -0.13557 Hartree

- Select “View” → “Converged Geometries only”

MCFF: Hands-on

Running the MCFF optimizer

The training data can be extracted from linear transit TAPE21 files using a PYTHON script. Assuming the Python script and the TAPE21 file are in the same directory:

1. using the shell, navigate into the folder you ran the linear transit in and type

```
startpython LT_to_trainset.py [filename].t21
```

2. use a text editor to merge the data from this transit into the existing `geo` and `trainset.in` file
   - the geometries can just be appended into the `geo` file with one blank line separating them
   - the entries in the `trainset.in` file should be grouped according to their type, e.g. collect all charges in one block:

```
CHARGE
 ... charges ...
ENDCHARGE
```

see format → trainset.in → geo

MCFF – Hands on

Running the MCFF optimizer

MCFF: Hands-on

Running the MCFF optimizer

MCFF inputfiles and scripts are found in the folder 'MCFF_INPUT_AND_SCRIPTS'

Running the MCFF optimizer itself is simple:

- Edit the settings you want to use in mcff.run and execute it

  `./mcff.run` or ./mcff.run &

  the latter command will start the MCFF run as a background process

- The MCFF run can be stopped by editing the first line in the file `istop`:

  ```
  2 !Key: 1 – tell program to use the values below, 2 – stop
  ```

  save this change. The optimizer will stop after the current step is finished

- use the command `tail -f [filename]` to monitor the progress of the optimization

  - a breakdown of the error function is written to the file `fort.99`

  - the logfile of the optimizer is called called `MCFFOptimizer.log`

- for plotting you can run the following script to produce the file `FITNESS_MCFF`

  ```
  startpython get_fitness.py
  ```

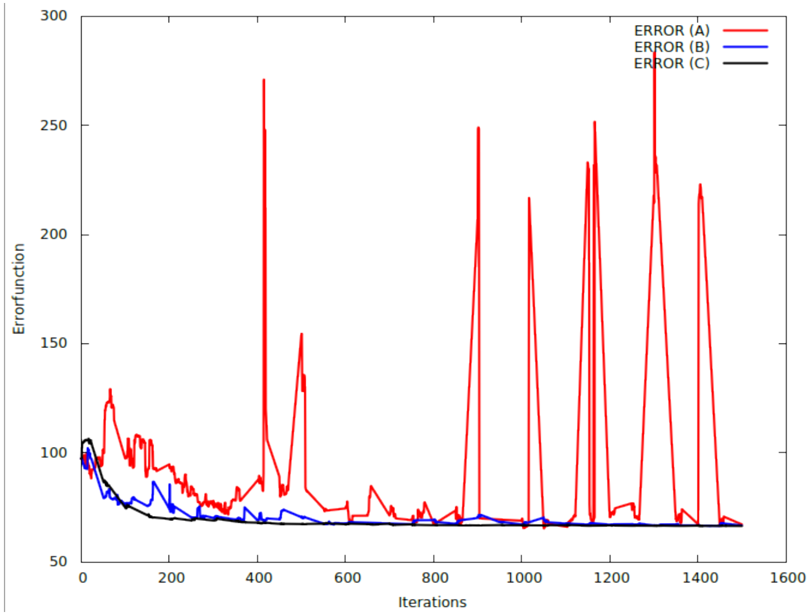
MCFF: Hands-on

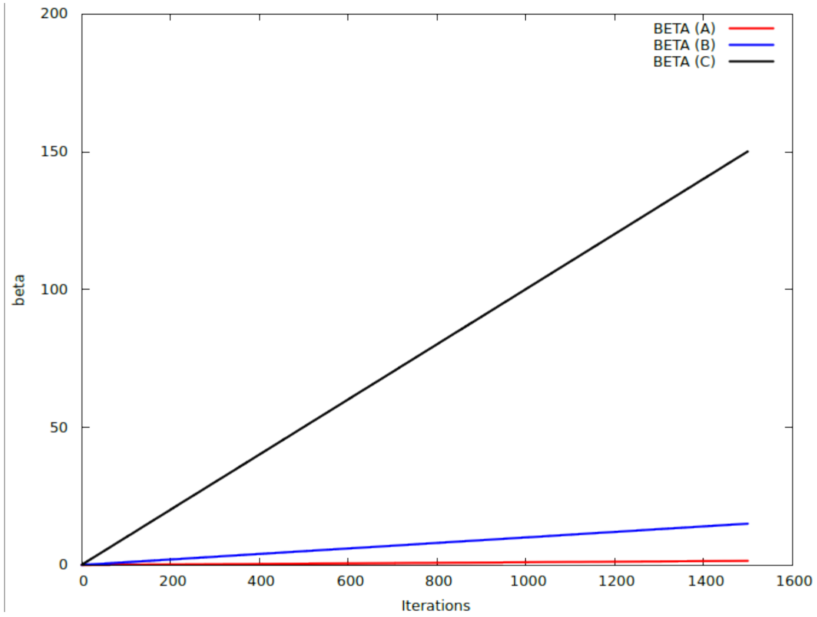
Running the MCFF optimizer: Exploring the Simulated Annealing settings

Start by running 3 optimizations with 1500 optimization steps each (`1500 mcffit`) using your geometry optimization training data. Use the following annealing settings

```
a) 0.0010   mcbeta        b)  0.0100  mcbeta        c)  0.1000  mcbeta
   0.0010   mcdbet            0.0100  mcdbet            0.1000  mcdbet
   1.0000   mcbsca            1.0000  mcbsca            1.0000  mcbsca
     ...                        ...
```

inspect the Error function in `fort.99` and plot the fitness functions:

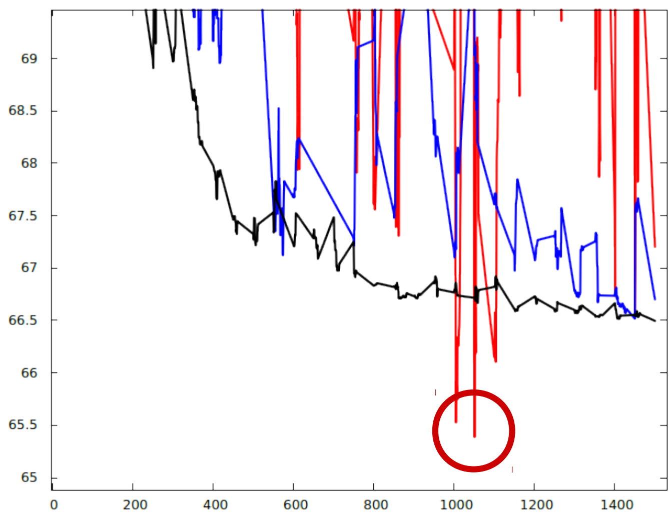


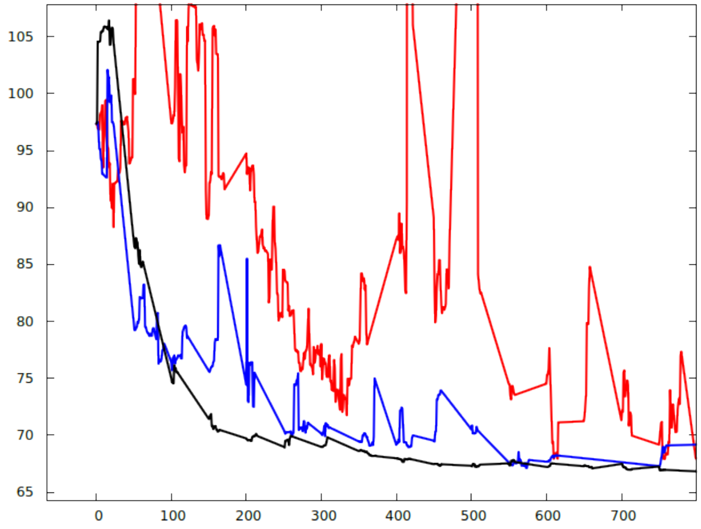


When does the cooling take over the exploration of parameters pace?
What are the obtained errors of each run?

MCFF: Hands-on

Running the MCFF optimizer: Exploring the Simulated Annealing settings





In this case setting a) with the highest initial temperature and the slowest cooling rate resulted in both the worst and the best reproduction of the training data.

However, the simulated annealing is far from being completed for setting a), which you can see when comparing the errors of all settings.

When trying different annealing settings you should try to balance the exploration and cooling periods over the amount of iterations you want to run → both are important.

For the current setup these settings gave the best balance (so far)

`0.0001 mcbeta`
`0.0001 mcdbet`
`0.9915 mcbsca` // divide beta by this value at every step

Keep in mind that these values depend on the problem at hand.

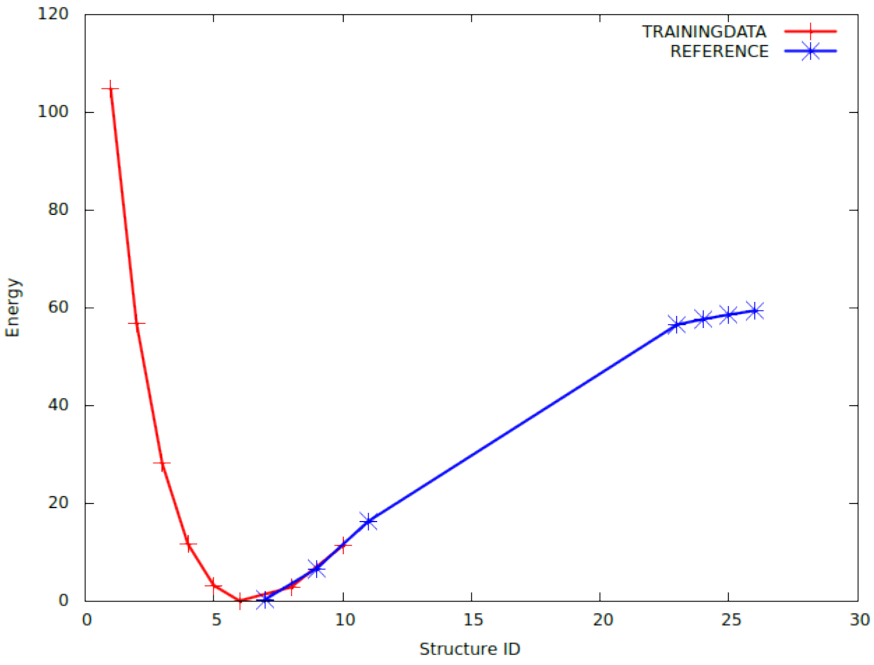
Ole's rule of thumb: if beta exceeds a value of 1000 you can usually stop the calculations.

MCFF: Hands-on

Running the MCFF optimizer: Smaller Error – better force field?

In the following tutorial we inspect the effect of overfitting, i.e. we test the predictive power of an optimized parameterset against a reference.

The training sets in 'OVERFITTING' and 'REFERENCE' are taken from a C-Br linear transit of CH_3Br splitting the linear transit data:



Run the following calculations:

- Run 500 MCFF optimizations with the training data
- Copy the file `ffield_best` into the `REFERENCE` folder: `cp ffield_best ./REFERENCE/ffield`
- Run 0 steps MCFF optimizations on the reference (just to calculate the Errorfunction)

```
cd REFERENCE
./mcff.run
```

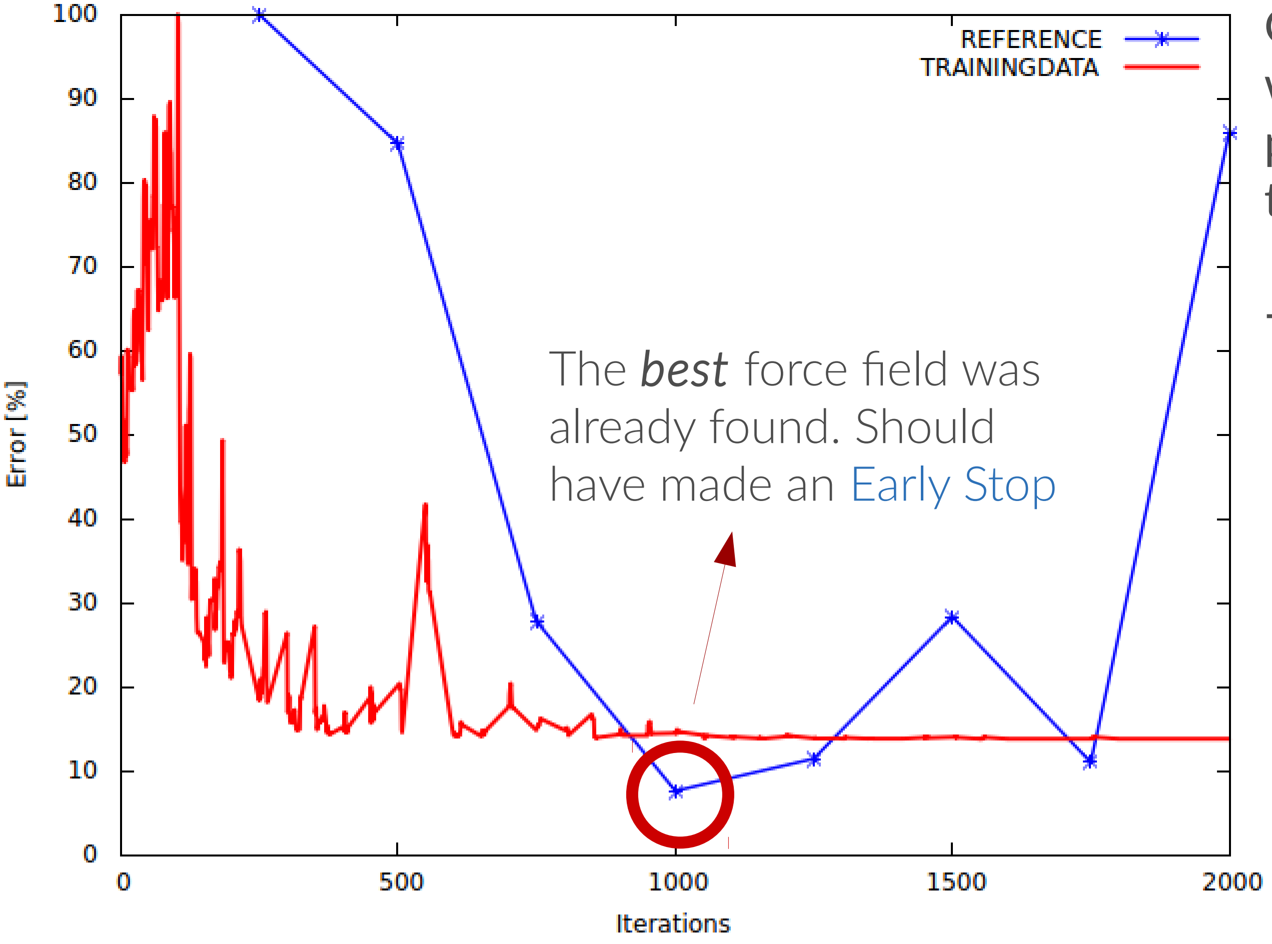
Repeat these steps with 1000 and 1500 steps of MCFF optimization. Compare the resulting Errors of both sets. **Does overfitting show?**

MCFF

The input files for this part are found in the folder 'OVERFITTING'.

MCFF: Hands-on

Running the MCFF optimizer: Smaller Error – better force field?



The ***best*** force field was already found. Should have made an Early Stop

Overfitting shows from on step 1000, when the predictive capabilites of our parameterset starts worsening after the initial improvements.

→ a smaller value of the errorfunction does not (always) mean we have created a better force field.

Note: the training set is somewhat constructed with the aim of drastic overfitting in mind, but it *is* a known problem, actively adressed in the *Machine Learning* and *Statistical Models* communities.

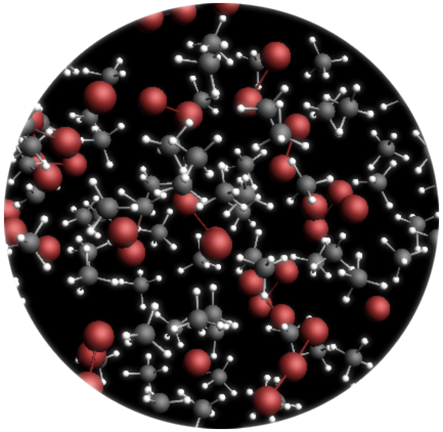
Always check your optimized parameters against a reference.

MCFF: Hands-on

Running the MCFF optimizer -serious optimization

A more decent – still rather small – training set featuring more linear transit data is provided for this tutorial. Following the strategies outlined above:

- Run an MCFF optimization with varying amounts of iterations (ideally up to 10.000)
- Check your force fields against the provided references (stop early enough)
- Check your force fields by running geometry optimizations with ReaxFF
- Run reactive MD simulations with ReaxFF
  - create a periodic box with Methane (see tutorial)
  - add some Br atoms and/or Br_2
- Extend the training set and references, missing are:
  - interactions between molecules
  - transition states
  - methane, methyl, ethane
  -…



- Good practice: Remove the unused atom types from your force field before sharing it.

- Enjoy your new force field!
  You are amongst the first people ever to have run a ReaxFF simulation with Bromine… :-)

MCFF The input files for this part are found in the folder 'CHBr_Opt'.

SCM
Software for
Chemistry &
Materials

contact us:

Licenses	**license@scm.com**
General information	**info@scm.com**
User support	**support@scm.com**