# Quickstart

1 **Setup and Installation**
   1.1 Getting the Program .................................................. 3
   1.2 Setting up **xtb** ....................................................... 4
      1.2.1 Parallelisation .................................................. 4
      1.2.2 Environment Variables for **xtb** ............................... 5
      1.2.3 Configuration Script ............................................ 5
   1.3 Getting Help from **xtb** ........................................... 5
      1.3.1 The Verbose Mode .............................................. 6
   1.4 Using xTB with Orca ................................................ 6

2 **Quickstart into Production**
   2.1 Singlepoint Calculations ........................................ 7
   2.2 Geometry Optimizations ........................................... 8
   2.3 Characterisation of Stationary Points ........................... 9
      2.3.1 Dealing with Small Imaginary Frequencies .................. 10

3 **Commandline Usage**
   3.1 RunTypes ............................................................ 11
   3.2 Options ............................................................... 14

4 **Detailed Input**
   4.1 Fixing, Constraining and Confining .............................. 16
      4.1.1 Exact Fixing .................................................... 16
      4.1.2 Constraining Potentials ..................................... 16
      4.1.3 Confining in a Cavity ....................................... 17
   4.2 Absolute Control .................................................. 18
      4.2.1 Global Configuration File .................................. 19

5 **Singlepoint Calculations**
   5.1 Input ................................................................. 21
   5.2 Charge and Multiplicity .......................................... 22
   5.3 Accuracy and Iterations ........................................... 24
      5.3.1 Accuracy ....................................................... 24
      5.3.2 Iterations ..................................................... 24
   5.4 Fermi-smearing ..................................................... 24
   5.5 Vertical Ionization Potentials and Electron Affinities ........ 25
   5.6 Global Electrophilicity Index ................................... 27
5.7  Fukui Index ................................................................. 27
    5.7.1  Example: BF$_3$ ...................................................... 28

6  Properties ................................................................. 29
    6.1  General printout .................................................... 29
    6.1.1  GFN1-xTB .......................................................... 30
    6.1.2  GFN2-xTB .......................................................... 31
    6.2  Density Properties ................................................ 31
    6.2.1  Cube Files ........................................................ 31
    6.2.2  Density and Spin-Density .................................... 32
    6.2.3  Fractional Occupation Density (FOD) calculation .......... 32
    6.3  Redirecting Property Printout .................................. 33
    6.4  Machine Readable Data Dump ................................... 33

7  Geometry Optimization ................................................ 35
    7.1  Optimization levels ............................................... 35
    7.2  Running a geometry optimization ................................ 36
    7.2.1  Example 1: ethyne ............................................. 36
    7.2.2  Example 2: cyclopentadienyl anion ......................... 38
    7.2.3  Example 3: $p$-benzyne in toluene ......................... 39
    7.3  Convergence problems ............................................ 40
    7.4  Optimization Engines ............................................ 41

8  Exploration of the potential energy surface (PES) ................. 43
    8.1  Input ............................................................... 43
    8.2  Dihedral angle scan .............................................. 44
        8.2.1  Ethane ......................................................... 44
        8.2.2  1-Bromo-2-chloroethane .................................. 46
    8.3  Angle and Distance scan ....................................... 48
        8.3.1  Ammonia ....................................................... 48

9  Implicit Solvation ..................................................... 51
    9.1  General command-line control .................................. 51
    9.2  Parameterized Solvents ......................................... 52
    9.3  Available Grids .................................................. 53
    9.4  Reference States ................................................ 53
    9.5  Extended Functionality ......................................... 53
        9.5.1  Solvent Accessable Surface Area ......................... 53

10 Calculation of Vibrational Frequencies .............................. 55
    10.1  Performing simple Vibrational Frequency calculations .... 55
    10.2  Calculation of thermochemical properties .................. 56
    10.3  Dealing with imaginary modes and non-minimum structures . 58
    10.4  Advanced options ............................................... 58

11 Molecular Dynamics Simulations ..................................... 59
    11.1  General command-line control ................................ 59
    11.2  Parameters ....................................................... 60
    11.3  MD specific Files ............................................... 60
        11.3.1  Restart ...................................................... 60
        11.3.2  Example/Case study ...................................... 61

12 Meta-Dynamics Simulations ........................................... 63
    12.1  General MTD Setup ............................................. 63
    12.1.1  MTD specific Files ......................................... 64
This user guide focuses on the semiempirical quantum mechanical methods GFNn-XTB, their descendants, and corresponding composite schemes as implemented in the xtb (extended tight binding) program package.

We provide a number of detailed guides dealing with common tasks that can be performed easily with the xtb program. All guides are usually structured the same way, starting with some simple examples using only the commandline and the default settings followed by a trouble shooting section. Detailed inputs are provided in a ready to use fashion to solve some more special but still common tasks with xtb together with some insights into the theory used behind the scenes.
CHAPTER 1

Setup and Installation

This guide deals with the general setup and local installation of the \texttt{xtb} program.

\begin{table}
\begin{tabular}{l}
\hline
\texttt{bin/xtb} \\
\texttt{lib/libxtb.so} \rightarrow \texttt{libxtb.so.6} \\
\texttt{lib/libxtb.so.6} \rightarrow \texttt{libxtb.so.6.2} \\
\texttt{lib/libxtb.so.6.2} \\
\texttt{include/xtb.h} \\
\texttt{python/xtb.py}
\hline
\end{tabular}
\end{table}

1.1 Getting the Program

The \texttt{xtb} program is available for academic use free of charge on request from Stefan Grimme at the \texttt{xtb-mailing list}. It usually comes as a tarball with following content:
The binary is usually compiled with the Intel Fortran compiler and statically linked against Intel’s Math Kernel Library (Intel MKL). Newer versions of xtb (6.2 and newer) additionally include a shared library, the header specification of the C-API and a Python wrapper to use the API within the Atomic Simulation Environment (ASE).

First check the version by

```bash
> xtb --version
```

This should print some fancy banner, the version number, say 6.1 RC1, the last programmer worked on the project (usually SAW, meaning Sebastian Ehlert) and the date the program was last compiled and tested by this programmer, as YYMMDD.

### 1.2 Setting up xtb

This section will give you the basic information you need to know about the xtb program. Some of the steps are elemental for your calculation to succeed, so please consider to follow my instructions carefully.

Some part of the xtb program can be quite wasteful with stack memory, to avoid stack overflows when calculating large molecules, you should unlimit the system stack, *e.g.* with bash by

```bash
> ulimit -s unlimited
```

Note that the memory management of xtb is constantly improved to avoid using large amounts of stack memory, but to be on the save side include this option for production runs.

#### 1.2.1 Parallelisation

The xtb program uses OMP parallelisation, to calculate larger systems an appropriate OMP stacksize must be provided, chose a reasonable large number by

```bash
> export OMP_STACKSIZE=1G
```

To distribute the number of threads reasonable in the OMP section it is recommended to use

```bash
> export OMP_NUM_THREADS=<ncores>,1
```

You might want to deactivate nested OMP constructs by

```bash
> export OMP_MAX_ACTIVE_LEVELS=1
```
1.2.2 Environment Variables for xtbd

A number of environment variables is used by xtbd to perform calculations. Please set the XTBPATH variable to include all locations where you store information relevant for your xtbd calculation, like configuration files and parameter files. The present working directory is implicitly included for most files that are searched in the XTBPATH.

The old XTBHOME variable is used if you have not set the XTBPATH variable and is used in the same manner. xtbd will print the values of XTBPATH and XTBHOME at the beginning of each calculation if set to verbose mode.

An easy way to setup the environment variables is to use the distributed Config_xtb_env. For a bash shell this might be done locally for one session by sourcing the Config_xtb_env.bash script. To use this setup in every session include

```bash
source $XTBHOME/Config_xtb_env.bash
```

in your .bashrc (requires that XTBHOME is set to the appropriate directory).

1.2.3 Configuration Script

The “configuration” scripts Config_xtb_env.* hardly deserve to be called that way, in fact they contains the lines you would manually write to your .bashrc or .cshrc if you would “install” xtbd locally by hand. If you prefer to do it by hand or differently, just ignore the script.

Just take a look into one, there is some neat trick included found in a Turbomole “configuration” script to find the location of the script and the most probable location of the content of the tarball, but that’s it. Here is the contents of the one shipped with 6.2 for quick reference:

```bash
#!/bin/bash
# run this script to set up a xtbd environment
# requirements: $XTBHOME is set to `pwd'
if [ -z "${XTBHOME}" ]; then
    XTBHOME="$(cd -P "${dirhome "${BASH_SOURCE[0]}")" & & pwd)"
fi

# set up path for xtbd, using the xtbd directory and the users home directory
XTBPATH=${XTBHOME}:${HOME}

# to include the documentation we include our man pages in the users manpath
MANPATH=${MANPATH}:${XTBHOME}/man

# finally we have to make the binaries and scripts accessible
PATH=${PATH}:${XTBHOME}/bin:${XTBHOME}/python
LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${XTBHOME}/lib
PYTHONPATH=${PYTHONPATH}:${XTBHOME}/python

export PATH XTBPATH MANPATH LD_LIBRARY_PATH PYTHONPATH
```

It will set XTBHOME to the location of the script if you have not set it already and just assumes that XTBHOME contains the content of shipped tarball, then it will append the directories bin/ and python/ to your PATH variable, man/ to your MANPATH, lib/ to your LD_LIBRARY_PATH and python/ to your PYTHONPATH.

1.3 Getting Help from xtbd

Beside this manual you can check the in-program help by
> xtb --help

Unfortunately, this might be outdated, therefore, you should refer to the man-pages distributed with the xtb program. Please check for the man-pages of xtb(1) and xcontrol(7). There is also an online documentation, but you already now that one, of course.

### 1.3.1 The Verbose Mode

If you think some information is missing in your calculation you can switch to the verbose mode by using `--verbose` in the command line arguments. This will increase the print level almost everywhere in the xtb program, also the input parser will print a lot of information that might be interesting for your current calculation.

Overall this can be an awful lot of information, so it is not recommended as a default option.

### 1.4 Using xTB with Orca

Orca 4.2 implements support for xTB calculations using an IO based interface calling the xtb binary and parsing its output.

The binaries of Orca will call an executable called `otool_xtb`, which should be placed in the directory containing the Orca binaries. We recommend to create a symbolic link to your local xtb binary by

```bash
> ln -s $(which xtb) otool_xtb
```

You can invoke xTB calculations in Orca by using one of the simple keywords

```bash
! XTB1 # for GFN1-xTB
! XTB2 # for GFN2-xTB
```

in your Orca input file, for more details refer to the Orca manual.

Orca will communicate with xtb mainly by using commandline arguments, requesting singlepoint calculations and parsing the total energy and gradient from the program output.

Of course you should setup the xtb related environment variables, such that xtb can find its parameter files and configuration files. The .xtbrc is still read if it is contained in XTBPATH and can be used to change the behaviour of xTB calculations in Orca, e.g. for setting the electronic temperature.
This chapter should serve as a quickstart tutorial guiding you through your first calculation employing the xTB methods. As an example, the equilibrium geometry of a water molecule is calculated. The description here is based on xtb version 6.1 RC2.

2.1 Singlepoint Calculations

Independent of all other commands, there will always be a singlepoint calculation carried out at the very beginning. To calculate something xtb needs information about the molecular geometry and the kind of atoms present.

The default input format is either the Turbomole coordinate file as a $coord data group starting in the very first line
Any valid Xmol file (xtb will actually count the lines and double check the number of atoms specified), here the suffix .xyz is optional since xtb will auto detect the file type. xtb also supports structure-data files (sdf), if the corresponding suffix is encountered.

By default xtb will search for .CHRG and .UHF files and obtain from these the molecular charge and the number of unpaired electrons, respectively. The molecular charge can also be specified by

```
> xtb molecule.xyz --chrg +1
```

which is equivalent to

```
> echo +1 > .CHRG && xtb molecule.xyz
```

This also works for the unpaired electrons as in

```
> xtb --uhf 2 input.sdf
```

To select the parametrization of the xTB method you can currently choose from three different geometry, frequency and non-covalent interactions (GFN) parametrization, which differ mostly in the cost–accuracy ratio,

```
> xtb --gfn 2 coord
```

to choose GFN2-xTB, which is also the default parametrization. Also available are GFN1-xTB, and GFN0-xTB.

Sometimes you might face difficulties converging the self consistent charge iterations, in this case it is usually a good idea to increase the electronic temperature and to restart at normal temperature

```
> xtb --etemp 1000.0 coord && xtb --restart coord
```

### 2.2 Geometry Optimizations

The main purpose of the xTB methods is to provide good geometries, so the xtb comes with a build-in geometry optimizer, which usually does a decent job. It is invoked by

```
> xtb coord --opt
> ls
coord xtbopt.coord xtbopt.log ...
```

The optimized coordinates is written to a new file (xtbopt.coord), which is in the same format as the input geometry. You can view the geometry optimization by opening the xtbopt.log with your favorite molecule viewer.
The log-file is in Xmol format and contains the current total energy and the gradient norm in the comment line, 
gmolden usually works fine for this.

A successful geometry optimization will print somewhere along the lines

```
*** GEOMETRY OPTIMIZATION CONVERGED AFTER 43 ITERATIONS ***

<table>
<thead>
<tr>
<th>total energy gain :</th>
<th>-0.0094907 Eh</th>
<th>-5.9555 kcal/mol</th>
</tr>
</thead>
<tbody>
<tr>
<td>total RMSD          :</td>
<td>0.7677834 a0</td>
<td>0.4063 Å</td>
</tr>
</tbody>
</table>
```

after finishing the optimization procedures, while in all other cases that not exit in error

```
*** FAILED TO CONVERGE GEOMETRY OPTIMIZATION IN 500 ITERATIONS ***
```

will be printed, additionally a NOT_CONVERGED file is created in the working directory, which might become handy for bulk jobs.

To get a geometry optimization to converge can be a hard job, usually the xTB methods can repair a lot, you might want to start from GFN0-xTB which does not have convergence issues and than improve with GFN2-xTB. Maybe you have to adjust the geometry by hand again, if even this fails.

xtb offers eight predefined levels for the geometry optimization, which can be chosen by appending the level to the optimization flag as in

```
> xtb coord --opt tight
```

The thresholds defined by simple keywords are given here

<table>
<thead>
<tr>
<th>level</th>
<th>Econv/Eh</th>
<th>Gconv/Eh-\alpha^1</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>crude</td>
<td>5 \times 10^4</td>
<td>1 \times 10^2</td>
<td>3.00</td>
</tr>
<tr>
<td>sloppy</td>
<td>1 \times 10^4</td>
<td>6 \times 10^3</td>
<td>3.00</td>
</tr>
<tr>
<td>loose</td>
<td>5 \times 10^3</td>
<td>4 \times 10^3</td>
<td>2.00</td>
</tr>
<tr>
<td>lax</td>
<td>2 \times 10^3</td>
<td>2 \times 10^3</td>
<td>2.00</td>
</tr>
<tr>
<td>normal</td>
<td>5 \times 10^3</td>
<td>1 \times 10^3</td>
<td>1.00</td>
</tr>
<tr>
<td>tight</td>
<td>1 \times 10^6</td>
<td>8 \times 10^4</td>
<td>0.20</td>
</tr>
<tr>
<td>vtight</td>
<td>1 \times 10^6</td>
<td>2 \times 10^4</td>
<td>0.05</td>
</tr>
<tr>
<td>extreme</td>
<td>5 \times 10^9</td>
<td>5 \times 10^9</td>
<td>0.01</td>
</tr>
</tbody>
</table>

The energy convergence (Econv) is the allowed change in the total energy at convergence, while the gradient convergence (Gconv) is the allowed change in the gradient norm at convergence. The accuracy is handed to the singlepoint calculations for integral cutoffs and self consistent field convergence criteria and is adjusted to fit the geometry convergence thresholds automatically.

The xTB methods are completely analytical, so you can in principle converge your results down to machine precision. Converging it down to the lower limit is more a development feature than a real life application but always possible.

### 2.3 Characterisation of Stationary Points

In xtb second derivatives are implemented by finite differences methods (numerical second derivatives). Normally you want to calculate the Hessian directly after a successful geometry optimization, this is done by using

```
> xtb coord --ohess
```
For the calculation on the input geometry use \texttt{--hess} instead.

### 2.3.1 Dealing with Small Imaginary Frequencies

For small imaginary modes \texttt{xtb} offers an automatic distortion feature of these modes, say you have optimized a geometry and performed a frequency calculation which leads to an imaginary frequency of 14 wavenumbers:

```plaintext
> xtb coord --ohess
...

<table>
<thead>
<tr>
<th>Frequency Printout</th>
</tr>
</thead>
<tbody>
<tr>
<td>projected vibrational frequencies (cm(^{-1}))</td>
</tr>
<tr>
<td>eigval : -0.00 -0.00 0.00 0.00 0.00 0.00</td>
</tr>
<tr>
<td>eigval : -14.26 8.12 9.26 12.09 15.85 17.73</td>
</tr>
<tr>
<td>eigval : 19.45 28.85 39.18 41.30 64.61 71.84</td>
</tr>
</tbody>
</table>
...

imag cut-off (cm\(^{-1}\)) : 5.00
found 1 significant imaginary frequency
writing imag mode distorted coords to <xtbhess.coord>
for further optimization.
...
```

In this case \texttt{xtb} will generate a distorted structure, you can continue to optimize with

```plaintext
> xtb xtbhess.coord --ohess
...

<table>
<thead>
<tr>
<th>Frequency Printout</th>
</tr>
</thead>
<tbody>
<tr>
<td>projected vibrational frequencies (cm(^{-1}))</td>
</tr>
<tr>
<td>eigval : -0.00 -0.00 -0.00 -0.00 0.00 0.00</td>
</tr>
<tr>
<td>eigval : 2.02 7.99 10.10 12.08 16.16 18.57</td>
</tr>
<tr>
<td>eigval : 23.88 28.93 38.35 42.18 64.86 73.76</td>
</tr>
</tbody>
</table>
...
```

The optimization will only take a few steps and the artificial imaginary frequency is gone after checking the frequency calculation.
For *xtb* we usually enjoy to build our workflows via commandline, so most tasks can be performed without ever writing any kind of input file (except for the geometry input, of course).

**Note:** If you need more control you should resort to the *Detailed Input* file.

### 3.1 Runtypes

The most basic flags used in *xtb* are the runtypes. We have a bunch of them, but many of the elaborate composite runtypes are constructed from four basic runtypes: singlepoint (SP), geometry optimization (ANCopt), frequency calculation (Hessian) and molecular dynamics (MD). Every calculation performs the basic setup and at some (reasonable) point a property calculation.

**Singlepoint**

- **flag** `--scc`
- **description** just self-consistent charge (SCC) calculation. See *Singlepoint Calculations* for details.
- **queue** setup, SP, properties

**Gradient**

- **flag** `--grad`
**description**  self-consistent charge (SCC) calculation, afterwards energy and gradient will be printed in a Turbomole readable format

**queue**  setup, SP, properties

**Vertical IP**

**flag**  --vip

**description**  vertical ionisation potential (IP), calculates SCC on input structure and than removes an electron to perform another SCC calculation.

**queue**  setup, SP, SP, properties

**Vertical EA**

**flag**  --vea

**description**  vertical electron affinity (EA), calculates SCC on input structure and than adds an electron to perform another SCC calculation.

**queue**  setup, SP, SP, properties

**Vertical IP and EA**

**flag**  --viipa

**description**  both IP and EA are calculate by removing and adding an electron, respectively.

**queue**  setup, SP, SP, SP, properties

**Global Electrophilicity Index**

**flag**  --vomega

**description**  global electrophilicity index from vertical IP and EA.

**queue**  setup, SP, SP, SP, properties

**Fukui Indices**

**flag**  --vfukui

**description**  calculates Mulliken partial charges from the neutral, positive and negatively charged structure and calculates Fukui indices.

**queue**  setup, SP, SP, SP, properties

**Electrostatic Potential**

**flag**  --esp

**description**  calculate electrostatic potential on VdW-grid

**queue**  setup, SP, properties (with ESP calculation)

**STM picture**

**flag**  --stm

**description**  simulate a STM measurement (molecule should be aligned to xy-plane)

**queue**  setup, SP, properties (with STM calculation)

**Geometry optimization**

**flag**  --opt
**description** approximate normal coordinate optimization, performs an initial singlepoint calculation and a final singlepoint calculation on the optimized structure. See *Geometry Optimization* for details.

**queue** setup, SP, ANCopt, SP, properties

**Minimum Hopping**

**flag** --metaopt

**description** try to find conformers by geometry optimization, for each minimum located a bias potential is generated to push the optimizer to another local minimum.

**queue** setup, SP, ANCopt, SP, properties, ANCopt, ...

**Guided Path Finder**

**flag** --path [file]

**description** apply a bias potential between the input and final geometry (from file) and force the geometry optimizer to generate a path between the two structures.

**queue** setup, SP, properties, ANCopt, ...

**Modefollowing**

**flag** --modef mode

**description** follow mode which specifies the nth eigenmode from a previously done frequency calculation.

**queue** setup, SP, properties, ANCopt, ...

**Frequency calculation**

**flag** --[o]hess

**description** second derivative calculation, see *Calculation of Vibrational Frequencies*

**queue** setup, SP, [ANCopt, SP] SP, Hessian, properties

**Molecular dynamics**

**flag** --[o]md

**description** molecular dynamics simulation, see *Molecular Dynamics Simulations* for details

**queue** setup, SP, [ANCopt, SP] properties, MD

**Metadynamics**

**flag** --metadyn [snapshots]

**description** activates metadynamics simulation on start geometry, where snapshots is the number of structures from the trajectory should be used in the biasing potential. See *Meta-Dynamics Simulations* for details.

**queue** setup, SP, properties, MD

**Simulated annealing**

**flag** --siman

**description** performs a number of simulated annealing steps on the input coordinates and tries to find a conformer ensemble. We recommend the CREST workflow (see *Introduction to CREST*) instead of this runtyp since it is faster and more reliable in finding the lowest conformer. **This runtyp has been deprecated in version 6.2!**
queue  setup, SP, properties, MD, ANCopt, ...
The xcontrol instruction set is inspired by the Turbomole control file syntax. I decided to call it xcontrol instructions back then, but here we will just call it (detailed) input for convenience.

Note: The parser implemented is more general and limited by arbitrary choice to this syntax. At some point more common formats like JSON, YAML or XML might become available as alternative input formats.

To read an input file called xtb.inp use

```bash
> xtb --input xtb.inp coord
```

In the detailed input you have control about almost very global variable in the program, some instructions even check your input, but most of the time you should know what you are doing. Developed as a feature for developers, this is incredible powerful and naturally way to complicated for the average application. So in most cases you can safely rely on the internal defaults or the shipped global configuration file (should usually be the same).

I will walk you through some selected instructions you might find useful for your application.

**Contents**

- Detailed Input
  - Fixing, Constraining and Confining
    - Exact Fixing
    - Constraining Potentials
    - Confining in a Cavity
  - Absolute Control
    - Global Configuration File
4.1 Fixing, Constraining and Confining

In *xtb* different concepts of constraints are implemented, so you should know which tool is best for your problem before you start writing the detailed input.

4.1.1 Exact Fixing

In the *exact fixing* approach the Cartesian position of the selected atom is fixed in space by setting its gradient to zero and the degrees of freedom are removed from the optimization procedure and therefore the atoms stay in place in geometry optimizations.

For dynamics this exact fixing is *automatically deactivated*, since it usually leads to instabilities in the simulation.

To activate the exact fixing for atoms 1–10 and atom 12 as well as for all oxygen atoms, add

```plaintext
$fix
  atoms: 1-10,12
  elements: O
$end
```

to your detailed input, the atoms keyword refers to the numbering of the individual atoms in your input geometry.

4.1.2 Constraining Potentials

Almost absolute control about anything in your system is archived by applying *constraining potentials*. First of all the constraining potentials offer a weaker version of the exact fixing, which is invoked by the same syntax in the `$constrain` data group as

```plaintext
$constrain
  atoms: 11
  elements: C,N,8
$end
```

the program will not attempt to hold the Cartesian positions constant, but the distances between all selected atoms, here number 11 and all carbon, nitrogen and oxygen. For each atom pair a harmonic potential is generated to hold the distances at roughly the starting value, this even works without problems in dynamics.

To constrain the atoms more tightly the force constant can be adjusted

```plaintext
$constrain
  force constant=1.0
$end
```

this variable goes directly into the constraining procedure and is given in Hartree, for very high force constants this becomes equivalent to the exact fixing. Note the difference in the syntax as you are required to use an equal-sign instead of a colon, as you are modifying a global variable.

It is also possible to constrain selected internal coordinates, possible are distances, angles and dihedral angles as done here

```plaintext
$constrain
  distance: 1, 2, 2.5
  angle: 5, 7, 8, 120
  dihedral: 3, 4, 1, 7, auto
$end
```
Distance constraints are given in Ångström, while angle constraints are given in degrees. The distances are defined by two atom number referring to the order in your coordinate input, angles are defined by three atom numbers and dihedral angles by four atoms, in any case the atoms do not have to be connected by bonds. The last argument is always the value which should be used in the constraining potential as reference, if you decide to use the current value auto can be passed. The constraints will be printed to the screen (the newer implementation may require the verbose mode, to trigger the printout of the constraint summary).

If you are not quite sure which distances or angles you want to constrain, run

```bash
> cat geosum.inp
$write
distances=true
angles=true
torsions=true
$end
> xtb --define --verbose --input geosum.inp coord
```

and have a look at the geometry summary for your molecule. The $write data group toggles the printout in the property section and also some printouts in the input section.

### 4.1.3 Confining in a Cavity

If you are running dynamics for systems that are non-covalently bound, you may encounter dissociation in the dynamics. If you want to study the bound complex, you can try to confine the simulation in a little sphere, which keeps the molecules from escaping. The detailed input looks like

```bash
$cma
$sql
   potential=logfermi
   sphere: auto, all
$end
```

You can be more precise on the radius by giving the value in bohr instead of auto. I personally recommend to use the logfermi potential, since it is best suited for confinements, but yet not the default.

**Note:** When using a confining potential for confining you should make sure the origin is close to the center of geometry or center of mass of the molecule, since all confining potentials are centered at the origin (0,0,0), which is rather a limitation of the current input than the underlying implementation. To avoid problems with misplaced confining potentials the $cma logical instruction should be included to shift the molecule back to the center of mass and align it to its principal axes of inertia.

**Example for using wall potentials:**

```bash
> cat wall.inp
$chrg -1
$spin 0
$sql
   ellipsoid: 13.5,11.1,8.6,all
$end
xtb input-geometry.xyz --input constrain.inp --sp > sp.out
```

For visualization purposes the transparent-green dots are placed on the surface of the potential.

### 4.1. Fixing, Constraining and Confining
The influence of the ellipsoidal potential on the caffeine molecule in a single-point calculation is listed in the summary output block:

```
:: SUMMARY ::
:: total energy -42.277068245167 Eh ::
:: gradient norm 0.125348812811 Eh/a0 ::
:: HOMO-LUMO gap 0.387517637701 eV ::
:: SCC energy -42.804281029385 Eh ::
:: -> isotropic ES 0.200135046318 Eh ::
:: -> anisotropic ES 0.005440996407 Eh ::
:: -> anisotropic XC 0.010691562913 Eh ::
:: -> dispersion -0.024921224966 Eh ::
:: repulsion energy 0.492228803150 Eh ::
:: add. restraining 0.034887396892 Eh ::
:::-----------------------------
```

### 4.2 Absolute Control

As I promised you can control almost everything, the `xcontrol(7)` man page is a good starting point to get acquainted with the detailed input. This proses the usual hinderance of actually reading the documentation (since you are here, you are already above average, thumbs up).

A practical alternative is to just dump the complete internal settings of the program to an input file and start playing around with it. To do so, run

```
> xtb --input default.inp --define --copy coord
```

The file `default.inp` has not to be present when starting the program in `--copy` mode, since the `default.inp` will be generated for you. The `--define` flags makes sure that the program only checks your setup and does not
perform any calculation on the input coordinates.

Have a look at the first lines of `default.inp`:

```
$cmd xtb --input default.inp --define --copy coord
$date 2019/03/05 at 08:50:26.651
$chrg 0
$spin 0
...
```

This is actually the command you used in the first place to invoke the program, next you find the timestamp when
the program was started and then systemspecific information about charge and spinstate of your system, this is what I
understand as a self-documenting program run. `$cmd` and `$date` are cosmetic features and will never influence any
calculation if included in the detailed input, but I figured that they might become handy if you look back into your
calculations when putting together the manuscript or taking over a project from your, now graduated, fellow coworker.

The rest of the file represent every accessible variable documented in the `xcontrol(7)` man page with its current
setting, this should be quite a lot. So lets focus say on the `$wall` group:

```
...
$wall
  potential=polynomial
  alpha=30
  beta=6.000000000000000
  temp=300.0000000000000
  autoscale=1.000000000000000
  axisshift=3.500000000000000
...
```

The default potential is a polynomial one, you want to change this to the logfermi potential. alpha is only
needed for the polynomial potential, we use beta and temp in our potential. The steepness of our potential can
be adjusted by modifying the value of beta, since our potential is multiplied with the thermic energy we can scale it
by increasing it temperature in temp. autoscale is a factor the automatic determined sphere axes are multiplied
with, a default of 1.0 seems resonable here, but sometimes we need more space or want to squeeze everything a bit
together. We can also adjust the constant shift value used in the generation of the automatic axes, but on a second
thought this value might be just fine, so we do not modify axisshift today.

This is an awful lot of information in a small block and quite essential for your calculation using a confining potential,
all details on this can be found in `xcontrol(7)` man page at the group instruction of interest.

---

**Tip:** If you are happy with all this setting you can just use this file as your `.xtbrc` and place it somewhere in your
`XTBPATH`.

### 4.2.1 Global Configuration File

The global configuration file called `.xtbrc` has to be around somewhere in your `XTBPATH` so `xtb` is able to find
it and uses the very same syntax as the detailed input. Every instruction (key=value) you can use in your detailed
input file can be present in your global configuration file. Systemspecific instructions (key: value) will not work,
of course. To check which `.xtbrc` is read, start the program in verbose mode and check the `Calculation Setup` section
in the output.
CHAPTER 5

Singlepoint Calculations

Contents

• Singlepoint Calculations
  – Input
  – Charge and Multiplicity
  – Accuracy and Iterations
    * Accuracy
    * Iterations
  – Fermi-smearing
  – Vertical Ionization Potentials and Electron Affinities
  – Global Electrophilicity Index
  – Fukui Index
    * Example: BF$_3$

Note: Generally, a singlepoint calculation will be carried out automatically before every other calculation done with xtb.

5.1 Input

To start a singlepoint calculation with xtb only a molecular geometry is needed. xtb supports the TURBOMOLE coordinates (coord), any valid Xmol (e.g. .xyz) and Structure-Data files (.sdf).

Example TURBOMOLE input coordinates for H$_2$O (e.g. coord):
Example Xmol input coordinates for H$_2$O (e.g. h2o.xyz):

<table>
<thead>
<tr>
<th>3 Comment Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>O 0.000000 0.000000 -0.389361</td>
</tr>
<tr>
<td>H 0.762984 0.000000 0.194680</td>
</tr>
<tr>
<td>H -0.762984 0.000000 0.194680</td>
</tr>
</tbody>
</table>

Note: For any valid Xmol file xtb will actually count the lines and double check the number of atoms specified, here the suffix .xyz is optional since xtb will automatically detect the file type.

Example SDF input for H$_2$O (e.g. h2o.sdf)

Water
Comment Line 1
Comment Line 2
3 2 0 0 0 999 V2000
0.0021 -0.0041 0.0020 H 0 0 0 0 0 0 0 0 0 0 0 0
-0.0110 0.9628 0.0073 O 0 0 0 0 0 0 0 0 0 0 0 0
0.8669 1.3681 0.0011 H 0 0 0 0 0 0 0 0 0 0 0 0
1 2 1 0 0 0 0
2 3 1 0 0 0 0
M END
$$$$

Note: To use input coordinates in SDF format the .sdf suffix is required.

## 5.2 Charge and Multiplicity

By default xtb will search for .CHRG and .UHF files which contain the molecular charge and the number of unpaired electrons as an integer, respectively.

Example .CHRG file for a molecule with a molecular charge of +1:

```
> cat .CHRG
1
```

Example .CHRG file for a molecule with a molecular charge of -2:

```
> cat .CHRG
-2
```

Example .UHF file for a molecule with two unpaired electrons:

```
> cat .UHF
2
```
The molecular charge can also be specified directly from the command line:

```
> xtb coord --chrg <INTEGER>
```

which is equivalent to

```
> echo <INTEGER> > .CHRG && xtb coord
```

This also works for the unpaired electrons as in

```
> xtb coord --uhf <INTEGER>
```

being equivalent to

```
> echo <INTEGER> > .UHF && xtb molecule.xyz
```

Example for a +1 charged molecule with 2 unpaired electrons:

```
> xtb --chrg 1 --uhf 2
```

**Note:** The molecular charge or number of unpaired electrons specified from the command line will override specifications provided by `.CHRG`, `.UHF` and the `xcontrol` input!

The imported specifications are documented in the output file in the *Calculation Setup* section.

```
<table>
<thead>
<tr>
<th>Calculation Setup</th>
</tr>
</thead>
</table>

program call : xtb molecule.xyz  
hostname : user  
coordinate file : molecule.xyz  
omp threads : 4  
number of atoms : 3  
number of electrons : 7  
charge : 1 # Specified molecular charge  
spin : 1.0 # Total spin from number of unpaired electrons (S=2*0.5=1)  
first test random number : 0.54680533077496
```

**Note:** Note that the position of the input coordinates is totally unaffected by any command-line arguments, if you are not sure, whether `xtb` tries to interpret your filename as flag use `--` to stop the parsing as command-line options for all following arguments.

```
> xtb -- -oh.xyz
```

To select the parametrization of the xTB method you can currently choose from three different geometry, frequency and non-covalent interactions (GFN) parametrizations, which differ mostly in the cost–accuracy ratio,

```
> xtb --gfn 2 coord
```

to choose GFN2-xTB, which is also the default parametrization. Also available are GFN1-xTB, and GFN0-xTB.

5.2. Charge and Multiplicity
5.3 Accuracy and Iterations

5.3.1 Accuracy

The accuracy of the xTB calculation can be adjusted by the commandline option `--acc`. The accuracy determines the integral screening thresholds and the SCC convergence criteria and can be adjusted continuous in a range from 0.0001 to 1000, where tighter criteria are set for lower values of accuracy. To change the calculation accuracy call `xtb` with

```
> xtb coord --acc <REAL>
```

By default the accuracy multiplier is set to 1, for a few accuracy settings the resulting numerical thresholds are shown below:

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>30</th>
<th>1</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integral cutoff</td>
<td>20.0</td>
<td>25.0</td>
<td>32.0</td>
</tr>
<tr>
<td>Integral neglect</td>
<td>3.0 · 10^7</td>
<td>1.0 · 10^8</td>
<td>2.0 · 10^9</td>
</tr>
<tr>
<td>SCC convergence / $E_h$</td>
<td>3.0 · 10^3</td>
<td>1.0 · 10^6</td>
<td>2.0 · 10^7</td>
</tr>
<tr>
<td>Wavefunction convergence / $e$</td>
<td>3.0 · 10^3</td>
<td>1.0 · 10^4</td>
<td>2.0 · 10^9</td>
</tr>
</tbody>
</table>

**Note:** The wavefunction convergence in GFN2-xTB is chosen automatically a bit tighter than for GFN1-xTB.

5.3.2 Iterations

The number of iterations allowed for the SCC calculation can be adjusted from the command line:

```
> xtb coord --iterations <INTEGER>
```

The default number of iterations in the SCC is set to 250.

5.4 Fermi-smearing

The electronic temperature $T_{el}$ is used as an adjustable parameter, employing so-called Fermi smearing to achieve fractional occupations for systems with almost degenerate orbital levels. This is mainly used to take static correlation into account or to e.g. investigate thermally forbidden reaction pathways.

$T_{el}$ enters the GFNn-xTB Hamiltonian as

$$G_{fermi} = -T_{el}S_{el}$$

and the orbital occupations for a spin orbital $\psi_i$ are given by

$$n_i(T_{el}) = \frac{1}{\exp[(\epsilon_i - \epsilon_F)/(k_B T_{el})] + 1}$$

The default electronic temperature is $T_{el} = 300$ K.

$T_{el}$ can be adjusted by the command line:

```
> xtb --etemp <REAL> molecule.xyz
```

The specified electronic temperature is documented in the output file in the *Self-Consistent Charge Iterations* section.
### 5.5 Vertical Ionization Potentials and Electron Affinities

*xtb* can be used to calculate vertical ionization potentials (IP) and electron affinities (EA) applying a specially reparameterized GFN1-xtB version. The special purpose parameters are documented in the `.param_ipea.xtb` parameter file.

The vertical ionization potential or electron affinity is obtained as the energy difference between the corresponding molecule groundstate and its ionized species in the same geometry.

\[
IP_v = E(M^{n+1}) - E(M^n)
\]

\[
EA_v = E(M^{n-1}) - E(M^n)
\]

**Note:** The sign of the IP and EA can differ in the literature due to different definitions.

The vertical IP and EA calculations can be evoked from the command line either separately or combined.

>`xtb coord --vip`

**Note:** Sometimes you may face difficulties converging the self consistent charge iterations. In this case increasing the electronic temperature and restarting at the converged calculation with normal temperature can help.

>`xtb coord --etemp 1000.0 && xtb coord --restart`
Note: It is recommended to optimize the molecule geometry prior to the vipea calculation.

The calculated IP and/or EA are then corrected empirically, both the empirical shift and the final IP and/or EA are documented in the output in the vertical delta SCC IP calculation and vertical delta SCC EA calculation sections.

Example output for the optimized Water molecule:

```
--- vertical delta SCC IP calculation ---

*** removed SETUP and SCC details for clarity ***

:: SUMMARY ::
:: total energy -5.141603209729 Eh ::
:: gradient norm 0.051348781702 Eh/α ::
:: HOMO-LUMO gap 6.668725933430 eV ::
:: SCC energy -5.189558706232 Eh ::
:: -> electrostatic 0.159050410368 Eh ::
:: repulsion energy 0.048093066315 Eh ::
:: dispersion energy -0.000137569813 Eh ::
:: halogen bond corr. 0.000000000000 Eh ::
:: add. restraining 0.000000000000 Eh ::

empirical IP shift (eV): 4.8455 # Empirical shift
delta SCC IP (eV): 13.7897 # Finally calculated vertical IP (Exp.: 12.6 eV)
```

```
--- vertical delta SCC EA calculation ---

*** removed SETUP and SCC details for clarity ***

:: SUMMARY ::
:: total energy -5.929826433613 Eh ::
:: gradient norm 0.016238133270 Eh/α ::
:: HOMO-LUMO gap 7.760066297206 eV ::
:: SCC energy -5.97781930116 Eh ::
:: -> electrostatic 0.169754616317 Eh ::
:: repulsion energy 0.048093066315 Eh ::

(continues on next page)
```

(continues on next page)
5.6 Global Electrophilicity Index

_xtb_ can be used for direct calculation of Global Electrophilicity Indexes (GEI) that can be used to estimate the electrophilicity or Lewis acidity of various compounds from vertical IPs and EAs. In _xtb_ the GEI is defined as:

\[
GEI = \frac{(IP + EA)^2}{8(IP - EA)}
\]

The GEI calculation can be evoked from the command line:

```bash
tbx coord --vomega
```

The calculated GEI is documented in the output after the _vertical delta SCC EA calculation_ section

---

Calculation of global electrophilicity index \((IP+EA)^2/(8\cdot(IP-EA))\)
Global electrophilicity index (eV): 1.0923 #GEI for water
---

5.7 Fukui Index

The Fukui indexes or condensed Fukui function can be calculated to estimate the most electrophilic or nucleophilic sites of a molecule.

\[
f(r) = \frac{\delta p(r)}{\delta N_{\text{electron}}}
\]

The two finite representations of the Fukui function are defined as

\[
f_+(r) = \rho_{N+1}(r) - \rho_N(r)
\]

representing the electrophilicity (susceptibility of an nucleophilic attack) of an atom in a molecule with N electrons and

\[
f_-(r) = \rho_N(r) - \rho_{N-1}(r)
\]

representing the nucleophilicity (susceptibility of an electrophilic attack) of an atom.

The radical attack susceptibility is described by

\[
f_0(r) = 0.5(\rho_{N+1}(r) - \rho_{N-1}(r))
\]
As the Fukui indexes depend on occupation numbers and population analysis (see Properties), they are sensitive toward basis set changes. Therefore Fukui indexes should not be recognized as absolute numbers but as relative parameters in the same system.

A Fukui index calculation can be evoked from the command line:

```bash
> xtb coord --vfukui
```

The calculated Fukui indexes are documented in the Fukui index Calculation section of the output.

### 5.7.1 Example: BF$_3$

<table>
<thead>
<tr>
<th>Fukui index Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 -15.6291014 -0.156291E+02 0.835E+00 13.96 0.0 T</td>
</tr>
<tr>
<td>2 -15.6761217 -0.470203E-01 0.533E+00 13.46 1.0 T</td>
</tr>
<tr>
<td>3 -15.6768113 -0.689578E-03 0.156E+00 13.00 1.0 T</td>
</tr>
<tr>
<td>4 -15.6769156 -0.104364E-03 0.175E-01 12.86 1.0 T</td>
</tr>
<tr>
<td>5 -15.6769184 -0.275858E-05 0.213E-02 12.90 2.3 T</td>
</tr>
<tr>
<td>6 -15.6769197 -0.132996E-05 0.325E-03 12.91 19.8 T</td>
</tr>
<tr>
<td>7 -15.6769197 -0.144533E-07 0.264E-05 12.91 1896.8 T</td>
</tr>
<tr>
<td>8 -15.6769197 -0.126121E-11 0.650E-06 12.91 7694.1 T</td>
</tr>
</tbody>
</table>

SCC iter. ... 0 min, 0.001 sec
gradient ... 0 min, 0.000 sec

<table>
<thead>
<tr>
<th>#</th>
<th>f(+)</th>
<th>f(-)</th>
<th>f(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 B</td>
<td>-0.300</td>
<td>0.005</td>
<td>-0.148</td>
</tr>
<tr>
<td>2 F</td>
<td>-0.233</td>
<td>-0.335</td>
<td>-0.284</td>
</tr>
<tr>
<td>3 F</td>
<td>-0.233</td>
<td>-0.335</td>
<td>-0.284</td>
</tr>
<tr>
<td>4 F</td>
<td>-0.233</td>
<td>-0.335</td>
<td>-0.284</td>
</tr>
</tbody>
</table>

The Fukui indexes for BF$_3$ indicate the most negative f(+) value and a positive value for f(-) at the boron atom. Thus, a nucleophilic attack can be expected at the boron atom.
In this chapter, all necessary information about the properties of xTB will be given. Description of how to acquire different output information will be provided. Calculation of FOD will be described.

### Contents

- **Properties**
  - General printout
    - GFN1-xTB
    - GFN2-xTB
  - Density Properties
    - Cube Files
    - Density and Spin-Density
    - Fractional Occupation Density (FOD) calculation
  - Redirecting Property Printout
  - Machine Readable Data Dump

### 6.1 General printout

First the orbital energies and occupation are printed, where the highest occupied molecular orbital (HOMO) and the lowest unoccupied molecular orbital (LUMO) are marked. The HOMO-LUMO gap and the Fermi-level are summed up.

<table>
<thead>
<tr>
<th>Property Printout</th>
<th></th>
</tr>
</thead>
</table>
* Orbital Energies and Occupations

<table>
<thead>
<tr>
<th>#</th>
<th>Occupation</th>
<th>Energy/Eh</th>
<th>Energy/eV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0000</td>
<td>-0.6801050</td>
<td>-18.5066</td>
</tr>
<tr>
<td>2</td>
<td>2.0000</td>
<td>-0.5683264</td>
<td>-15.4649</td>
</tr>
<tr>
<td>3</td>
<td>2.0000</td>
<td>-0.5108650</td>
<td>-13.9013</td>
</tr>
<tr>
<td>4</td>
<td>2.0000</td>
<td>-0.4463919</td>
<td>-12.1469</td>
</tr>
<tr>
<td>5</td>
<td>0.0826818</td>
<td>2.2499</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>0.2518567</td>
<td>6.8534</td>
<td>0.000</td>
</tr>
</tbody>
</table>

HL-Gap 0.5290737 Eh 14.3968 eV
Fermi-level -0.1818551 Eh -4.9485 eV

The information provided by the printout can be modified and extended. This can be done either by using the option-flags when calling the program (Commandline Usage), or by editing the input file (Detailed Input). The kind of default information given is determined by the GFN-xTB version used. The default values called by the program are given:

--pop requests printout of Mulliken population analysis
--molden requests printout of molden file
--dipole requests printout of dipole moments
--wbo requests Wiberg bond order printout

6.1.1 GFN1-xTB

Default settings for GFN1-xTB first prints the Mulliken and CM5 charges. n(x) denotes the population partitioned to the x = s/p/d shells:

<table>
<thead>
<tr>
<th>Mulliken/CM5 charges</th>
<th>n(s)</th>
<th>n(p)</th>
<th>n(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 O 0.67569 0.33312 1.682 4.994 0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 H -0.33784 -0.16656 0.662 0.000 0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 H -0.33784 -0.16656 0.662 0.000 0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wiberg bond orders describe the partial bond orders and their disposition onto the atoms:

<table>
<thead>
<tr>
<th>Wiberg/Mayer (AO) data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>largest (&gt;0.10) Wiberg bond orders for each atom</td>
</tr>
<tr>
<td>total WBO</td>
</tr>
<tr>
<td>1 O 1.782</td>
</tr>
<tr>
<td>2 H 0.892</td>
</tr>
<tr>
<td>3 H 0.892</td>
</tr>
</tbody>
</table>

The molecular dipole moment and its cartesian components calculated from the electron density. The components are given in atomic units while the total dipole moment is given in Debye, to convert from atomic units to Debye multiply by 2.5417 D/au.

dipole moment from electron density (au)

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8659</td>
<td>0.0000</td>
<td>0.6123</td>
</tr>
</tbody>
</table>

total (Debye): 2.696
6.1.2 GFN2-xTB

Default settings for GFN2-xTB first prints populations and coefficients. From left to right, these are the atomic number Z, Coordination number CN, Atomic partial charge q, Dispersion coefficient C6, Polarizability alpha:

<table>
<thead>
<tr>
<th>#</th>
<th>Z</th>
<th>covCN</th>
<th>q</th>
<th>C6AA</th>
<th>α (0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>O</td>
<td>1.613</td>
<td>-0.568</td>
<td>24.435</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>H</td>
<td>0.806</td>
<td>0.284</td>
<td>0.771</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>H</td>
<td>0.806</td>
<td>0.284</td>
<td>0.771</td>
</tr>
</tbody>
</table>

The C6, C8 and alpha coefficients are denoted explicitly in a.u.:

- Mol. C6AA /au·bohr^6: 44.553640
- Mol. C8AA /au·bohr^8: 796.459844
- Mol. α(0) /au: 9.429351

Wiberg bond orders:

| largest (>0.10) Wiberg bond orders for each atom |
| total WBO | WBO to atom ... |
| 1 O 1.839 H | 3 0.919 H 2 0.919 |
| 2 H 0.919 O | 1 0.919 |
| 3 H 0.919 O | 1 0.919 |

Molecular dipole and quadropole moments. The contributions are separted into their respective cartesian dimensions. ‘Full’ represents the corresponding contributions of the molecular dipole or quadropole moments.

- molecular dipole:
  - x: 0.481
  - y: 0.000
  - z: 0.340
  - tot (Debye): 2.167

- molecular quadrupole (traceless):
  - xx: 0.305
  - xy: 0.000
  - yy: -0.916
  - xz: -0.432
  - yz: 0.000
  - zz: 0.610

All is summed up in the end in both GFN-xTB versions:

<table>
<thead>
<tr>
<th>TOTAL ENERGY</th>
<th>-5.070322476938 Eh</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRADIENT NORM</td>
<td>0.019484395925 Eh/α</td>
</tr>
<tr>
<td>HOMO-LUMO GAP</td>
<td>14.652302902752 eV</td>
</tr>
</tbody>
</table>

6.2 Density Properties

6.2.1 Cube Files

The xtb program is able to calculate the density, spin-density and the fractional occupation number weighted density (FOD). For these calculations, the program first creates a proper cube grid. The corresponding file is created in your working directory and marked as .cub file. It provides density and step size informations. An overview is already given in the printout:
Here, various information are provided, like the density matrix neglect threshold `cube_pthr` and the grid step size `cube_step` (in Bohr). These values can be changed in the input (xcontrol) file (Detailed Input).

For visualization, programs like chimera can be used, for which the `.cub` file can be loaded as volume data.

### 6.2.2 Density and Spin-Density

To calculate the density or the spin density, the input (xcontrol) file has to be manipulated. Here, the bools `density='bool'` or respectively `spin density='bool'` have to be set to 'true'. This will create a `.cub` cube file, where the corresponding information is gathered.

For visualization, programs like chimera can be used, for which the `.cub` file can be loaded as volume data.

### 6.2.3 Fractional Occupation Density (FOD) calculation

The fractional occupation density analysis (FOD) is a diagnostic scheme that displays the static electron correlation localized on a molecule. The density is hereby obtained by performing a computationally cheap Finite-Temperature DFT computation. The electrons are therefore self-consistently smeared over the molecular orbitals according to a Fermi-Dirac distribution. For a more detailed insight and the theory behind the FOD analytics, please see the original publication. To use FOD for selecting active spaces in CASSCF calculations, refer to our later work on this topic.

To access the FOD analysis, simply use the flag `--fod` or set `fod='true'` in the input (xcontrol) file. This will create a `fod.cub` file and calculate the FOD on the cube grid. Be sure to set the electronic temperature to a higher value, e.g. 5000 K (`--etemp 5000`). The FOD population will be displayed in the printout section as:

<table>
<thead>
<tr>
<th>NFOD</th>
<th>0.6698</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loewdin</td>
<td></td>
</tr>
<tr>
<td>FODpop</td>
<td></td>
</tr>
<tr>
<td>n(s)</td>
<td></td>
</tr>
<tr>
<td>n(p)</td>
<td></td>
</tr>
<tr>
<td>n(d)</td>
<td></td>
</tr>
<tr>
<td>1 C</td>
<td>0.1924</td>
</tr>
<tr>
<td>0.018</td>
<td>0.175</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>2 C</td>
<td>0.0673</td>
</tr>
<tr>
<td>0.000</td>
<td>0.064</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>3 C</td>
<td>0.0673</td>
</tr>
<tr>
<td>0.000</td>
<td>0.064</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>4 C</td>
<td>0.1924</td>
</tr>
<tr>
<td>0.018</td>
<td>0.175</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>5 C</td>
<td>0.0673</td>
</tr>
<tr>
<td>0.000</td>
<td>0.064</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>6 C</td>
<td>0.0673</td>
</tr>
<tr>
<td>0.000</td>
<td>0.064</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>7 H</td>
<td>0.0039</td>
</tr>
<tr>
<td>0.004</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>8 H</td>
<td>0.0039</td>
</tr>
<tr>
<td>0.004</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>9 H</td>
<td>0.0039</td>
</tr>
<tr>
<td>0.004</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>10 H</td>
<td>0.0039</td>
</tr>
<tr>
<td>0.004</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td></td>
</tr>
</tbody>
</table>

The NFOD number indicates the static electron correlation.

If you do not want to write a full `fod.cub` file, but still want to analyse the FOD population at least qualitatively, change the `fod population ='bool'` in the input (xcontrol) file to `true`. This will display the fractional loewdin population of the system (see above) and only writes the `fod` file, where this information is stored.
6.3 Redirecting Property Printout

For large systems the property printout can become quite lengthy and will clutter maybe thousands of lines in the standard output. One possibility is to rigourously deactivate all printouts using the \$write instruction in the input file, but if one might need this information later it is hard to recover, as an alternative the property printout can be redirected. Simply add

\begin{verbatim}
$write
  output file=properties.out
\end{verbatim}

to your input and specify the name for the redirection. The calculations of the properties are performed as usual but the standard output will show something like

\begin{verbatim}
Property printout bound to 'properties.out'
\end{verbatim}

instead of the header, the usual printout can be found in \texttt{properties.out}. In the file the command line call and current time is saved additionally to ensure that the printout is reproducible.

6.4 Machine Readable Data Dump

xtb is able to dump parts of the calculated data in a machine-readable way using the json-format. To activate the dump into a json file use the input

\begin{verbatim}
$write
  json=true
\end{verbatim}

which will write a \texttt{xtbout.json} file containing partial charges, cumulative atomic multipole moments, occupation number and orbital energies for single point calculations or frequencies, reduced masses and IR intensities from hessian calculations.
The basics of calculating geometry optimizations with xtb are presented in this chapter. Please check out the chapter Singlepoint Calculations for options that are not optimization-specific, as the charge or the number of SCF iterations, for example. Solvation can be included easily by using GBSA. For this approach, please read the chapter Implicit Solvation.

Contents

- Geometry Optimization
  - Optimization levels
  - Running a geometry optimization
    * Example 1: ethyne
    * Example 2: cyclopentadienyl anion
    * Example 3: p-benzyne in toluene
  - Convergence problems
  - Optimization Engines

7.1 Optimization levels

xtb has a build-in geometry optimizer called approximate normal coordinate rational function optimizer (ANCopt) which uses a Lindh-type model Hessian to generate an approximate normal coordinate system. It is activated by the flag --opt [level]. The following levels are available:
Here, energy convergence (Econv) is the allowed change in the total energy at convergence and the gradient convergence (Gconv) the allowed change in the gradient norm at convergence. The accuracy is handed to the singlepoint calculations for integral cutoffs and self consistent field convergence criteria. It is adjusted to fit the geometry convergence thresholds automatically. The maximal number of optimization cycles can be defined by using the flag \(--\)cycles integer. By default, the optimization level ‘normal’ is used. The maximum number of optimization cycles is usually automatically determined at runtime from the total degrees of freedoms and is at least 200 and at most 10000.

### 7.2 Running a geometry optimization

#### 7.2.1 Example 1: ethyne

Input structures in TURBOMOLE (coord) or Xmol coordinates can be optimized. An example xyz input for ethyne is (e.g. inp.xyz):

```
4
H 0.00 0.00 0.00
C 0.00 0.00 1.00
C 0.00 0.00 2.00
H 0.00 0.00 3.00
```

For running the geometry optimization using the defaults, call

```
> xtb inp.xyz --opt
```

A singlepoint calculation is performed. Then, the optimization setup is printed:

```
=-----------------=
|    A N C O P T  |
=-----------------=
| Approximate Normal Coordinate |
| Rational Function Optimizer   |
=-----------------=
```

(continues on next page)
This is followed by the printout of the optimization cycles. Here every 20 cycles the ANC coordinate system will be reset and an additional short summary block regarding the progress in the last few cycles will be shown.

Tip: you can find information about the accuracy of the BFGS model of the PES at each optimization step and the error between predicted and actual change in energy.

Note: the displacement summary at each step refers always to the next optimization cycle.

The convergence of the geometry optimization is confirmed by the printout

*** GEOMETRY OPTIMIZATION CONVERGED AFTER 6 ITERATIONS ***

Afterwards, a final singlepoint calculation is performed (including a property printout). The total energy and the name of the file containing the optimized coordinates are printed at the end of the output:

```
optimized geometry written to: xtbopt.xyz

-------------------------------------------------
| TOTAL ENERGY | -5.206771946579 Eh |
| GRADIENT NORM | 0.000476954973 Eh/\alpha |
-------------------------------------------------
```
Note: The input coordinates are not overwritten by xtb. The optimized geometry can be found either in the file xtbopt.xyz or xtbopt.coord depending on the format of the input.

The file xtbopt.xyz for this example looks like:

```
4
SCF done -5.206771946579 0.000476954973
H 0.00000000000000 -0.00000000000000 -0.14662251809779
C -0.00000000000000 0.00000000000000 0.90317992211836
C -0.00000000000000 0.00000000000000 2.09682010367354
H -0.00000000000000 0.00000000000000 3.14662249230588
```

Further, a trajectory of the geometry optimization written in Xmol format (even if the input was a coord file) is written to the file xtbopt.log.

### 7.2.2 Example 2: cyclopentadienyl anion

The second example is a geometry optimization of Cp. The input coordinates are far from a planar structure and are given in TURBOMOLE format as coord file.

```
$coord
  0.00000000000000 0.00000000000000 0.00000000000000   c
  0.00000000000000 0.00000000000000 2.92151660144120   c
  2.85226569757548 0.00000000000000 3.55384920112287   c
  3.90292319184177 2.03158598395524 1.73614809006603   c
  2.27186844120391 1.64373103353725 -0.65644172568502  c
-0.86886206083043 1.75686326793472 3.63081207733690  h
  3.11610359107057 0.44206741371820 5.57363951455663  h
  5.95582339684982 1.75824654746399 1.50061226486316  h
  3.42008871625882 0.73590659929899 -2.14036617906866 h
  0.31080892625410 -1.94044071311390 -0.69474836412474 h
$end
```

Now, the optimization level tight and a maximal number of 50 optimization cycles is chosen. This is done for teaching purposes only.

```
> xtb coord --opt tight --cycles 50 --charge -1
```

The ANCopt setup is adjusted as follows:

```
:                      : SETUP     :
:                       :                        :
: optimization level    tight     :
: max. optcycles        50        :
: ANC micro-cycles      20        :
: degrees of freedom    24        :
:                        :                        :
```

(continues on next page)
The geometry optimization is converged after 22 iterations. The optimized coordinates are written to the file $\texttt{xtbopt\_coord}$.

```
$coord
0.440603 77782450 -0.01412168126920 0.18353526062450 C
0.297595 94746033 0.20416120151187 2.80401943168676 C
2.639656 10517835 1.02998458234760 3.68100113536889 C
4.229990 47646770 1.3223523397087 1.60243655937779 C
2.871229 6158385 0.67587612191465 -0.55901104575941 C
-1.350636 21036312 -0.20020256016136 3.96194626998985 H
3.138096 69169362 1.3838842398555 5.64254720998168 H
6.187144 7820806 1.94516496134903 1.65710127132652 H
3.582520 7369211 0.7046430056577 -2.48621742790732 H
-1.075181 151114132 -0.62376537124033 -1.06233682418088 H
$end
```

### 7.2.3 Example 3: $p$-benzyne in toluene

As third example, the geometry optimization of $p$-benzyne in the triplet state solved in toluene is presented. The following input structure saved as inp.xyz is utilized:

```
10
C  0.000000 0.000000 0.000000
C  0.000000 0.000000 1.400000
C  1.212436 0.000000 2.100000
C  2.424871 0.000000 1.400000
C  2.424871 0.000000 0.000000
C  2.424871 0.000000 0.000000
C  1.207822 -0.105671 -0.700000
H -0.910967 0.244093 1.944500
H  1.219600 0.163768 3.176592
H  3.367973 0.000000 -0.544500
H  1.207822 -0.105671 -1.789000
```

The number of unpaired electrons (uhf) and the solvent have to be specified. Further, the optimization level ‘loose’ is chosen here for teaching purposes.

```
> xtb inp.xyz --opt loose --gbsa toluene --uhf 2
```

The thresholds corresponding to the optimization level ‘loose’ can be found in the ANCopt setup.
### SETUP

- **optimization level**: loose
- **max. optcycles**: 200
- **ANC micro-cycles**: 20
- **degrees of freedom**: 24

---

- **RF solver**: spevx
- **input Hessian**: false
- **write xtbopt.log**: true
- **linear?**: false

---

- **energy convergence**: $0.5000000\times 10^{-04}$ Eh
- **grad. convergence**: $0.4000000\times 10^{-02}$ Eh/\(\alpha\)
- **maximum RF displ.**: 1.000000
- **Hlow (freq-cutoff)**: $0.2000000\times 10^{-01}$
- **Hmax (freq-cutoff)**: 5.000000
- **S6 in model hess.**: 20.000000

---

The geometry optimization converges after five iterations, resulting in the following coordinates (written to the file `xtbopt.xyz`):

<table>
<thead>
<tr>
<th>10</th>
<th>SCF done</th>
<th>-14.6423053765</th>
<th>0.001879475862</th>
<th>! total energy in Eh and gradient norm in Eh/(\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>0.078670711152305</td>
<td>0.00730041248664</td>
<td>0.04608303752229</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.01150775744363</td>
<td>0.08123575674794</td>
<td>1.4116013847040</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>1.21188251791186</td>
<td>0.08194614686924</td>
<td>2.10875425439614</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>2.35260556407908</td>
<td>0.02986595253321</td>
<td>1.35144422932303</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>2.4304468441166</td>
<td>-0.0301861041761</td>
<td>-0.0149981049637</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>1.21887020608816</td>
<td>-0.05479836016580</td>
<td>-0.7105251252580</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>-0.9461262103426</td>
<td>0.13538712165891</td>
<td>1.93110285483949</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>1.23696707335328</td>
<td>0.12186857053414</td>
<td>3.18813400290200</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>3.773764916301</td>
<td>-0.0593253738648</td>
<td>-0.5351780739377</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>1.1921508707789</td>
<td>-0.1167795910162</td>
<td>-1.78882684103049</td>
</tr>
</tbody>
</table>

### 7.3 Convergence problems

The failure of the geometry convergence is indicated by the printout:

```plaintext
*** FAILED TO CONVERGE GEOMETRY OPTIMIZATION IN 500 ITERATIONS ***
```

Additionally, the empty file `NOT_CONVERGED` is written. If convergence problems in the SCC occur, it is recommended to start with `GFN0-xTB` which does not have to perform SCC iterations. Then the geometry optimization can be improved using `GFN2-xTB`. It can occur sometimes that a geometry does not converge correctly or at all if the calculation is carried out in the gas phase. It is recommended to use GBSA in this cases. An example for the difference made by using GBSA during the geometry optimization can be seen below. If the system is optimized in chloroform, the chloride anion coordinates the cation while the neutral compounds are formed in the gas phase.

![Fig. 1: optimized in the gasphase](image_url)
Fig. 2: optimized in chloroform

### 7.4 Optimization Engines

**Note:** feature implemented in version 6.1.4

_xtb_ offers different optimizers which can be switched using the detailed input with

```bash
$opt
  engine=<name>
```

possible optimizers are

**Approximate Normal Coordinate Rational Function optimizer (ANCopt)**
- `engine=rf` (default)
- rational function for optimal step
- BFGS update of Hessian
- approximate normal coordinate system

**L-BFGS Approximate Normal Coordinate optimizer (L-ANCopt)**
- `engine=lbfgs`
- L-BFGS step and update of Hessian
- approximate normal coordinate system

**Fast Inertial Relaxation Engine (FIRE)**
- `engine=inertial`
- MD propagation with preconditioning
- BFGS update of Hessian
- cartesian coordinate system
CHAPTER 8

Exploration of the potential energy surface (PES)

Contents

• Exploration of the potential energy surface (PES)
  – Input
  – Dihedral angle scan
    * Ethane
    * 1-Bromo-2-chloroethane
  – Angle and Distance scan
    * Ammonia
      • Concerted Scan
      • Sequential Scan

8.1 Input

xtb can be used to scan the potential energy surface. It is possible to adjust distances as well as angles and the dihedral angles via the xcontrol file. To scan the potential energy surface, you first have to optimize your molecule. Then you have to find out the atom numbers you wish to constrain and write them to your xcontrol file (see examples later). There are two different ways to modify your xcontrol.

1) First constrain your geometry, using the $constrain block, then scanning using the $scan block.

2) Constrain and scan in one step, only using the $scan block.

Both will do exactly the same within your calculation. Lastly you have to run an optimization.
Note:  
xtb uses force constants to constrain the geometry. Therefore the geometry data you use for the scan can deviate from your given input during the optimization. It is not possible to scan the potential energy surface without doing an optimization. In consequence, it is always a relaxed surface scan.

8.2 Dihedral angle scan

8.2.1 Ethane

Let's start with an easy example. You all know the potential energy surface of a dihedral angle scan of ethane, so this is a good start to check, if you are doing everything right. The following is a converged ethane structure given as xyz file.

```
> cat ethane.xyz
8
-7.46994680
C -0.01503441120750 0.04820403778911 -0.01075686629161
H -0.02618280545732 0.12324247717853 1.07751478989514
H 1.02025706611374 0.10521724412205 -0.34995019287160
H -0.56108089374185 0.89840506905011 -0.42160774836772
C -0.65104926960692 -1.26213120711759 -0.46045958867132
H -1.67328207424848 -1.33815676349645 -0.08720971228148
H -0.67543548798470 -1.32088660386766 -1.54947400918083
H -0.08161812386697 -2.11204925365809 -0.08140067223057
```

To scan the dihedral angle I choose the atoms 1 (first carbon), 5 (second carbon), 4 (hydrogen at first carbon) and 8 (hydrogen at second carbon). The dihedral angle is therefore between 8, 5, 1, 4 or vice versa. Now you have to modify your input. As I already said, there are two different ways to scan the PES.

1) Constrain and scan in two steps

```
> cat scan.inp
$constraint
force constant=0.05
dihedral: 8,5,1,4,60.0
$scan
1: 60.0,420.0,72
$end
```

The $constraint will fix the dihedral between atom 8, 5, 1 and 4 to 60.0 degrees. It is also possible to change the force constant used to constrain the geometry. For further hints see chapter Detailed Input: Fixing, Constraining and Confining - Constraining Potentials. Afterwards a scan is conducted with the 1. constraint (in this case the dihedral) from 60.0 to 420.0 degrees in 72 steps.

2) The constraint can also be done on-the-fly

```
$constraint
force constant=0.05
$scan
dihedral: 8,5,1,4,60.0; 60.0,420.0,72
$end
```

The part up to the semicolon (dihedral: 8,5,1,4,60.0) is passed to the $constraint instruction and evaluated there, and afterwards a scan with this constraint is conducted from 60.0 to 420.0 degrees in 72 steps.
Which methods you want to use is up to you, as they are doing exactly the same.

Now you are ready to start the calculation:

```
> xtb ethane.xyz --opt --input scan.inp
```

The calculation gives the usual files described in the other chapters, and an extra file called `xtbscan.log`. This is a file in XMol format, which can be read by e.g. `molden`. All optimized structures of the scan and their energy are written to that file, so it contains in our case 72 structures, starting and ending like this:

```
> cat xtbscan.log

8
SCF done -7.33636977
C -0.016167577954 0.046756019273 -0.010925351519
H -0.722837769279 0.635952345713 0.569507202626
H 0.858799528033 -0.144587130449 0.606495402607
H 0.295104980741 0.636307029349 -0.871373136822
C -0.649893664036 -1.261548762672 -0.459251334661
H -0.962364182156 -1.850869022139 0.400300369375
H -1.523304826531 -1.069936539515 -1.078572304172
H 0.057237511182 -1.850228939559 -0.39524673024
8
SCF done -7.33633001
C -0.014508419361 0.046085202187 -0.010980431860
H -0.712592003942 0.622234788963 0.592857532524
H 0.875734542981 -0.143892036194 0.583801797058
H 0.267743868859 0.650856540715 -0.870728921217
C -0.650432967813 -1.260775477944 -0.461051561431
H -0.987374035660 -1.836977921850 0.398253949047
H -1.507838805925 -1.070682914383 -1.102253264664
H 0.065841820861 -1.86503181495 -0.10324309455
8
SCF done -7.33633001
C -0.013197790454 0.045694504417 -0.011256531141
H -0.701582052023 0.608208846300 0.616415577820
H 0.892470173219 -0.14338451024 0.559597860026
H 0.239350600054 0.665739016442 -0.86935008077
C -0.650618419000 -1.26038410477 -0.462367255784
H -1.013262191587 -1.822961711987 0.395720305781
H -1.490594450886 -1.071189460974 -1.126101769970
H 0.074008130676 -1.879876638403 -0.985997178655
... ...
8
SCF done -7.33622017
C -0.013004744475 0.045540690542 -0.011114232961
H 1.030144565230 0.09505635597 -0.314432978298
H -0.533071797992 0.89433527576 -0.449705894194
H -0.065637709250 0.143792574844 1.071609413617
C -0.651185106898 -1.260114193001 -0.462340695966
H -0.711362672977 -1.30568812601 -1.547024982931
H -0.064984727672 -2.10932137114 -0.11700615435
H -1.654328060878 -1.357635285842 -0.053334013833
8
SCF done -7.33633308
C -0.014346704593 0.046121859404 -0.011308096497
H 1.020906551190 0.102932886427 -0.33987245036
```

(continues on next page)

8.2. Dihedral angle scan

45
The resulting scan as well as the resulting energy curve are shown.

Fig. 1: Dihedral scan of ethane.

![Dihedral scan of ethane](image)

Fig. 2: Energy diagram of the dihedral scan of ethane.

8.2.2 1-Bromo-2-chloroethane

xtb is also able to constrain and scan at the same time. The optimized input structure in this case (the above named disubstituted ethane) is shown below.

<table>
<thead>
<tr>
<th>Atom</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-0.551739904813</td>
<td>0.895891227197</td>
<td>-0.426589805208</td>
</tr>
<tr>
<td>C</td>
<td>-0.037115500933</td>
<td>0.129430347168</td>
<td>1.073605709020</td>
</tr>
<tr>
<td>H</td>
<td>-0.650547319963</td>
<td>-1.260694835302</td>
<td>-0.461135641012</td>
</tr>
<tr>
<td>H</td>
<td>0.684295266226</td>
<td>-1.318119390348</td>
<td>-1.546743615925</td>
</tr>
<tr>
<td>H</td>
<td>-0.080580374590</td>
<td>-2.110437772732</td>
<td>-0.091714398274</td>
</tr>
<tr>
<td>H</td>
<td>1.665707480072</td>
<td>-1.343279321813</td>
<td>-0.079585407068</td>
</tr>
</tbody>
</table>

SCF done  
-7.33637035
Now, the input can be modified. The modification shown below are only for training purposes, so they don’t have to make any sense.

```plaintext
$constrain
distance: 1,5,auto
dihedral: 8,5,1,4,60.0
$scan
  2: 60.0,780.0,100
$opt
  maxcycle=5
$end
```

I chose to firstly set the constraints and then the scanning part. The constraints are written one after another. The first constraint keeps the distance of atom 1 and atom 5 (both carbons) to their actual value, whereas the second one constrains the dihedral angle of atoms 8, 5, 1 and 4 to 60.0°. In the scan block, the second constraint is changed from 60.0° to 780.0° in 100 steps. The maximum cycle of the optimization is set in the $opt part to 5. Consequently the optimization has not much steps to shift the constrained atoms. All in all, the chosen settings results in the following.

Fig. 3: Scan of the dihedral angle between chloride and bromide of the disubstituted ethane.

Fig. 4: Energy diagram of the dihedral scan.

8.2. Dihedral angle scan
8.3 Angle and Distance scan

8.3.1 Ammonia

Concerted Scan

It is of course also possible to scan along angle and distance. Therefore in the next example, the angle and the distance is scanned in one scan. If you want to scan all options simultaneously, you have to use the mode=concerted option in the $scan block. It requires, that the number of steps in the scan is always the same, otherwise xtb will exit in error.

The optimized input geometry is written below.

> cat ammonia.xyz

4

N  -0.00990404770994  -0.01698500657667  -0.00712107610609
H   0.00434306677482  0.00733732515324  1.00490910707310
H 0.94901246801925  0.00720047578638 -0.33083175918033
H -0.46827248708413  0.82547620563705 -0.33095427178668

Now, the constraining and the scanning options are set in the input file.

$constrain
force constant=0.5
distance: 1, 4, 0.5
angle: 2, 1, 3, 150.0
dihedral: 2, 1, 3, 4, auto
$scan
mode=concerted
# different steps for each constraint!
1: 0.5, 1.4, 50
2: 150.0, 90.0, 60
$opt
maxcycle=5
$end

The constraint of the dihedral angle between all given atoms to their actual value was set, since otherwise the first optimization would lead to a planar molecule. As you can see, I chose mode=concerted, but different number of steps for the scan. Since a concerted scan can only performed if all scans are performed with the same number of steps, xtb should exits in error, of course it does printing the message:

#ERROR! Wrong setup for concerted scan, aborting...

Note: A concerted scan can only carried out if all constraints are scanned with the same number of steps.

The correct input is given below.

$constrain
force constant=0.5
distance: 1, 4, 0.5
angle: 2, 1, 3, 150.0
dihedral: 2, 1, 3, 4, auto
$scan
mode=concerted
1: 0.5, 1.4, 60

(continues on next page)
The resulting path can be seen in the following movie.

**Note:** Nobody stops you from scanning the same constraint twice, this usually does not make much sense for concerted scans, but is not caught by the parser.

### Sequential Scan

Another way to scan would be in *sequential* mode. **xtb** will then scan along all constraints one after the other, always leaving the last scanned constraint at its last value. An example *input* can look like this, using the ammonia example from above.

```
$constrain
  force constant=0.5
  distance: 1, 2, 0.5
  angle: 4, 1, 3, 140.0
  dihedral: 2, 1, 3, 4, auto
$scan
  mode=sequential
  2: 140.0, 90.0, 40
  1: 0.5, 2.0, 60
$opt
  maxcycle=5
$end
```

The *mode=sequential* flag is the default value for scans, due to the nature of the scan different step sizes are possible here.

**Note:** There is no multidimensional scan supported on purpose since they tend to be expensive on high-dimensional potential energy surfaces and are difficult to visualize. But they can be easily constructed, by repeatedly scanning the same constraint.

I chose to firstly scan along the angle and then along the distance. The resulting path can be seen below.

**Tip:** If your resulting path oscillates at some point, try to increase the number of maximum cycles *maxcycle* in your $opt block. Sometimes **xtb** just needs more steps to properly converge your structure.
In this chapter, all necessary information will be given in order to use the implicit solvent model GBSA in xTB calculations. Parameterized solvents and available grids are given as well.

**Contents**

- Implicit Solvation
  - General command-line control
  - Parameterized Solvents
  - Available Grids
  - Reference States
  - Extended Functionality
    * Solvent Accessible Surface Area

### 9.1 General command-line control

The generalized born (GB) model with solvent accessible surface area (SASA) termed GBSA is envoked with the flag `--gbsa [Solvent]` or alternative `-g [Solvent]`. As an example the single point calculation employing the GBSA model for solvation in water would be started by

```bash
> xtb coord --gbsa water
```

As an example the energy printout of a singlepoint calculation of a H$_2$O molecule in implicit water is given.

```
:: SUMMARY ::
```

(continues on next page)
The solvation free energy is printed as $G_{\text{solv}}$ and is also added to all total energy printouts.

Optimizing a geometry with the GBSA model can be done with the following input:

```bash
> xtb coord --opt --gbsa water
```

The order of the flags can be altered and the input is not case sensitive. Like in a optimization without GBSA the optimized coordinates are written to a new file (`xtbopt.coord`). In General the GBSA can be used in combination with all available run types implemented in the `xtb`.

## 9.2 Parameterized Solvents

The GBSA model is parameterized for the Hamiltonian of GFN1-xTB and GFN2-xTB, but not for GFN0-xTB. Also some solvents were parameterized only for GFN1 or GFN2. Here is a list of the available solvents.

<table>
<thead>
<tr>
<th>solvents</th>
<th>GFN1</th>
<th>GFN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acetone</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Acetonitrile</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Benzene</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>CH$_2$Cl$_2$</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CHCl$_3$</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CS$_2$</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>DMF</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>DMSO</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Ether</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Water (H$_2$O)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Methanol</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>n-Hexan</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>THF</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Toluene</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
9.3 Available Grids

Different Lebedev grids for the calculation of the SASA term are implemented in GFN-xTB. The grids are independent of the used GFNn method and are called as example like this

```bash
> xtb coord --opt --gbsa toluene tight
```

The default grid level is `normal`. The available grid levels are given in the table below with the corresponding number of gridpoints.

<table>
<thead>
<tr>
<th>Gridlevel</th>
<th>Gridpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>230</td>
</tr>
<tr>
<td>tight</td>
<td>974</td>
</tr>
<tr>
<td>verytight</td>
<td>2030</td>
</tr>
<tr>
<td>extreme</td>
<td>5810</td>
</tr>
</tbody>
</table>

Larger grids increase the computation time and reduce numerical noise in the energy. They may help to converge geometry optimizations with GBSA for large molecules which would otherwise not converge due to numerical noise.

9.4 Reference States

The default reference state option is `bar1M` which should not be changed for normal production runs. In order to compare the solvation free energy with solvation free energies from COSMO-RS the reference state can be set to `reference` which corresponds to the same `reference` option as in COSMO-RS. This could be done with

```bash
> xtb coord --opt --gbsa water reference
```

9.5 Extended Functionality

9.5.1 Solvent Accessable Surface Area

**Note:** feature implemented in version 6.2

To get more insights and diagnostics for a GBSA calculation the Born radii and the solvent accessible surface area can be printed by toggling the property-printout with

```bash
$write
gbsa=true
```

The printout for a branched octane isomer using GBSA(Water) looks like

<table>
<thead>
<tr>
<th>#</th>
<th>Z</th>
<th>Born rad/Å</th>
<th>SASA/Å²</th>
<th>H-bond</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6 C</td>
<td>3.761</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>6 C</td>
<td>3.761</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>6 C</td>
<td>2.741</td>
<td>1.820</td>
<td>-0.000</td>
</tr>
<tr>
<td>4</td>
<td>6 C</td>
<td>2.741</td>
<td>1.839</td>
<td>-0.000</td>
</tr>
</tbody>
</table>

(continues on next page)
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>C</td>
<td>2.741</td>
<td>1.817</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>C</td>
<td>2.741</td>
<td>1.820</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>C</td>
<td>2.741</td>
<td>1.839</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>C</td>
<td>2.741</td>
<td>1.817</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>H</td>
<td>2.136</td>
<td>11.404</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>H</td>
<td>2.130</td>
<td>12.571</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>H</td>
<td>2.098</td>
<td>14.966</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>H</td>
<td>2.130</td>
<td>12.563</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>H</td>
<td>2.098</td>
<td>14.979</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>H</td>
<td>2.136</td>
<td>11.403</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>H</td>
<td>2.136</td>
<td>11.412</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>H</td>
<td>2.130</td>
<td>12.524</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>H</td>
<td>2.098</td>
<td>14.948</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>H</td>
<td>2.136</td>
<td>11.404</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>H</td>
<td>2.130</td>
<td>12.571</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>H</td>
<td>2.098</td>
<td>14.966</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>H</td>
<td>2.130</td>
<td>12.563</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>H</td>
<td>2.098</td>
<td>14.979</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>H</td>
<td>2.136</td>
<td>11.403</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>H</td>
<td>2.136</td>
<td>11.412</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>H</td>
<td>2.130</td>
<td>12.524</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>H</td>
<td>2.098</td>
<td>14.948</td>
</tr>
</tbody>
</table>

**total SASA / Å² : 244.491**

The quartary carbon atoms are shown with no solvent accessible surface area, which means they are completely buried in the molecule leading to large Born radii.
CHAPTER 10

Calculation of Vibrational Frequencies

In this chapter, all necessary information about the calculation of vibrational spectra and thermostatistical contributions are given.

Contents

- Calculation of Vibrational Frequencies
  - Performing simple Vibrational Frequency calculations
  - Calculation of thermochemical properties
  - Dealing with imaginary modes and non-minimum structures
  - Advanced options

10.1 Performing simple Vibrational Frequency calculations

Vibrational frequency calculations are available only through two-sided numerical differentiation of analytical gradients.

Consider a simple example like the following hydrogen abstraction reaction:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-0.12888312425142</td>
<td>-0.00640246259879</td>
<td>-0.00997057133406</td>
</tr>
<tr>
<td>H</td>
<td>1.44011699709596</td>
<td>0.12229812355524</td>
<td>-0.02854203428735</td>
</tr>
<tr>
<td>H</td>
<td>-0.41612454870604</td>
<td>1.02694842152161</td>
<td>-0.04938812535015</td>
</tr>
<tr>
<td>H</td>
<td>-0.26306601703832</td>
<td>-0.58286887757121</td>
<td>-0.90445094952445</td>
</tr>
<tr>
<td>H</td>
<td>-0.26440375738028</td>
<td>-0.51708010658031</td>
<td>0.92386857306799</td>
</tr>
<tr>
<td>O</td>
<td>2.45008586500521</td>
<td>0.26032015001761</td>
<td>0.01133571198248</td>
</tr>
<tr>
<td>H</td>
<td>2.61210033905108</td>
<td>0.98358191645276</td>
<td>0.62026402303033</td>
</tr>
</tbody>
</table>
By invoking the `--hess` command line argument, xTB executes a calculation of the Hessian matrix. The `--ohess` keyword may be used instead if a prior optimization of the structure is desired.

```
xtb min.xyz --hess --uhf 1
```

At the end of the frequency job you get an output like this:

```
-------------------------------------------------
| Frequency Printout |
-------------------------------------------------
projected vibrational frequencies (cm⁻¹)
eigval : -0.00 -0.00 -0.00 0.00 0.00 0.00
eigval : 40.69 211.99 360.40 405.89 601.08 759.17
eigval : 829.07 1371.91 1375.70 1477.42 2297.37 3115.69
eigval : 3190.88 3197.05 3648.64
reduced masses (amu)
9: 1.45 10: 1.77 11: 1.92 12: 1.45 13: 2.39 14: 2.02 15: 2.03 16: 2.07
17: 2.17 18: 1.07 19: 2.09 20: 2.09 21: 1.86
IR intensities (amu)
1: 0.17 2: 0.46 3: 0.44 4: 0.41 5: 0.08 6: 0.35 7: 0.48 8: 0.26
9: 0.26 10: 0.27 11: 0.25 12: 0.41 13: 0.30 14: 0.04 15: 0.07 16: 0.52
17: 0.51 18: 0.06 19: 0.14 20: 0.12 21: 0.18
Raman intensities (amu)
1: 0.00 2: 0.00 3: 0.00 4: 0.00 5: 0.00 6: 0.00 7: 0.00 8: 0.00
9: 0.00 10: 0.00 11: 0.00 12: 0.00 13: 0.00 14: 0.00 15: 0.00 16: 0.00
17: 0.00 18: 0.00 19: 0.00 20: 0.00 21: 0.00
output can be read by thermo (oz use thermo option).
writing <g98.out> molden fake output.
recommended (thermochemical) frequency scaling factor: 1.0
```

This output consists of the calculated vibrational frequencies and the vibrational modes. In the example above there are six frequencies which are identically zero. These frequencies correspond to the rotations and translations of the molecule. They have been projected out of the Hessian before the calculation of the frequencies and thus, the zero values do not tell you anything about the quality of the Hessian that has been diagonalized.

xTB writes an `g98.out` file in GAUSSIAN-format, which can be opened with the popular MOLDEN program to visualize the vibrational modes. Further, a `hessian` file is written, containing the projected Hessian matrix in turbomole format.

### 10.2 Calculation of thermochemical properties

Each frequency job provides the thermochemical properties at 298.15 K. (for other temperatures, see below). No further user-input is required to obtain all important thermostatistical contributions. The contributions are calculated following a coupled rigid-rotor-harmonic-oscillator approach. If a molecular symmetry is detected, the resulting rotational number is automatically accounted for. The symmetry detection can be adjusted in the `$symmetry` block of the `xcontrol` file if necessary.
Multiple temperatures can be calculated using the build in thermodynamic functions calculator by using an input file similar to this

$thermo$
```
temp=150.0,200.0,250.0,273.15,298.15
```

The final summary looks like

<table>
<thead>
<tr>
<th>T/K</th>
<th>H(0)−H(T)+PV</th>
<th>H(T)/Eh</th>
<th>T*S/Eh</th>
<th>G(T)/Eh</th>
</tr>
</thead>
<tbody>
<tr>
<td>150.00</td>
<td>0.250495E-02</td>
<td>0.546739E-01</td>
<td>0.135034E-01</td>
<td>0.411705E-01</td>
</tr>
<tr>
<td>200.00</td>
<td>0.361203E-02</td>
<td>0.557809E-01</td>
<td>0.192424E-01</td>
<td>0.365386E-01</td>
</tr>
<tr>
<td>250.00</td>
<td>0.484240E-02</td>
<td>0.570113E-01</td>
<td>0.253913E-01</td>
<td>0.316200E-01</td>
</tr>
<tr>
<td>273.15</td>
<td>0.545010E-02</td>
<td>0.576190E-01</td>
<td>0.283634E-01</td>
<td>0.292557E-01</td>
</tr>
<tr>
<td>298.15</td>
<td>0.617016E-02</td>
<td>0.583013E-01</td>
<td>0.316937E-01</td>
<td>0.266076E-01</td>
</tr>
</tbody>
</table>

(used)
xtb will always use the last entry from the temperature list for all further calculations and printouts.

### 10.3 Dealing with imaginary modes and non-minimum structures

If a frequency calculation is invoked using the `--hess` command line argument, *xTB* automatically checks the gradient norm for a non-zero value. For unoptimized structures with significant remaining grad. norm, a warning is printed. If you want *xTB* to exit with an error code instead of this warning, use the `--strict` command line argument.

```
# WARNING! Some non-fatal runtime exceptions were caught, please check:
# - Hessian on incompletely optimized geometry!
```

A *xtbhess.coord* file is created in this case, containing the input structure distorted along the imaginary mode. In case of unwanted imaginary modes, this structure can be used as a starting point to perform further optimizations to get rid of the imaginary frequency and locate the true minimum.

### 10.4 Advanced options

Of course, the calculated frequencies depend on the masses used for each atom. Several options exist to modify/scale the default atomic masses in the `$hess` block of the *xcontrol* file.

```
$hess
  sccacc=real
      SCC accuracy level in Hessian runs
  step=real
      Cartesian displacement increment for numerical Hessian
  isotope: int,real
      set mass of atom type int to real
  modify mass: int,real
      set mass of atom number int to real
  scale mass: int,real
      scale mass of atom number int by real
```

Changes regarding `sccacc` or `step` should be made with caution, as large displacements or loose SCC accuracy can lead to unreliable frequencies due to excessive numerical noise in the calculations.

The thermostatistical calculations can be influenced by the `$thermo` block of the *xcontrol* file.

```
$thermo
  temp=real
      temperature for thermostatistical calculation (default: 298.15 K)
  sthr=real
      rotor cut-off (cm\(^{-1}\)) in thermo (default: 50.0)
```
In this chapter, all necessary information will be given in order to perform MD simulations with \textit{xTB}. The adjustable parameters will be discussed and a guide to how to change them will be given.

### Contents

- \textit{Molecular Dynamics Simulations}
  - General command-line control
  - Parameters
  - MD specific Files
    - Restart
    - Example/Case study

### 11.1 General command-line control

There are two main possibilities how to evoke a MD simulation. With the flag \texttt{--omd} geometry optimization will be performed and this structure then will be used for the MD simulation, a loose optimization level will be chosen.

```
> xtb coord --omd
```

By using the flag \texttt{--md} the MD simulation will be performed directly with the user given input structure.

```
> xtb coord --md
```

It is strongly recommended to start the MD simulation from an xTB optimized structure. Otherwise there may be instabilities during the MD and the equilibration will be severely hindered.
11.2 Parameters

In order to change the parameters of the MD simulation the $md block in the xcontrol/input file has to be modified.

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump</td>
<td>real</td>
<td>50 fs</td>
<td>interval for trajectory printout</td>
</tr>
<tr>
<td>lmass</td>
<td>integer</td>
<td>4 times</td>
<td>mass of hydrogen atoms</td>
</tr>
<tr>
<td>nvt</td>
<td>boolean</td>
<td>true</td>
<td>perform simulation in NVT ensemble</td>
</tr>
<tr>
<td>restart</td>
<td>boolean</td>
<td>false</td>
<td>read velocities from mdrestart</td>
</tr>
<tr>
<td>temp</td>
<td>real</td>
<td>298.15 K</td>
<td>thermostat temperature</td>
</tr>
<tr>
<td>time</td>
<td>real</td>
<td>50 ps</td>
<td>total run time of simulation</td>
</tr>
<tr>
<td>sccacc</td>
<td>real</td>
<td>2.0</td>
<td>accuracy of xTB calculation in dynamics</td>
</tr>
</tbody>
</table>
| shake | integer | 2       | use SHAKE algorithm to constrain bonds  
0 = off, 1 = X-H only, 2 = all bonds |
| step  | real  | 4 fs    | time step for propagation                       |
| velo  | boolean | false  | also write out velocities                        |

The above default setting should look like below in your input file

```
$md
  temp=298.15 # in K
  time= 50.0  # in ps
  dump= 50.0  # in fs
  step= 4.0  # in fs
  velo=false
  nvt =true
  lmass=4
  shake=2
  sccacc=2.0
$end
```

11.3 MD specific Files

After the xtb program has performed the desired MD simulation the trajectory of the structures can be found in xtb.trj. Furthermore, files with the names scoord.* are generated. After every picosecond of simulation the structure at this point will be written into these files. After a successful completion of the MD simulation a xtbmdok file will be touched. The structure and velocities at the end of the simulation will be written into a mdrestart file.

11.3.1 Restart

The mdrestart file can be used to restart an MD simulation. This can be very helpful for equilibration purposes. In order to achieve this, in the $md block the mdrestart parameter has to be set to true.

```
> cat xcontrol
$md
  mdrestart=true
```

Chapter 11. Molecular Dynamics Simulations
11.3.2 Example/Case study

To summarize the most important topics of this chapter we will perform an MD simulation of the ethane molecule with xTB. Make sure that xtb is properly set up and you have the following files in your working directory

```
> cat coord
$coord
 1.82409443250962 -0.02380488009596 0.17250251620479 c
 4.68095348739630 -0.02380488009596 0.17250308312263 c
 1.09744635742609 1.41159121722257 -1.12629926294082 h
 1.09744579050825 0.38329239564274 2.06499275150500 h
 1.09744635742609 -1.86629844212581 -0.42118612892243 h
 5.40760175145245 1.81868868193389 0.76619172824984 h
 5.40760212939767 -0.43090215583466 -1.71998734115020 h
 5.40760175145245 -1.45920097741449 1.47130486226824 h
$end
> cat xcontrol
$md
time=10
step=1
temp=500
shake=1
```

As you can see, we will run the simulation for 10 ps with a timestep of 1 fs at a temperature of 500 Kelvin. Furthermore, all hydrogen-containing bonds will be constrained using the SHAKE algorithm. To start the simulation we call xtb as follows

```
> xtb coord --input xcontrol --omd
```

The program will start with performing a geometry optimization, the optimized structure used to start the dynamic can be found and inspected in xtbopt.coord.

In the file xtb.trj we can find our trajectory. We can analyze the structures now by displaying them in a molecular graphics editor (e.g., MOLDEN, VMD etc.) or a trajectory analyzer (e.g., TRAVIS).
In this guide, all necessary information will be given in order to perform meta-dynamics (MTD) simulations with the xtb program. For the theory behind our MTD approach please refer to:


### Contents

- Meta-Dynamics Simulations
  - General MTD Setup
    - MTD specific Files
    - Restart
  - Example/Case study
  - Constrained MD/MTD simulations

From a practical point of view the application of meta-dynamics is quite similar to molecular-dynamic simulations. In MTD simulations a biasing potential given as a sum of Gaussian functions is additionally expressed. The root-mean-square deviation (RMSD) in Cartesian space is chosen as a metric for the collective variables.

All adjustable parameters will be discussed and a guide to how to change them will be given as well as an example.

## 12.1 General MTD Setup

For any MTD calculation a *Detailed Input* file is necessary to enter the correct calculation mode. The basic parameters for dynamics are taken from the $\texttt{md}$ block as described in the section regarding *Molecular Dynamics Simulations*. The $\texttt{metadyn}$ data group has to be present in the input file. All available instructions for this data group are shown here:
To avoid accidental activation of the bias potential conservative default values are chosen in the program. So you cannot simply use a commandline-only approach to perform a MTD calculation. First of all you want to create a \texttt{metadyn.inp} file with this content:

```plaintext
$metadyn
  save=10
  kpush=1.0
  alp=0.2
$end
```

You can start the metadynamic calculation now by using the \texttt{--md} commandline flag as:

```plaintext
> xtb --md --input metadyn.inp coord
```

By using the flag \texttt{--metadyn integer}, the number of saved structures may also be entered via the commandline and need not to be present in the detailed input:

```plaintext
> xtb --metadyn 10 --input metadyn.inp coord
```

### 12.1.1 MTD specific Files

After the \texttt{xtb} program has performed the desired MTD simulation the trajectory of the structures can be found in \texttt{xtb.trj}. Furthermore, files with the names \texttt{scoord.*} are generated. After every picosecond of simulation the structure at this point will be written into these files. After a successful completion of the MTD simulation a \texttt{xtbmdok} file will be touched. The structure and velocities at the end of the simulation will be written into a \texttt{mdrestart} file.

### 12.1.2 Restart

The \texttt{mdrestart} file can be used to restart an MTD simulation. This can be very helpful for equilibration purposes. In order to achieve this, in the \texttt{$md} block the \texttt{mdrestart} parameter has to be set to \texttt{true}.

```plaintext
> cat xcontrol
$md
  mdrestart=true
```

### 12.2 Example/Case study

To summarize the most important topics of this chapter we will perform an MTD simulation of the water dimer molecule with \texttt{xTB}. Make sure that \texttt{xtb} is properly set up and you have the following files in your working directory:
As you can see, we will run the MTD simulation for 10 ps with a timestep of 1 fs at a temperature of 298 Kelvin. For the meta-dynamics, only the structure of the first water molecule will be taken into account in the rmsd criteria. To start the simulation we call xtb as follows

```
> xtb --md --input metadyn.inp coord
```

The output for the example MTD simulation of the water dimer will look like this:

```
------------------------------------------------------------------------
| Meta Dynamics |
------------------------------------------------------------------------
trajectories on xtb.trj or xtb.trj.<n>
------------------------------------------------------------------------
MD time /ps : 10.00
dt /fs : 1.00
SCC accuracy : 1.00
temperature /K : 298.00
max steps : 10000
block length (av.) : 5000
dumpstep(trj) /fs : 100.00 100
dumpstep(coords)/fs: 1000.00 1000
H atoms mass (amu) : 2
# deg. of freedom : 14
SHAKE on. # bonds : 4 all: T
Berendsen THERMOSTAT on
kpush : 0.020
alpha : 1.200
update : 10
time (ps)  <Epot>  Ekin  <T>  T  Etot
     0  0.00  0.00000  0.0198  0.  0. -10.10916
est. speed in wall clock h for 100 ps : 0.01
200 0.20 -10.09118  0.0116  559.  524. -10.12881
400 0.40 -10.11436  0.0105  454.  471. -10.13041
600 0.60 -10.12671  0.0071  431.  321. -10.13081
800 0.80 -10.12671  0.0071  431.  321. -10.13081
       ...  ...  ...  ...  ...  ...
```

(continues on next page)
<xtb doc, Release 6.2>

(continued from previous page)

<table>
<thead>
<tr>
<th>Block</th>
<th>Epot</th>
<th>T</th>
<th>Ekin</th>
<th>Etot</th>
<th>Tbath</th>
</tr>
</thead>
<tbody>
<tr>
<td>4800</td>
<td>4.80</td>
<td>-10.13763</td>
<td>0.0084</td>
<td>469.379</td>
<td>-10.13198</td>
</tr>
<tr>
<td>5000</td>
<td>5.00</td>
<td>-10.13775</td>
<td>0.0082</td>
<td>465.368</td>
<td>-10.13253</td>
</tr>
<tr>
<td>5200</td>
<td>5.20</td>
<td>-10.13783</td>
<td>0.0129</td>
<td>469.582</td>
<td>-10.12808</td>
</tr>
<tr>
<td>5400</td>
<td>5.40</td>
<td>-10.13794</td>
<td>0.0105</td>
<td>471.474</td>
<td>-10.13014</td>
</tr>
<tr>
<td>5600</td>
<td>5.60</td>
<td>-10.13804</td>
<td>0.0090</td>
<td>470.407</td>
<td>-10.13140</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>9800</td>
<td>9.80</td>
<td>-10.13918</td>
<td>0.0083</td>
<td>462.376</td>
<td>-10.13258</td>
</tr>
</tbody>
</table>

Average properties:

- Epot: -10.1392169717059
- Epot (accurate SCC): -10.1402473210558
- Ekin: 1.019492766065306E-002
- Etot: -10.1290220440452
- T: 459.900938472654

Thermostating problem

Normal exit of md()

In the file xtb.trj we can find our trajectory. We can analyze the structures now by displaying them in a molecular graphics editor (e.g., MOLDEN, VMD etc.) or a trajectory analyzer (e.g., TRAVIS).

### 12.3 Constrained MD/MTD simulations

As you may have noticed in the example given above by checking the file xtb.trj, the water dimer dissociates within the MTD simulation due to the applied bias potential. If you run dynamics for systems that are non-covalently bound, you may encounter this problem from time to time. To avoid dissociation you can try to confine the simulation in a sphere by a repulsive potential. For further details check how to confine a cavity in Detailed Input.

To avoid dissociation of the water dimer by a logfermi potential, the input file has to be modified:

```bash
> cat metadyn.inp
$md
    time=10
    step=1
    temp=298
$end
$metadyn
    atoms: 1-3
    save=10
    kpush=0.02
    alp=1.2
$end
$wall
    potential=logfermi
    sphere: auto, all
$end
```

To start the constrained MTD simulation we call xtb as follows:

```bash
> xtb --md --input metadyn.inp coord
```

If you now check the trajectory file, you will see that the water molecules do not separate.

**Note:** The wall potential does not only work for MD/MTD simulations. It may also be applied in the same manner.
for single point calculations and geometry optimizations.
13.1 Input

\texttt{gsm} is a method to find a reaction path and a transition state. In the following, we are going to work with the Double-ended Growing String Method (DE-GSM), therefore a converged start and end structure is needed. The atomic order needs to be the same in both files otherwise the \texttt{gsm} calculation will not give the transition state you are looking for.
Let’s first have a look at the structure needed. The files are explained later in detail. Let’s call the directory, where you want to execute your calculations, `cwd/` (current working directory). In `cwd/` you need to have the directory `scratch/`. Here, all files created and needed during the calculation are saved, but in the beginning, there is only one file in `scratch/`, named `initial0000.xyz`. The file `initial0000.xyz` contains the start and the end structure in any valid Xmol format. The next two files needed are in the directory, where you want to carry out your calculation. Both are available upon personal request or from the ZimmermanGroup (see links attached). With the `inpfileq` the user is able to set specific parameters for the `gsm` calculation, whereas the `ograd` wraps `xtb` and converts input and output for `gsm`. This is necessary, as `gsm` can read `orca` output but not `xtb` output. To use `gsm` with `xtb`, we therefore have to fake an `orca` output, which is done using the `tm2orca` script. Furthermore, you need two directories, where you optimize your start and end structure, namely `start/` and `end/`.

```
> cd cwd/
> ls
end/  inpfileq  ograd*  scratch/  start/
```

### 13.2 Inversion

This example deals with the inversion of cyclohexane from the chair to the boat conformation. Firstly, you need a converged starting structure `start.xyz`. You therefore have to build your molecule in `start/`, where you build your molecule using a smiles string, `avogadro` or any other graphical program of your choice. Afterwards you have to execute a quick geometry optimization (e.g. with `xtb`) and copy the obtained converged structure back in your `cwd/`.

```
> cd start/
> obabel -ismi unconverged.smi --gen3d -oxyz -O unconverged.xyz
> xtb unconverged.xyz --opt
> cat xtbopt.xyz
18
-19.1953921
C 0.72843434470456 1.25982478073651 -0.24012456780476
C -0.72777610989508 1.26067546579378 0.23600481175104
C -1.45606227367562 -0.00012407730597 -0.24050672835883
C -0.72883134272817 -1.26099459852248 0.23739090970526
C 0.72737798955291 -1.26188502513029 -0.23874262122560
C 1.45568794883483 -0.00106824921650 0.23769683369676
H 0.75509375057143 1.30484737240468 -1.33473424471687
H 1.24080411887910 2.14749990671418 0.14406690070669
H -1.23941415236514 2.14836879734283 -0.14911579915314
H -0.75440938068291 1.30686679161267 1.33056700638662
H -2.48123885422840 0.00052444409439 0.14306417819097
H -1.50769080223365 -0.0069664832176 -1.33514006744288
H -0.7548303390087 -1.30594264398711 1.3320209562400
H -1.2412058934463 -2.14869071264549 -0.14673782557395
H 1.23901244269262 -2.14956140418753 0.14642959499471
H 0.7541085032508 -1.30814283105027 -1.33303896959568
H 1.50743880779513 -0.00049661774011 1.33233065893244
H 2.48082883569879 -0.00171475059154 -0.14597723871678
> cp xtbopt.xyz ../start.xyz
```

Obviously, the `xtb` calculation can be done using all flags explained in this documentation, e.g. `--chrg`, `--uhf`, `--gbsa` and so on. The same has to be done with the end structure. It is advisable to take the optimized start structure and change, whatever you want to change, using a graphical program, which does not change your atomic order, e.g. `avogadro`, as a change in the atomic order will definitely cause problems during the DE-GSM calculation!

Before you can start the calculation, a couple of other things have to be done. First, you have to generate your `initial0000.xyz`. 


Then you have to modify your input file. Normally, all default values can be used, and you only have to care about the last two entries TS_FINAL_TYPE and NNODES. TS_FINAL_TYPE can be 0 or 1. 0 means no bond breaking and is used for an inversion, whereas you have to use 1 for a bond breaking. If you use the wrong setting here, for example 1 for the inversion of cyclohexane, gsm tries to break a bond leading to a wrong path. NNODES is the maximum number of nodes for the DE-GSM calculation and should be set to 15 for xtb.

<table>
<thead>
<tr>
<th>TS_FINAL_TYPE</th>
<th>0</th>
<th># any/delta bond: 0/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNODES</td>
<td>15</td>
<td># including endpoints</td>
</tr>
</tbody>
</table>

Last, you have to modify the xtb call in ograd*. The $ofile.xyz as well as the --grad flag are necessary, but you have to modify e.g. your charge or gbssa flag. In the case of cyclohexane, the charge is 0 and for simplifications I just calculate it in gasphase, therefore no gbssa is used.

13.2. Inversion 71
Now, you have done everything to start the calculation.

After the calculation, the two most important files are the reaction path in your cwd/, called stringfile.xyz0000, and the transition state in scratch/tsq0000.xyz, both in a valid Xmol format. The reaction path of the Inversion of cyclohexane can be seen below.

![Fig. 1: Inversion of cyclohexane](image1)

Fig. 1: Inversion of cyclohexane

![Fig. 2: Energy diagram of the inversion of cyclohexane](image2)

**13.3 Bond breaking**

The next example is a simple Claisen rearrangement of an allyl vinyl ether and consequently includes a bond breaking and building. The initial0000.xyz is build as described above by writing the converged start and end structure on after the other.

<table>
<thead>
<tr>
<th>C</th>
<th>0.34045581</th>
<th>-0.40506398</th>
<th>0.07097230</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0.11887830</td>
<td>0.26450745</td>
<td>1.37067084</td>
</tr>
<tr>
<td>H</td>
<td>1.33494198</td>
<td>-0.62381082</td>
<td>-0.28316830</td>
</tr>
</tbody>
</table>
Next, the `inpfileq` is modified. As we are now dealing with a bond breaking, the `TS_FINAL_TYPE` has to be adapted. The `NNODES` is also changed to a higher value to give a more detailed reaction path. This is not necessary and was just done for a nicer movie and a nicer energy diagram.

<table>
<thead>
<tr>
<th>TS_FINAL_TYPE</th>
<th>1</th>
<th># any/delta bond: 0/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNODES</td>
<td>20</td>
<td># including endpoints</td>
</tr>
</tbody>
</table>

At the end, the `ograd*` has to be modified. As Claisen rearrangements are often done in polar solvents, and a water / ethanol mixture accelerates the reaction, the calculation was done using `gbsa(water)`.

```
xtb $ofile.xyz --grad --chrg 0 --gbsa h2o > $ofile.xtbout
tm2orca.py $basename
```

Now, the `gsm` calculation is done

```
>gsm.orca
```

The reaction path as well as the energy diagram are given below.

Fig. 3: Reaction path of a claisen rearrangement

### 13.4 Wrong atomic order

The following is an example that shows how important a proper atom order is. It deals with the same Claisen rearrangement as shown above, but with a different atom order in the start and end structure file, as shown below.
Except for the different atom order the same as above was done. Both structures are written to the initial0000.xyz in the scratch/* directory. In the inpfile the TS_FINAL_TYPE is 1, and the NNODES is set to 20. The xtb call in ograd* is given below:

```bash
xtb $ofile.xyz --grad --chrg 0 --gbsa h2o > $ofile.xtbout
```

Now gsm is just started as already shown.

```
> gsm.orca
```

The resulting path as well as the energy diagram is shown below.

---

Fig. 4: Energy diagram of a wrong reaction path

Fig. 5: vimdiff of different atomic order in the start (left) and end (right) file

Fig. 6: Reaction path of a claisen rearrangement with wrong atom order
Fig. 7: Example of an energy diagram of a wrong reaction path
Periodic Boundary Conditions

Periodic xTB calculations are either possible with the standalone xtb program or the atomic simulation environment (ASE) using the libxtb.so and the C-API (see Using xtb in Python).

xtb is supposed to throw a lot of feature-not-implemented errors at you as you try out the very recently added periodic boundary conditions. You can make use of the C-API and the ASE calculator class to get around this errors for now, but we promise to add more features and runtypes in the future.

Contents

• Periodic Boundary Conditions
  – Input Formats
  – Geometry Optimizations
    * With the Atomic Simulation Environment

14.1 Input Formats

We support Turbomole’s coordinate files in a riper compatible format. It requires to have the $periodic information, one of $lattice or $cell and, of course, a $coord data group present. In contrast to Turbomole we want them all in one file, so the file=<elsewhere> does not work with xtb.

A valid input for diamond is

```
$periodic 3
$cell angs
  3.570  3.570  3.570  90 90 90
$coord angs
  0.00000  0.00000  0.00000  C
  0.89250  0.89250  0.89250  C
  1.78500  1.78500  0.00000  C
```

(continues on next page)
or using different keywords and order like for calciumfluoride here

```
$coord frac
0.250000000 0.250000000 0.250000000 f
0.750000000 0.750000000 0.750000000 f
0.000000000 0.000000000 0.000000000 ca
$lattice angs
3.153833580 1.115048556 1.931320751
0.000000000 3.345145667 1.931320751
0.000000000 0.000000000 3.862641503
$periodic 3
$end
```

**Note:** we do not care if you end your file or maybe even all your data groups, since the parser politely ignores its presence.

While this format in principle is able to specify also 1D and 2D periodic systems xtb does not support them right now.

Both Vasp 4 and Vasp 5 POSCAR and CONTCAR files are supported, but we require to have the information on the atomtypes present in the file. For details on the format refer to the documentation of Vasp.

**Tip:** You can use ase convert to bring your cif or fort.34 files into Vasp format, as xtb currently cannot read them.

### 14.2 Geometry Optimizations

**Note:** feature implemented in version 6.2

To perform geometry optimizations with xtb on periodic systems we recommend to use an input file like

```
$opt
  engine=inertial
$end
```

Since the ANC optimizers do not support periodic boundary conditions right now, use the inertial relaxation procedure instead.

The optimization log is written in Vasp 5 POSCAR format and contains the current energy and gradient norm in the first (comment) line.
14.2.1 With the Atomic Simulation Environment

**Note:** This guide assumes that you were able to acquire the shared-library and to include it and the wrapper script to your systems path variables.

On its own the xtb.py wrapper is able to perform geometry optimizations using the preconditioned FIRE optimizer as implemented in the atomic simulation environment (ASE). We patch the Optimizer-Class to make sure that the convergence thresholds are tight enough and correspond to normal convergence thresholds used in the xtb standalone.

To start a geometry optimization from a POSCAR like

```
C
1.0
 3.570  0.000  0.000
 0.000  3.570  0.000
 0.000  0.000  3.570

8
Cartesian
 0.00000  0.00000  0.00000
 0.89250  0.89250  0.89250
 1.78500  1.78500  0.00000
 2.67750  2.67750  0.89250
 1.78500  0.00000  1.78500
 2.67750  0.89250  2.67750
 0.00000  1.78500  1.78500
 0.89250  2.67750  2.67750
```

call xtb.py as follows

```
> xtb.py POSCAR --optcell --precon --logfile --trajectory peeqopt.traj
Initial energy: eV, Eh -456.729799615 -16.78451069
preconFIRE: 0 14:22:01 -456.729800 0.0000 0.1207
preconFIRE: 1 14:22:01 -456.731178 0.0000 0.1200
...  
preconFIRE: 25 14:22:02 -456.843257 0.0000 0.0023
preconFIRE: 26 14:22:02 -456.843269 0.0000 0.0018
preconFIRE: 27 14:22:02 -456.843280 0.0000 0.0013
Final energy: eV, Eh -456.843279887 -16.7886810131
> ls
xtbopt.POSCAR  xtbopt.traj  xtb.out  POSCAR
```

After the optimization you find the optimized structure in xtbopt.POSCAR and the details on the last calculation in xtb.out. The optimization can be viewed by opening the trajectory-file using ase gui.
xtb supports external electrostatic potentials for GFN1-xTB and GFN2-xTB.

### 15.1 Example: The Water Tetramer in Pieces

As input geometry for the QM half of the water cluster we use

```
$coord
-2.75237178376284  2.43247309226225  -0.01392519847964  O
-0.93157260886974  2.79621404458590  -0.01863384029005  H
-3.43820531288547  3.30583608421060  1.42134539425148  H
-2.43247309226225  -2.75237178376284  0.01392519847964  O
-2.79621404458590  -0.93157260886974  0.01863384029005  H
-3.30583608421060  -3.43820531288547 -1.42134539425148  H
$end
```

The setup will look somewhat similar to this

```
> ls
pcem.input  water_4.coord  water_4.pc
> cat pcem.input
$embedding
  input=water_4.pc
$end
$end
> xtb water_4.coord -I pcem.input
...
The file `water_4.pc` contains the partial charges and its positions as

```
6
-0.69645733  2.75237178376284 -2.43247309226225 -0.01392519847964  O
0.36031084   0.93157260886974 -2.79621404458590 -0.01863384029005  H
0.33614649   3.43820531288547 -3.30583608421060  1.42134539425148  H
-0.69645733  2.43247309226225  2.75237178376284  0.01392519847964  O
0.36031084   2.79621404458590  0.93157260886974  0.01863384029005  H
0.33614649   3.30583608421060  3.43820531288547 -1.42134539425148  H
```

The first column contains the partial charge, the second to fourth columns contain the cartesian coordinates in Bohr (or in Ångström if `interface=orca` is used in the input file). The fifth column is optional, but can contain, like here, element symbols to specify the chemical hardnesses of the partial charges. Note that we are not using real point charges here but a damped Coulomb interaction consistent to the electrostatic interactions used in the respective xTB Hamiltonian.

The read in point charges are shown in the setup block of the SCC as

```
... skip ...
```

```plaintext
-----------------------------------------
| Self-Consistent Charge Iterations      |
-----------------------------------------

<table>
<thead>
<tr>
<th>Number</th>
<th>E</th>
<th>dE</th>
<th>RMSdq</th>
<th>gap</th>
<th>omega</th>
<th>full diag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-10.2141283</td>
<td>-0.102141E+02</td>
<td>0.418E+00</td>
<td>13.25</td>
<td>0.0</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>-10.2169842</td>
<td>-0.285597E-02</td>
<td>0.241E+00</td>
<td>13.15</td>
<td>1.0</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>-10.2175075</td>
<td>-0.523314E-03</td>
<td>0.392E-01</td>
<td>12.35</td>
<td>1.0</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>-10.2176392</td>
<td>-0.131616E-03</td>
<td>0.871E-02</td>
<td>12.57</td>
<td>1.0</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>-10.2176471</td>
<td>-0.790990E-05</td>
<td>0.417E-02</td>
<td>12.50</td>
<td>1.0</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>-10.2176488</td>
<td>-0.172257E-05</td>
<td>0.237E-03</td>
<td>12.52</td>
<td>17.3</td>
<td>T</td>
</tr>
<tr>
<td>7</td>
<td>-10.2176488</td>
<td>-0.524712E-08</td>
<td>0.835E-04</td>
<td>12.52</td>
<td>48.9</td>
<td>T</td>
</tr>
<tr>
<td>8</td>
<td>-10.2176488</td>
<td>0.134836E-09</td>
<td>0.623E-04</td>
<td>12.52</td>
<td>65.5</td>
<td>T</td>
</tr>
</tbody>
</table>

*** convergence criteria satisfied after 8 iterations ***
```

(continues on next page)
... skip ...

<table>
<thead>
<tr>
<th>total energy</th>
<th>-10.155934019173 Eh</th>
</tr>
</thead>
<tbody>
<tr>
<td>gradient norm</td>
<td>0.024959693967 Eh/a0</td>
</tr>
<tr>
<td>HOMO-LUMO gap</td>
<td>12.520985724021 eV</td>
</tr>
</tbody>
</table>

SCC energy -10.217648797540 Eh

- isotropic ES -0.066792092832 Eh
- anisotropic ES -0.002669047466 Eh
- anisotropic XC -0.001310421597 Eh
- dispersion -0.000808230531 Eh
- repulsion energy 0.061714760838 Eh
- add. restraining 0.000000000000 Eh

... skip ...

To obtain point charge like behaviour for the partial charges the chemical hardness can be set to a large value. This can be done by specifying the chemical hardnesses in the fifth column instead of giving an element symbol. For this setup the `water_4.pc` would look like

<table>
<thead>
<tr>
<th>6</th>
<th>0.36031084</th>
<th>0.33614649</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.69645733</td>
<td>0.93157260886974</td>
<td>-2.79621404458590</td>
</tr>
<tr>
<td>0.33614649</td>
<td>3.43820531288547</td>
<td>-3.03583608421060</td>
</tr>
<tr>
<td>-0.69645733</td>
<td>2.43247309226225</td>
<td>-2.75237178376284</td>
</tr>
<tr>
<td>0.36031084</td>
<td>2.79621404458590</td>
<td>0.93157260886974</td>
</tr>
<tr>
<td>0.33614649</td>
<td>3.30583608421060</td>
<td>3.43820531288547</td>
</tr>
</tbody>
</table>

15.1. Example: The Water Tetramer in Pieces
CHAPTER 16

Introduction to CREST

Contents

• Introduction to CREST
  – What is CREST?
  – Conformers and Rotamers
  – Conformational Search Algorithms
    * MF-MD-GC (V1)
    * MTD-GC (V2) / iMTD-GC (V2i)

16.1 What is CREST?

CREST is an utility/driver program for the xtb program. Originally it was designed as conformer sampling program, hence the abbreviation Conformer–Rotamer Ensemble Sampling Tool, but now offers also some utility functions for calculations with the GFN$n$–xTB methods. The key procedures implemented in CREST are two different conformational search workflows abbreviated as MF-MD-GC (V1) and MTD-GC (V2). The older of these procedures, MF-MD-GC, combines a mode following (MF), molecular dynamics sampling (MD), and genetic z-matrix crossing (GC) approach for the generation of conformer/rotamer ensembles (CREs). The newer workflow, MTD-GC, makes use of extensive metadynamic sampling (MTD), with an additional GC step at the end. Of the two algorithms, MTD-GC showed significant improvements while being slightly more expensive compared to MF-MD-GC for large cases. Other functionalities that are included in CREST are parallel optimization (MDOPT) and screening (SCREEN) functions for GFN$n$–xTB, a Z-matrix sorting function (ZSORT) and a function to sort (e.g. for NMR equivalencies) externally created CREs (CREGEN). Following version 2.7 of CREST, the program also includes some automated procedures for the protonation, deprotonation and tautomerization of structures.
16.2 Conformers and Rotamers

The thermally accessible ensemble of minimum-energy structures generally consists of conformers as well as rotamers, and hence can be called a conformer/rotamer ensemble (CRE). Many physical observables are obtained as time average over the different low energy conformations of a molecule. Hence, in computational chemistry, it can be important to perform calculations on the favored conformation or an ensemble of conformations. Some examples for macroscopic observables that are depending on the conformation are pKa values, CD spectra, NMR spectra, IR spectra, etc. While calculations have generally only to be performed for different conformers (each having a distinct energy minimum, see figure below), rotamers become important, e.g., for the calculation of NMR spectra where they represent the rapid (sub-experiment time scale) nuclei interchange leading to an average of NMR parameters.

Rotamers also contribute substantially to the molecular entropy and the completeness of the overall CRE. This can be assessed by a maximized entropy $S_{CR}$ according to the standard thermodynamic expressions

$$S_{CR} = R \sum_{i=1}^{CRE} p_i \log p_i,$$

where $R$ is the molar gas constant and the sum runs over all populations $p_i$ of all species with energy $\Delta E_i$ at temperature $T$, given as

$$p_i = \frac{\exp(-\Delta E_i/RT)}{\sum_{j=1}^{CRE} \exp(-\Delta E_j/RT)}.$$

The ensemble entropy $S_{CR}$ is also linked to the ensemble free energy (at $T = 298$ K) $G_{CR} = -T S_{CR}$, which has to be minimized for a complete CRE.

In practice we use three descriptors to distinguish between conformers and rotamers: The energy, the rotational constant of the molecule and the RMSD between two molecules. If two structures have completely different energies they are conformers. If they have the same energy (within a chosen threshold) they could be either two different conformers with similar energies, two rotamers of the same molecule or a duplicate of the same rotamer. In the first case (two conformers with similar energies) the RMSD will be high and the rotational constant will be different. For two rotamers the RMSD will be high, but the rotational constant is the same (within a chosen threshold). Enantiomers (mirror image molecules) are a special case of rotamers matching the same criterion. Only for duplicates of the same rotamer (that have to be sorted out) the RMSD, the rotational constants, and the energies will be the same.
16.3 Conformational Search Algorithms

16.3.1 MF-MD-GC (V1)

The MF-MD-GC workflow is a composite procedure consisting out of mode following, molecular dynamics sampling and genetic z-matrix crossing. The idea here is to combine different approaches of conformer sampling to obtain good and complete CREs. Low-lying, thermally accessible conformers and rotamers can be obtained from normal modes (NMs) oder localized modes (LMs) by creating a one-dimensional PES by displacement along the respective low-frequency modes and determining the minima on this PES. This MF is a physically plausible approach for conformer sampling, since the transition pathways of one conformer into another is always connected to the natural movement of the molecule, i.e., the modes. If new, lower lying conformers are found in one of the modefollowing steps the procedure is iteratively restarted with these conformers, which can be seen as some kind of variational approach to finding new conformations.

Molecular dynamics sampling (MD) is a common approach to the generation of conformations. Here a short MD simulation is conducted and snapshots are equidistantly taken from the trajectory, which are then surpassed to a geometry optimization. For the MF-MD-GC workflow it was shown that not many new conformers are generated in this step. It is however crucial to the generation of rotamers. Since the time scales of small rotations that lead to new rotamers (e.g. phenyl- and methyl-group rotations) are typically much shorter than the total MD simulation time, a single MD at higher temperature (400 K) showed to be sufficient for this step.

Genetic Z-matrix crossing (GC) is related to the concept of genetic algorithms in such that structural elements present only in already generated structures are projected onto a reference to create new structures. By repeating the crossing procedures structural elements that appear more frequently would be inherited more often, being responsible for the ‘genetic’ character of this approach. Internal (Z-matrix, $R$) coordinates are employed and a new structure is generated by taking the differences to the reference $R_{ref}$ over all internal coordinates (i.e., bond length, bond angles, an dihedral angles) according to

$$R_{new} = R_{ref} + R_i - R_j ,$$

where $R_i$ and $R_j$ label the pairs and $R_{new}$ is the generated new structure, which is subjected to a full geometry optimization. In this way, structural differences, e.g. a methyl group rotation, relative to $R_{ref}$ present only in $R_i$ and $R_j$ are combined in the resulting new conformer/rotamer.
The whole workflow is sketched in the figure below. Each run is started off with a hessian calculation to get the normal and localized modes for the modefollowing. MF is then performed up to 5 times with different settings for the number of points on the mode, step width between points on the mode and mode update factors. Should a better conformer be found between any of these steps, the procedure is restarted. After the MF a single MD simulation is performed to get the rotamer degeneracies. In the last step the GC is performed with the CRE that was found up to this point.

16.3.2 MTD-GC (V2) / iMTD-GC (V2i)

The MTD-GC workflow was designed to find low lying conformers more efficiently and more safely than the older MF-MD-GC algorithm. Furthermore this new algorithm is more robust and general applicable than more complicated schemes since it does not require any pre-definition of special system coordinates. MTD-GC is rooted in the basic
idea to combine GFNn–xTB calculations with root-mean-square-deviation (RMSD) based meta-dynamics (see section Meta-Dynamics Simulations). In practice a history-dependent biasing potential is applied, where the collective variables (CVs) for the meta-dynamics are previous minima on the PES, expressed as RMSD between the structures. The biasing contribution is given by Gaussian-type potential as

\[ V_{bias} = \sum_i^n k_i \exp(-\alpha \Delta_i^2), \]

where the RMSD enters as collective variables \( \Delta_i \), \( n \) is the number of reference structures, \( k_i \) are the pushing strengths and the parameter \( \alpha \) determines the potentials’ shape. From this energy expression atomic forces are derived that enter as additional forces in the MTD simulations (in the context of meta-dynamics also sometimes referred to as guiding forces). Since the addition of each bias Gaussian drives the structure further away from previous geometries this allows otherwise unlikely high-barrier crossings where all atoms collectively explore huge regions of the potential energy surface.

The GC was included in the MTD-GC procedure for the same reasons it was included in the MF-MD-GC workflow. The ensemble can be improved regarding the rotamers efficiently by the Z-matrix crossing. This effect is best visible for acyclic chains with a number of rotateable bonds, e.g., alkanes, but in principle it also works for more complicated cases, such as macrocyclic systems.

In practice the MTD simulation length is determined automatically by a flexibility measure of the molecule (typically \( t = 0.3 - 0.4 \times N \) ps per MTD). Several independent MTDs (at 300 K) are performed with different settings for \( \alpha \) (in Bohr\(^{-1}\)) and \( k_i/N \) (in \( mE_h \)). This has to be done since each molecule in principle requires a unique set of optimal \( \alpha \) and \( k \) and thus a variety of parameters ensures that the algorithm is performing well for all types of molecules. The snapshots are geometry optimized in a multi-level, three-step-filtering procedure by firstly applying two loose threshold settings followed by very tightly converged optimization and energy windows of 15, 10, and 6 kcal/mol, respectively. After the second step of this filtering also some short regular MD simulations are performed on the 6 lowermost conformers (at different temperatures 400 and 500 K), which is done to A) get rotamers and B) more extensively sample around these minima on the PSE (i.e., find low-barrier conformers missed by the high-energy MTD treatment). In the last step the GC procedure is performed to further complete the CRE. The number of generated structures in this step is limited to \( \min(3000, t \times 50) \) in order to limit the computational cost. Furthermore a two-step-filtering procedure is used to optimize the generated geometries, similar to the three-step-filtering before.

**Note:** The new MTD-GC algorithm is much better than the MF-MF-GC workflow in regards of finding low-energy conformations and complete CREs and as such replaces it as default runtype of CREST. Additionally it is much more streamlined from the technical point of view and its implementation makes full use of OMP parallelization (parallelization on a single computer/CPU node).

Following version 2.6 of CREST an iterative version of the MTD-GC workflow (called iMTD-GC) is the default runtype. In this slightly different scheme a fewer number of MTDs is conducted, but if a new lower conformer is found the procedure is restarted with this conformer as an input. The process is also restarted if a better conformer is found after the normal MD sampling around the lowest conformers or the GC. Compared to the regular MTD-GC workflow the optimization thresholds are set differently. Hence, for typical drug sized molecules the total CPU time of the iMTD-GC workflow is approximately the same as with MTD-GC, while better CREs are produced. All CREs that are found within the iterations are included in the conformer/rotamer ranking process. The iMTD-GC workflow is outlined graphically in the figure below.

16.3. Conformational Search Algorithms 89
CREST is usually invoked via commandline, and requires only a coordinate input file. The program supports the TURBOMOLE coordinates (coord) and Xmol (*.xyz) format and can be called via

```bash
> crest [INPUT] [OPTIONS]
```

If no file is given as [INPUT], then CREST automatically searches for a file called `coord` in the TURBOMOLE format. The different [OPTIONS] are listed below and refer to Version 2.7 of the CREST code.

### 17.1 Runtypes

Several different applications are available within the CREST program. The most important usage are the two different conformational search algorithms MF-MD-GC and iMTD-GC, but there are also some smaller utility tools that can be used, such as an CRE sorting function (CREGEN), or a standalone z-martix sorting function (ZSORT). The different runtypes are:
MF-MD-GC algorithm
flag -v1
description First generation of the GFN\textit{n–xTB} driven conformational search algorithm, consisting out of mode following, molecular dynamics sampling and genetic structure crossing.

MTD-GC algorithm
flag -v2
description Second generation of the GFN\textit{n–xTB} driven conformational search algorithm, consisting out of a meta-dynamics approach and genetic structure crossing.

iMTD-GC algorithm [DEFAULT]
flag -v2i, -v3
description Iterative version of the MTD-GC workflow, which is the default runtype of CREST.

CREGEN ensemble sorting tool
flag -cregen <FILE>
description Tool to sort a given ensemble <FILE> according to energy, atomic RMSD and rotational constant. A reference structure (e.g. coord) has to be provided.

ZSORT z-matrix sorting tool
flag -zsort
description The atom order of the given input file is sorted in order to yield a more consistent z-matrix, i.e., atoms are grouped together according to the molecular structure (e.g. methyl groups).

MDOPT parallel ensemble optimization
flag -mdopt <FILE>
description Optimize each point on a given trajectory or ensemble file <FILE> with GFN\textit{n–xTB}.

SCREEN ensemble screening tool
flag -screen <FILE>
description Optimize each point on a given trajectory or ensemble file <FILE> with GFN\textit{n–xTB} in a multilevel approach and sort the resulting ensemble (CREGEN).

Other structure screening modes
flag -protonate
description A tool that can be used to find protonation sites, i.e., the protomers of the input structure. In the approach first localized molecular orbitals (LMOs) are calculated and LP- and \(\pi\)-centers are identified. Then, a proton is added to each of these centers and the resulting structures are optimized and sorted.

flag -deprotonate
description A tool to find deprotomers of the input structure. Each H atom is removed and the resulting structures are optimized and sorted.

flag -tautomerize
description A tool that combines the -protonate and -deprotonate options to find (prototropic) tautomers of the input structure.
17.2 Options

17.2.1 General Options

- **-h, -help**  show help page
- **-chrg INT** specify molecular charge as \( \text{INT} \), overrides .CHRG file
- **-uhf INT** specify \( N_\alpha - N_\beta \) as \( \text{INT} \), overrides .UHF file
  - **-gfn0** use GFN0-xTB
  - **-gfn1** use GFN1-xTB
- **-gfn2 [DEFAULT]** use GFN2-xTB, which is the default.
- **-g, -gbsa SOLVENT** generalized born (GB) model with solvent accessible surface (SASA) model, available solvents are acetone, acetonitrile, benzene (only GFN1-xTB), CH2Cl2, CHCl3, CS2, DMF (only GFN2-xTB), DMSO, ether, H2O, methanol, n-hexane (only GFN2-xTB), THF and toluene. The solvent input is not case-sensitive.
- **-opt LEVEL** set the optimization accuracy for final GFN\( n \)-xTB optimizations. See Geometry Optimization for valid LEVEL arguments. The [DEFAULT] is vtight.
- **-zs [DEFAULT]** perform z-matrix sorting (i.e. ZSORT) for the input coordinate file.
  - **-nozs** do not perform z-matrix sorting of the input file.
- **-ewin REAL** set the energy threshold to \( \text{REAL} \) kcal/mol. This affects several runtypes and the [DEFAULT] is depending on the application (6 kcal/mol conformational searches, 30 kcal/mol screening tools).
- **-xnam BIN** specify the name (and path) of the xtb binary that should be used as BIN. The [DEFAULT] is xtb.
  - **-prsc** create a scoord.* file for each conformer in the TURBOMOLE format.
  - **-niceprint** in-line progress bar printout for optimizations.
  - **-T INT** specify the number of CPU threads \( \text{INT} \) that shall be used. CREST automatically adjusts the number of processes according to this variable in each step, in order to achieve optimal parallelization of the calculations.

17.2.2 Options for MF-MD-GC

- **-nomf** skip mode following
- **-nomd** skip MD part
- **-nocross** skip genetic crossing part.
- **-loose** decrease used number of selected modes
- **-vloose** decrease used number of selected modes a lot
- **-tight** increase used number of selected modes
- **-mdlen, -len REAL** set length of the molecular dynamics simulation to \( \text{REAL} \) ps. The [DEFAULT] is 40 ps.
- **-shake INT** set SHAKE mode for MD. \( \text{INT} \) can be 0 (= off), 1 (= H-only), 2 (= all bonds). The [DEFAULT] is 2.
- **-quick** conduct only one MF/MD (no GC) run to obtain a crude conformer ensemble.
17.2.3 Options for iMTD-GC

-**cross** [DEFAULT] do the genetic structure crossing (GC) part.
-**nocross** don’t do the GC part.
-**mrest** INT maximum number of MTD restarts in iMTD-GC algorithm. The [DEFAULT] is 5 cycles.
-**shake** INT set SHAKE mode for MD. INT can be 0 (= off), 1 (= H-only), 2 (= all bonds) The [DEFAULT] is 2.
-**tstep** INT set MD time step to INT fs. The [DEFAULT] is 5 fs.
-**mdlen, -len** REAL set length of the meta-dynamics simulations (MTD) to REAL ps. The [DEFAULT] is depending on the size and flexibility of the system.
-**mddump** INT set dumpstep in which coordinates are written to the trajectory file to INT fs. The [DEFAULT] is 100 fs.
-**vbdump** REAL set dump frequency in which a new reference structure is taken for $V_{bias}$ to REAL ps. The [DEFAULT] is 1.0 ps.
-**tnmd** REAL set temperature for the additional normal MDs on the lowest conformers after the MTD step. The [DEFAULT] is 400 K.
-**norotmd** don’t do the additional MDs on the lowest conformers after the MTD step.
-**quick** perform a search with reduced settings for a crude conformer ensemble.
-**sqquick, -superquick** perform an even more crude conformational search than with -quick.
-**origin** [DEFAULT] track the step of generation for each conformer/rotamer.
-**keepdir** keep sub-directories of the conformer production run.
-**nci** specialized NCI mode that can be used to find aggregates of NCI complexes. The option generates an ellipsoid potential around the input structure and adds it to the MTD simulation. Also, settings for $k$ and $\alpha$ are adjusted and some settings are reduced, in order to achieve lower computation times.
-**wscal** REAL scale the ellipsoid potential axes in the NCI mode by factor REAL.

17.2.4 Options for CREGEN

Note: The CREGEN routine is also used to sort in between the steps of the conformational searches. Therefore the following options also affect the performance of the two conformer algorithms.

-**rthr** REAL set RMSD threshold in Ångström. The [DEFAULT] is 0.125 Å.
-**ethr** REAL set energy threshold between conformer pairs in kcal/mol. The [DEFAULT] is 0.10 kcal/mol.
-**bthr** REAL set Rotational constant threshold to REAL. The [DEFAULT] is 0.02.
-**athr** REAL similarity threshold to determine internal rotation equal atoms for NMR. The [DEFAULT] is 0.04.
-**pthr** REAL Boltzmann population threshold. The [DEFAULT] is 0.05 (= 5%).
-**temp** REAL set temperature for the calculation of Boltzmann weights. The [DEFAULT] is 298.15 K.
-**nmr, -eqv** activate determination and printout of NMR-equivalencies. Writes the files anmr_rotamer and anmr_nucinfo, which are required by the ENSO python script.
-metac automatic methyl group rotamer equivalence correction.
-esort sort only based on energy (i.e., no RMSD and rotational constant comparison)
-nowr don’t write new ensemble files (crest_rotamers_‘*’.xyz, crest_conformers.xyz)
-rot use only rotational constant for checks (and no RMSD)

17.2.5 Options for other modes

-compare <FILE1> <FILE2> compare two ensembles <FILE1> and <FILE2>. Both ensembles must have the same order of atoms of the molecule and should contain rotamers.

-maxcomp INT Select the lowest INT conformers out of each ensemble to be compared with -compare. The [DEFAULT] is the 10 lowest conformers.

-iter INT Number of Protonation/Deprotonation Iterations for -tautomerize mode. The [DEFAULT] is 2 iterations.

-swel STR Change H⁺ in the protonation tool to some other ion STR, e.g. Na
Example applications

Contents

• Example applications
  – iMTD-GC conformational search
  – MF-MD-GC conformational search
  – Sorting an ensemble
  – Comparing two ensemble
  – Constrained conformational sampling
  – Sampling of noncovalent complexes and aggregates (NCI mode)
  – Molecular prototropy screening
    * Protonation site screening
    * Deprotonation site screening
    * Tautomerization screening

18.1 iMTD-GC conformational search

The default application of CREST is the iMTD-GC conformational search as described in section Introduction to CREST. In the following a standard production run with this workflow is shown for the alanineglycine molecule. The structure is given as:

```bash
> cat struc.xyz
20
```
Let’s assume that we are interested in the conformations of Ala-Gly at the GFN2-xTB level with GBSA implicit solvation for water, and that we are using 4 CPUs. Then the command to invoke the conformational search would be:

```
> crest struc.xyz -gfn2 -g h2o -T 4
```

**Tip:** It usually is wise to preoptimize your input structure with xtb at the same level on which the conformational search shall be conducted. Since the input structure is taken as a reference for several sanity checks within the CREGEN routine, such as unchanging coordination numbers of the atoms, providing a structure on the same level
of theory is recommended.

The program call first creates a coord file from the given input structure and sorts the z-matrix (ZSORT). Then the length of the MTD simulation is determined and the algorithm is started. The following output can be expected (some printout was discarded for the documentation):

```plaintext
Starting z-matrix sorting

total number of atoms : 20
total number of frags : 1
terminated normally

Generating MTD length from a flexibility measure

Calculating WBOs... done.
flexibility measure : 0.821

Starting a trial MTD to test settings

Success!
Estimated runtime for one MTD (5.0 ps) on a single thread: 16 sec
Estimated runtime for a batch of 14 MTDs on 4 threads: 1 min 4 sec

NEW ITERATION CYCLE

MTD Iteration 1

Meta-MD (MTD) Sampling

<......>
<......>

Multilevel Optimization
```

(continues on next page)
1. crude pre-optimization

writing TMPCONF* Dirs from file "crest_rotamers_0.xyz" ... done.
Starting optimization of generated structures

<.......>
353 structures remain within 12.00 kcal/mol window

2. optimization with tight thresholds

writing TMPCONF* Dirs from file "crest_rotamers_1.xyz" ... done.
Starting optimization of generated structures

<.......>
90 structures remain within 6.00 kcal/mol window

MTD Iteration 2

Collecting ensembles.
running RMSDs... done.
E lowest : -33.88024
132 structures remain within 6.00 kcal/mol window

Additional regular MDs on lowest 4 conformer(s)

Appending file crest_rotamers_1.xyz with new structures

Ensemble optimization with tight thresholds

writing TMPCONF* Dirs from file "crest_rotamers_1.xyz" ... done.
Starting optimization of generated structures

<.......>
136 structures remain within 6.00 kcal/mol window

| Structure Crossing (GC) |
input file name : crest_rotamers_3.xyz
number of atoms : 20
number of points on xyz files : 136
conformer energy window /kcal : 6.00
CN per atom difference cut-off : 0.3000
RMSD threshold : 0.2500
max. # of generated structures : 250
reading xyz file ...
# in E window 136

Chapter 18. Example applications
generating pairs ... 9315
91.2 % done
generated pairs : 7838
number of clash discarded : 1342
average rmsd w.r.t input : 2.82902
sd of ensemble : 0.63747
number of new structures : 116
removed identical structures : 384

RRR RRR
generated pairs ... 9315
91.2 % done
generated pairs : 7838
number of clash discarded : 1342
average rmsd w.r.t input : 2.82902
sd of ensemble : 0.63747
number of new structures : 116
removed identical structures : 384

RRR RRR

<table>
<thead>
<tr>
<th>Final Geometry Optimization</th>
</tr>
</thead>
</table>

Ensemble optimization

writing TMPCONF* Dirs from file "crest_rotamers_4.xyz" ... done.
Starting optimization of generated structures
126 structures remain within 6.00 kcal/mol window

CREGEN - CONFORMER SYMMETRY ANALYSIS

input file name : crest_rotamers_5.xyz
output file name : crest_rotamers_6.xyz
number of atoms : 20
number of points on xyz files : 159
RMSD threshold : 0.1250
Bconst threshold : 0.0200
population threshold : 0.0500
conformer energy window /kcal : 6.0000
# fragment in coord : 1
number of reliable points : 159
reference state Etot : -33.8802301686000
number of doubles removed by rot/RMSD : 33
total number unique points considered further : 126

<table>
<thead>
<tr>
<th>Erel/kcal</th>
<th>Etot</th>
<th>weight/tot</th>
<th>conformer set</th>
<th>degen</th>
<th>origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0.000</td>
<td>-33.88023</td>
<td>0.04725</td>
<td>0.28280</td>
<td>1 6</td>
<td>mtd10</td>
</tr>
<tr>
<td>2 0.000</td>
<td>-33.88023</td>
<td>0.04725</td>
<td>mtd1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 0.000</td>
<td>-33.88023</td>
<td>0.04724</td>
<td>mtd1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 0.001</td>
<td>-33.88023</td>
<td>0.04718</td>
<td>gc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 0.003</td>
<td>-33.88022</td>
<td>0.04698</td>
<td>mdrd2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 0.005</td>
<td>-33.88022</td>
<td>0.04689</td>
<td>gc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 0.043</td>
<td>-33.88016</td>
<td>0.04392</td>
<td>0.17556</td>
<td>2 4</td>
<td>mtd10</td>
</tr>
<tr>
<td>8 0.043</td>
<td>-33.88016</td>
<td>0.04391</td>
<td>mtd1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 0.044</td>
<td>-33.88016</td>
<td>0.04391</td>
<td>mtd9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 0.045</td>
<td>-33.88016</td>
<td>0.04383</td>
<td>mtd2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 0.477</td>
<td>-33.87947</td>
<td>0.02116</td>
<td>0.06323</td>
<td>3 3</td>
<td>mtd5</td>
</tr>
<tr>
<td>12 0.478</td>
<td>-33.87947</td>
<td>0.02112</td>
<td>mtd6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 0.482</td>
<td>-33.87946</td>
<td>0.02096</td>
<td>mtd9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CREST terminated normally.

18.1. iMTD-GC conformational search
The production run yields 126 structures of Ala-Gly, distributed over 51 different conformers within 6 kcal/mol above the lowest conformer that was found at the GFN2-xTB level.

![Three lowest conformers of alanineglycine generated by CREST at the GFN2-xTB level.](image)

The final ensemble of all the found conformers is written to an ensemble file in the Xmol format called *crest_conformers.xyz*. The corresponding CRE, i.e., the ensemble containing also the rotamers is written to the file *crest_rotamers_X.xyz*, where *X* denotes the highest number of the present files (usually *crest_rotamers_6.xyz*).

### 18.2 MF-MD-GC conformational search

To use the old MF-MD-GC algorithm (which was implementet in a small tool called confscript) the flag `-v1` can be used. In the following example we conduct this conformational search, again for alanineglycine, using GFN1-xTB and GBSA implicit solvation for CHCl₃. The command is:

```bash
> crest struc.xyz -v1 -gfn1 -g chcl3 -T 4
```

The written files are the same as with the iMTD-GC conformational search.

**Note:** The MTD-GC workflow was designed to find low lying conformers more efficiently and more safely than the older MF-MD-GC algorithm. Hence it is not recommended to use this search mode.

### 18.3 Sorting an ensemble

The *CREGEN* routine that is used within the conformational search can also be used as an standalone tool. To use this you can simply call the routine by:

```bash
> crest struc.xyz -cregen ensemble.xyz
```

Here *ensemble.xyz* is the ensemble file that contains all the structures in the Xmol format.

**Note:** It is required to present a single reference structure (*struc.xyz* in the example above) of the molecule to check for CN clashes. Also, all structures in the ensemble must have the same atom order.
18.4 Comparing two ensemble

Two ensembles generated on different levels of theory can be compared with the \texttt{-compare} option. Let's assume that there are two ensembles \texttt{v1.xyz}, generated with the MF-MD-GC procedure and \texttt{v2.xyz}, generated with the default iMTD-GC workflow. To compare the 5 lowest conformers of each ensemble simply call:

\begin{verbatim}
> crest struc.xyz -compare v1.xyz v2.xyz -maxcomp 5
\end{verbatim}

Which produces the output:

\begin{verbatim}
==============================================
| C R E S T | C R E S T |
| Conformer-Rotamer Ensemble Sampling Tool | based on the GFN-xTB method |
| S.Grimme, P.Pracht | Universitaet Bonn, MCTC |
|-----------------------------------------|
Using the GFN-xTB code.
Compatible with XTB version 6.1 and later.

-----------------------------------------------
Sorting file \texttt{v1.xyz}
-----------------------------------------------
running RMSDs... done.
File \texttt{v1.xyz} contains 240 conformers.
The 5 lowest conformers will be taken for the comparison:
conformer #rotamers
  1 1
  2 5
  3 3
  4 1
  5 2

-----------------------------------------------
Sorting file \texttt{v2.xyz}
-----------------------------------------------
running RMSDs... done.
File \texttt{v2.xyz} contains 51 conformers.
The 5 lowest conformers will be taken for the comparison:
conformer #rotamers
  1 6
  2 4
  3 3
  4 6
  5 4

-----------------------------------------------
Comparing the Ensembles
-----------------------------------------------
Calculating RMSDs between conformers... done.
RMSD threshold: 0.1250 Å

RMSD matrix:
conformer: 1 2 3 4 5
\end{verbatim}
From the output it can be seen that there is a correlation between the lowest conformers, i.e., the lowest conformers were found by both workflows. As the display options in the terminal are limited, an additional file called `rmsdmatch.dat` is written, from which the exact correlation between the conformers of the two ensembles can be read. If, for example, two different levels of theory are used and the energies of the molecules in both ensembles are too different, then the output will not be of much use and one must refer to the `rmsdmatch.dat` file.

```
> cat rmsdmatch.dat
1 1
2 1
3 2
```

Each line in this file consists of only two values `a` and `b` which denote that conformer `a` from ensemble `A` matches conformer `b` from ensemble `B`. In the example case shown above, the MF-MD-GC produced the lowest conformer twice, which both naturally match conformer 1 from the iMTD-GC procedure. The second conformer also is the same in both ensembles.

**Note:** In order for the comparison to work, both ensembles must have the same number of atoms with the same atom order in each structure. Furthermore the ensembles should be full CREs, i.e., rotamers should be present.

### 18.5 Constrained conformational sampling

**Warning:** The following application is still under development and should be considered an experimental feature.
It is possible to include additional constraints to all xtb calculations that are conducted by CREST. To do this one has to create a file called .constrains (or .xcontrol, both is valid) in the working directory, which contains the constraints in the exact same syntax as used by the xtb (see section Detailed Input) Constraints that are included via the .constrains file will be included in ALL calculations of the conformer search run. Since this can overwrite settings created by CREST it should only be used very cautiously!

The main application for the additional constraints is the constrainment (fixing) of atoms, which could for example be used to sample only conformations for parts of a molecule. Another use could be the sampling of conformers for the transition state of an reaction.

To fix atoms it is also recommended to use an reference input file additionally to the normal structure input file, which is done with the argument reference=FILE in the .xcontrol file. Furthermore, fixed atoms should not be included in the RMSD of the MTD collective variables.

The content of the .xcontrol file for fixing atoms should look like the following example:

```
> cat .xcontrol
$constraint
(atoms: 4,8,10,12) # atoms 4, 8, 10 and 12 of some example molecule shall be constrained
(force constant=0.5)
(reference=coord.original) # name of the reference file (just a copy of the input coord-file)
$metadyn
(atoms: 1-3,5-7,9,11) # atoms *included* to RMSD in the MTD (typically NOT the constrained atoms)
$end
```

This should ensure correct constrainment (as far as possible) in the MTD, as well as in the GFNn-xtB geometry optimization within a CREST run.

### 18.6 Sampling of noncovalent complexes and aggregates (NCI mode)

A specialized application of CREST is the sampling of aggregates (also refered to as NCI mode). The idea here is to find different conformations of non-covalently bound complexes in which the arrangement of the fragments is of interest. The application can be called by:

```
> crest struc.xyz -nci
```

The procedure and output is essentially the same as a normal iMTD-GC production run, but with reduced settings (less MTDs, different $k$ and $\alpha$), and no genetic structure crossing. What is different, however, is that first a ellipsoide wall potential is created and added to the meta-dynamics. A nice example for this application are small molecular clusters, e.g. (H$_2$O)$_6$. The ellipsoide potential that is automatically determined for the input cluster is visualized in the figure below.

The ellipsoide potential is required in the MTDs to counteract the bias potential, which would simply lead to a dissociation of the NCI complex after a few pico seconds (due to the maximization of the RMSD). In the subsequent geometry optimization, however, the surrounding potential must not be present since the bias potential is also not there and the structure would be artificially compressed by the ellipsoide. Hence it is automatically removed in the geometry optimizations.

**Note:** The ellipsoide potential can be scaled by the factor $REAL$ with the flag `-wscal REAL`.

Many new clusters are generated even for small NCI complexes, typically much more than conformers are generated for a single medium sized molecule. In general, the task of finding new low lying aggregates is much more challeng-
ing than finding (only) conformers, since each fragment of the complex could also have several different low lying conformations. For the (H₂O)₆ cluster 3 examples are shown in the figure below. Note that all three structures are also part of the well established WATER27 benchmark set, but were generated automatically by CREST from a single input structure. In total 69 different clusters were found of which only 3 are shown.

Fig. 3: Visualization of an ellipsoid potential around (H₂O)₆ cluster.

Fig. 4: Three automatically generated structures for a (H₂O)₆ cluster.
18.7 Molecular prototropy screening

18.7.1 Protonation site screening

The screening for possible protonation sites, i.e., for the different protomers of an molecule is possible by using a localized molecular orbital LMO approach. Herein, first the $\pi$- and LP-centers are determined by a GFNn-xTB calculation, and then all possible input structures are generated where a proton is placed at one of these centers. This procedure was first described in *J. Comput. Chem.*, 2017, 38, 2618–2631.

The example calculation is performed for alanineglycine, in the gas phase, with the command

```bash
> crest struc.xyz -protonate
```

Which returns the following output:

```
LMO calculation ... done.

Multilevel Optimization

1. crude pre-optimization

writing TMPCONF+ Dirs from file "protonate_0.xyz" ... done.
Starting optimization of generated structures
<.......>
Now appending opt.xyz file with new structures
12 structures remain within 90.00 kcal/mol window

2. loose optimization

writing TMPCONF+ Dirs from file "protonate_1.xyz" ... done.
Starting optimization of generated structures
<.......>
Now appending opt.xyz file with new structures
11 structures remain within 60.00 kcal/mol window
```

(continues on next page)
3. optimization with user-defined thresholds

writing TMPCONF* Dirs from file "protonate_2.xyz" ... done.
Starting optimization of generated structures
<.......>
Now appending opt.xyz file with new structures
9 structures remain within 30.00 kcal/mol window

Identifying topologically equivalent structures:
Equivalent to 1. structure: 2 structure(s).
Equivalent to 3. structure: 5 structure(s).
Equivalent to 5. structure: 2 structure(s).
Done.
Appending file <protonated.xyz> with structures.

Initial 9 structures from file protonate_3.xyz have been reduced to 3 topologically unique structures.

--- ordered structure list ---
written to file <protonated.xyz>

structure   ΔE (kcal/mol)  Etot (Eh)
1          0.00  -33.964453
2          3.51  -33.958853
3          5.75  -33.955296

Wall Time Summary

LMO calc. wall time : 0h : 0m : 0s
multilevel OPT wall time : 0h : 0m : 3s
Overall wall time : 0h : 0m : 4s

CREST terminated normally.

As one can see from the output, three possible protomers of alanineglycine were found at the GFN2-xTB level (within the default 30 kcal/mol energy window around the most stable protomer). This ensemble of structures is written to a file called protoners.xyz. The first (lowest) protomer created by CREST for this molecule includes a ring-closure, apparently caused by the addition of the proton. This nicely demonstrates the ability of our approach to form and break new bonds. The three protomers are shown in the figure below.

18.7.2 Deprotonation site screening

The general approach to find deprotonation sites at a GFN level is much more simple than finding protonation sites. For each hydrogen atom in the structure a new (deprotonated) reference structure is created and optimized in a multilevel approach. The commandline argument to invoke this search is:

```bash
> crest struc.xyz -deprotonate
```
Fig. 5: Three lowest protomers of alanineglycine generated by CREST at the GFN2-xTB level.

For the example of alanineglycine, again three structures are obtained and written to a file called `deprotonated.xyz`:

```
<.......>
<.......>
```

```
=================================== ordered structure list =================================
written to file <deprotonated.xyz>
structure  ΔE (kcal/mol)  Etot (Eh)
1          0.00          -33.593702
2          21.83         -33.558913
3          25.12         -33.553669
```

However, two of the three structures have much higher energies and therefore mainly the lowest deprotomer should be considered.

Fig. 6: Lowest deprotomer of alanineglycine at the GFN2-xTB level. The deprotonation happens at the carboxyl group.
The last application of the different prototropy screening protocols is an automatized tautomerization tool, which utilizes both the protonation and deprotonation procedures presented in the previous two subsections. By first protonating a molecule and then deprotonation of the resulting protomers at all positions, prototropic tautomers relative to the initial input structure can be found. A single cycle of this protonation/deprotonation in principle yields all tautomers with a single hydrogen permutation relative to the input. If a higher number of hydrogen permutations is required, the procedure can simply be repeated with the created tautomers, i.e., tautomers with two or more hydrogen atom permutations are generated. From experience, however, it is generally sufficient to repeat this protonation/deprotonation cycle twice (which is the default in CREST), in order to get the relevant low energy tautomers. The approach was first described in *J. Comput.-Aided Mol. Des.*, 2018, 32, 1139-1149. The tautomerization search can be conducted by the command

```
> crest struc.xyz -tautomerize
```

**Tip:** The number of protonation/deprotonation cycles can be adjusted with the flag `-iter INT`, where `INT` is the number of cycles.

For alanineglycine the following output is generated:

```
======================================================================================================
|                                                                                                         |
| C R E S T                                                                                               |
| Conformer-Rotamer Ensemble Sampling Tool                                                                |
| based on the GFN-xTB method                                                                            |
| S.Grimme, P.Pracht                                                                                      |
| Universitaet Bonn, MCTC                                                                                |
======================================================================================================
Version 2.7.0, Mon 24. Jun 11:41:02 CEST 2019
Using the GFN-xTB code.
Compatible with XTB version 6.1 and later.

automated tautomeration script

*******************************************************************************************
** PROTONATION CYCLE 1 of 2 **

LMO calculation ... done.
-----------------------------
Multilevel Optimization
-----------------------------
<........>
Identifying topologically equivalent structures:
<.......>
Appending file <protonated.xyz> with structures.
Initial 9 structures from file protonate_2.xyz have been reduced to 3 topologically unique structures.

(continues on next page)
**DEPROTONATION CYCLE 1 of 2**

```
<table>
<thead>
<tr>
<th>Structure</th>
<th>ΔE(kcal/mol)</th>
<th>Etot(Eh)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>-33.964400</td>
</tr>
<tr>
<td>2</td>
<td>3.60</td>
<td>-33.958659</td>
</tr>
<tr>
<td>3</td>
<td>5.78</td>
<td>-33.955188</td>
</tr>
</tbody>
</table>
```

**PROTONATION CYCLE 2 of 2**

```
<table>
<thead>
<tr>
<th>Structure</th>
<th>ΔE(kcal/mol)</th>
<th>Etot(Eh)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Identification of topologically equivalent structures:**

---

**Multilevel Optimization**

---

**Collecting generated protomers ... done.**

---

**Molecular prototropy screening**
Deprotonation Cycle 2 of 2

-----------------------
Multilevel Optimization
-----------------------

Identifying topologically equivalent structures:

Appending file <deprotonated.xyz> with structures.

Initial 95 structures from file deprotonate_2.xyz have been reduced to 19 topologically unique structures.

Structure list written to file <deprotonated.xyz>

structure ΔE(kcal/mol) Etot(Eh)

1 0.00 -33.867777
2 1.99 -33.864606
3 3.84 -33.861657
4 3.84 -33.861656
5 4.42 -33.860731
6 4.68 -33.860314
7 10.63 -33.850839
8 10.79 -33.850575
9 10.92 -33.850381
10 10.95 -33.850329
11 12.18 -33.848371

(continues on next page)
As can be seen from the output, the entire procedure is constructed from the protonation and deprotonation site screening routines. The first protonation step yields the same three protomers that are also obtained by the standalone application, which are then automatically deprotonated. Two protonation/deprotonation cycles are performed. The final tautomer ensemble consists of 19 structures (within 30 kcal/mol) and is written to the file tautomers.xyz.
CHAPTER 19

Short API Introduction

The xtb program is not designed to be interfaced by an external program, while one can write a wrapper script to start calculations and parse the output this is in most cases cumbersome and error prone.

Interfacing xtb can be done using the application programmable interface (API) shipped with the shared library version of xtb. This section targets mainly developers trying to interface their programs with xtb.

Warning: The current state of the C-API is still somewhat experimental, this means the API definitions might change in the future without prior announcement, but this also means suggestions and feature request regarding the API are welcome and likely to be included.

Contents

- Short API Introduction
  - General Comments
  - Available Interfaces

19.1 General Comments

Remember that xtb is written in Fortran, since modern Fortran usually makes massive uses of name globbing and the automatically generated headers (module files) tend to be compiler specific and incompatible, the only two viable options are to write the interface without using modules or to use ISO-C compatible bindings to create the API. It turns out that we did both, we have an interface layer using module-free Fortran subroutines which setups all datatypes and does the actual calculation and an ISO-C compatible layer, which will eventually call this Fortran layer.

Since we are not planning to let you directly interface with our Fortran layer for now, we will focus on the ISO-C compatible layer, the C-API, here. The header definitions are found in include/xtb.h, for easy reference parts of it are included here.
As you might noticed, we prefer to get your data by reference, also there are different interfaces depending on what you are planning to do. For clarification we give you an interface definition for the actual Fortran function as well

```fortran
function gfn2_api &
  (natoms, attyp, charge, coord, opt_in, file_in, &
   etot, grad, dipole, q, dipm, qp, wbo) &
  result (status) bind(C, name="GFN2_calculation")

use iso_c_binding

implicit none

type, bind(C) :: c_scc_options
  integer(c_int) :: prlevel = 0_c_int
  integer(c_int) :: parallel = 0_c_int
```

(continues on next page)
The C-API in general needs the information on the number of atoms \( \text{natoms} \), a \( \text{natoms} \) wide array of integers \( \text{attyp} \) with the ordinal numbers of the atoms, the total charge of system \( \text{charge} \) and the cartesian coordinates in a continuous \( 3 \times \text{natoms} \) wide array of doubles \( \text{coord} \), with the coordinate triples next to each other. Additionally we require you to give us a struct \( \text{opt} \) containing some more specific information on the calculation and thresholds employed and a location \( \text{output} \) to write your output to. Since C file pointer and Fortran units might not be as compatible as we would wish, we decided to pass this information around as string ("-" can be used for STDOUT).

The function will return its status so you can check if the calculation done by the shared library was successful or not, note that Fortran can be quite drastic when using features like \texttt{error stop}, which is likely to kill the caller program too, without giving you even the chance to react or catch it. We promise to not use it when you are calling our API.

The calculated values are written to some location you have to reserve before calling the shared library, so make sure that you have enough memory reserved.

We will generally refrain from using any of the memory you reserved on the caller side, except for copying the results from our arrays to yours. This sounds actually quite wasteful on resources, it is not that we are not trusting your memory management, but we prefer to do the memory management on your side with proper Fortran.

### 19.2 Available Interfaces

Currently we have interfaces available for the three Hamiltonians (GFN2-xTB, GFN1-xTB and GFN0-xTB) which come in different flavours depending on the things you attempt to calculate.

#### molecular GFN2-xTB calculation

```c
extern int
GFN2_calculation(const int* natoms, const int* attyp, const double* charge,
                 const double* coord, const SCC_options* opt, const char* output,
```
double* energy, double* grad, double* dipole, double* q,
double* dipm, double* qp, double* wbo);

molecular GFN1-xTB calculation

extern int
GFN1_calculation(const int* natoms, const int* attyp, const double* charge,
const double* coord, const SCC_options* opt, const char* output,
double* energy, double* grad);

molecular GFN0-xTB calculation

extern int
GFN0_calculation(const int* natoms, const int* attyp, const double* charge,
const double* coord, const PEEQ_options* opt, const char* output,
double* energy, double* grad);

periodic GFN0-xTB calculation

extern int
GFN0_PBC_calculation(const int* natoms, const int* attyp, const double* charge,
const double* coord, const double* lattice, const bool* pbc,
const PEEQ_options* opt, const char* output,
double* energy, double* grad, double* glat);
In this section the application programmable interface (API) of the xtb program package is described. This section targets mainly developers trying to interface their (Python) scripts with xtb. The necessary files are included from version 6.2 RC2 in the distributed tarball.

20.1 Setting up ASE

First of all, get a version of the atomic simulation environment (ASE), usually

```bash
$ pip3 install ase [--user]
```

works fine on most machines. For more details refer to the ASE documentation.

20.2 Loading the Shared Library

Note: This is the basic approach to include an interface to a C-API in Python, in most circumstances you can skip this section since I already wrapped up everything nicely. If you plan to modify the C-API and the Python wrappers, this section is important for everything you do.
The xtb program package contains a shared object which has to be included in your LD_LIBRARY_PATH, you can simply do this by using

```
> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/xtb/build
```

to allow loading of the shared library.

**Warning:** (Ab)using your LD_LIBRARY_PATH this way is generally not recommended, unless I have figured out how to do it correctly in Python this might be your best choice.

Test this by running:

```python
import ctypes
from ctypes import cdll

try:
    xtb = cdll.LoadLibrary('libxtb.so')
except OSError:
    print("xtb library was not found in your LD_LIBRARY_PATH")
```

If you can successfully load the shared object, specify the necessary interface for calling xtb by defining the PEEQ_options structure and argument types as in:

```python
import ctypes
from ctypes import cdll, Structure, c_int, c_double, c_bool, c_char_p, POINTER

xtb = cdll.LoadLibrary('libxtb.so')

class PEEQ_options(Structure):
    _fields_ = [('prlevel', c_int),
                ('parallel', c_int),
                ('acc', c_double),
                ('etemp', c_double),
                ('grad', c_bool),
                ('ccm', c_bool)]

C_int_p = POINTER(c_int)
C_bool_p = POINTER(c_bool)
C_double_p = POINTER(c_double)
peeq.GFN0_PBC_calculation.argtypes = [C_int_p, C_int_p,
                                       C_double_p, C_double_p,
                                       C_double_p, C_bool_p,
                                       C_char_p, C_double_p,
                                       C_double_p, C_double_p]
```

now Python knows how to call xtb from the shared object. Remember that xtb is a Fortran program, so we prefer passing by reference over passing by value.

**Tip:** You can always check the header definitions in include/xtb.h.

### 20.3 Using as ASE Calculator

To perform a calculation with the ASE we not only need Python bindings but also an abstract interface to other ASE functions. The easiest way to provide such an interface is by creating an ASE Calculator class. My current
approach is to have an abstract class performing all the nasty interfacing stuff (loading the library, storing default values and stuff like that) and specific instances of this class for every available method from xtb, namely GFN2-xTB (as GFN2), GFN1-xTB (as GFN1) and GFN0-xTB (as GFN0 and GFN0_PBC for molecular and periodic calculations, respectively). An complete implementation of this setup is shipped with xtb at python/xtb.py and should be ready-to-use with some minor tweaking. To make it available for scripting in Python use

> export PYTHONPATH=$PYTHONPATH:/path/to/xtb/python

Here is an example with rutile using this VASP geometry input:

<table>
<thead>
<tr>
<th>Ti</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000000000000000</td>
<td>0.0000000000000000</td>
</tr>
<tr>
<td>4.6257</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0000</td>
<td>4.6257</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

To give you an idea how this is going to work out, here is the final code snippet:

```python
import xtb
from xtb import GFN0_PBC
import ase
from ase.io import read, write
from ase.units import Hartree
from ase.optimize.precon import Exp, PreconFIRE
from ase.constraints import ExpCellFilter

# read molecular structure data, here from a VASP geometry input
mol = read("POSCAR", format = 'vasp')

# create the calculator for GFN0-xTB under periodic boundary conditions
calc = GFN0_PBC(print_level = 3)
mol.set_calculator(calc)

# initial single point calculation
e = mol.get_potential_energy()
print("Initial energy: eV, Eh", e, e/Hartree)

# setup optimization of cell parameters
ecf = ExpCellFilter(mol)
precon = Exp(A = 3)
relax = PreconFIRE(ecf, precon = precon, trajectory = 'xtbopt.traj')

# do the optimization
relax.run(fmax = 5e-2)

# get the final single point energy
e = mol.get_potential_energy()
print("Final energy: eV, Eh", e, e/Hartree)
```

(continues on next page)
# write final geometry to file
write("xtbopt.POSCAR", mol, format = 'vasp')

running this script with the input for rutile we should find something similar to this output (maybe including some
warnings from the ASE).

<table>
<thead>
<tr>
<th>PreconFIRE: 0 09:28:06</th>
<th>-440.647107</th>
<th>1.7119</th>
<th>0.1061</th>
</tr>
</thead>
<tbody>
<tr>
<td>PreconFIRE: 1 09:28:07</td>
<td>-440.673281</td>
<td>1.7110</td>
<td>0.1056</td>
</tr>
<tr>
<td>PreconFIRE: 2 09:28:07</td>
<td>-440.725466</td>
<td>1.7076</td>
<td>0.1045</td>
</tr>
<tr>
<td>PreconFIRE: 3 09:28:07</td>
<td>-440.803152</td>
<td>1.6977</td>
<td>0.1026</td>
</tr>
<tr>
<td>PreconFIRE: 4 09:28:07</td>
<td>-440.905138</td>
<td>1.6747</td>
<td>0.0993</td>
</tr>
<tr>
<td>PreconFIRE: 5 09:28:07</td>
<td>-441.028875</td>
<td>1.6284</td>
<td>0.0941</td>
</tr>
<tr>
<td>PreconFIRE: 6 09:28:08</td>
<td>-441.169498</td>
<td>1.5430</td>
<td>0.0860</td>
</tr>
<tr>
<td>PreconFIRE: 7 09:28:08</td>
<td>-441.318524</td>
<td>1.3969</td>
<td>0.0738</td>
</tr>
<tr>
<td>PreconFIRE: 8 09:28:08</td>
<td>-441.462322</td>
<td>1.1298</td>
<td>0.0539</td>
</tr>
<tr>
<td>PreconFIRE: 9 09:28:08</td>
<td>-441.600489</td>
<td>0.6531</td>
<td>0.0220</td>
</tr>
<tr>
<td>PreconFIRE: 10 09:28:08</td>
<td>-441.654277</td>
<td>0.1566</td>
<td>0.0277</td>
</tr>
<tr>
<td>PreconFIRE: 11 09:28:09</td>
<td>-441.515093</td>
<td>0.1524</td>
<td>0.0275</td>
</tr>
<tr>
<td>PreconFIRE: 12 09:28:09</td>
<td>-441.652546</td>
<td>0.1441</td>
<td>0.0270</td>
</tr>
<tr>
<td>PreconFIRE: 13 09:28:09</td>
<td>-441.653083</td>
<td>0.1319</td>
<td>0.0264</td>
</tr>
<tr>
<td>PreconFIRE: 14 09:28:09</td>
<td>-441.653747</td>
<td>0.1161</td>
<td>0.0256</td>
</tr>
<tr>
<td>PreconFIRE: 15 09:28:09</td>
<td>-441.654502</td>
<td>0.0972</td>
<td>0.0247</td>
</tr>
<tr>
<td>PreconFIRE: 16 09:28:10</td>
<td>-441.655309</td>
<td>0.0756</td>
<td>0.0236</td>
</tr>
<tr>
<td>PreconFIRE: 17 09:28:10</td>
<td>-441.656129</td>
<td>0.0519</td>
<td>0.0225</td>
</tr>
<tr>
<td>PreconFIRE: 18 09:28:10</td>
<td>-441.656933</td>
<td>0.0242</td>
<td>0.0212</td>
</tr>
</tbody>
</table>

The final geometry can be found in *xtbopt.POSCAR* and can be viewed with e.g.

```bash
> ase gui xtbopt.POSCAR
```

The optimization log is kept in a pickle trajectory and can also be viewed with the *ase gui*.
Note: Release candidates and beta versions are not listed here.

Version 6.2
- Bugfix: Fukui index calculation
- Bugfix: wrong forces in FIRE optimizer
- Bugfix: Scube instructions were not read
- Bugfix: Input error for $gbsa data group
- GFN0-xTB Hamiltonian consistent with ChemRxiv preprint
- periodic boundary conditions for GFN0-xTB
- preliminary implicit solvation model GBSA for GFN0-xTB

Version 6.1.4
- Bugfix: parallisation error in GBSA

Version 6.1.3
- added FIRE and L-ANCopt as optimization engines
- Bugfix: FOD calculation was using wrong density

Version 6.1.2
- Bugfix: wrong convergence threshold for RF solver

Version 6.1.1
- Bugfix: wrong constraining energy in xtbscan.log
- Bugfix: symmetry finder was inactive

Version 6.1
• removed isotope input
• Turbomole basis and mos printout
• ORCA GBW file printout
• metadynamics runtyp added
• GFN0-xTB implemented
• completely tunable model Hessian for optimizer
• separated fixing and constraining
• elementwise fixing and constraining
• new geometry summary printout for optimizations
• better printout for optimizer (RMSD, energy gain)
• profiling printout for SCC and optimizer
• adjustable SASA grid for GBSA
• case insensitive solvent strings for GBSA
• Bugfix: mode following printout crashes for large systems (>100 atoms)

Version 6.0.2
• Bugfix: timings wrapped around

Version 6.0.1
• additional GFN2-xTB GBSA parameters added
• Bugfix: molden.input could not be disabled
• Bugfix: deallocation error in mode following

Version 6.0
• GFN2-xTB GBSA parameters added
• internal parameter files
• detailed input
• XTBPATH variable
• parallel Hessian with GBSA
• logfermi wallpotential added
• sdf input files supported
• automatic Fukui indices and electrophilicity index
CHAPTER 22

xTB related Publications

Contents

• xTB related Publications
  – Methods
  – Applications

22.1 Methods


22.2 Applications


ECD spectra calculation for entire proteins: Seibert, J.; Bannwarth, C.; Grimme, S. Biomolecular structure information from high-speed quantum mechanical electronic spectra calculation. J. Am. Chem. Soc. 2017, jacs.7b05833 DOI: 10.1021/jacs.inorgchem.7b01950


This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0). To view a copy of this license, visit creative commons or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.
CHAPTER 24

Need Help

If you’re heading trouble please contact xtb@thch.uni-bonn.de