ForceField Manual

Amsterdam Modeling Suite 2024.1

www.scm.com

Apr 05, 2024

CONTENTS

1 Introduction **1**
- 1.1 Available force fields . . . 1
- 1.2 ForceField-GUI . . . 1
- 1.3 What's new in ForceField 2024 . . . 1
- 1.4 What's new in ForceField 2023 . . . 1
- 1.5 What's new in ForceField 2022 . . . 2
- 1.6 What's new in ForceField 2021 . . . 2
- 1.7 What's new in ForceField 2020 . . . 2
  - 1.7.1 New features . . . 2
- 1.8 Force field setup for AMS applications . . . 3
- 1.9 Classical force fields . . . 3

2 Atom typing behavior **5**
- 2.1 UFF . . . 5
- 2.2 Other forcefields . . . 5

3 Input via the AMS system block **7**
- 3.1 ForceField input example . . . 7
- 3.2 Antechamber integration . . . 9

4 ForceField Engine Options **11**
- 4.1 Common options . . . 11
  - 4.1.1 Type . . . 11
  - 4.1.2 Non-bonded screening . . . 11
  - 4.1.3 Feedback . . . 12
  - 4.1.4 Bonds usage . . . 12
  - 4.1.5 Ewald summation . . . 13
  - 4.1.6 Disabling energy terms . . . 14
  - 4.1.7 Guessing or loading partial charges . . . 15
    - GuessCharges . . . 15
    - LoadCharges . . . 16
- 4.2 Amber force field options . . . 16
- 4.3 UFF options . . . 17
- 4.4 APPLE&P force field options . . . 18
- 4.5 Offloading calculations to LAMMPS . . . 19
  - 4.5.1 Setting up LAMMPS . . . 19
  - 4.5.2 Input options . . . 20

5 APPLE&P **23**
- 5.1 APPLE&P use cases . . . 23

5.1.1 Organic Molecules . . . 23
5.1.2 Energetic Materials . . . 23
5.1.3 Polymers . . . 24
5.1.4 Solvents/Additives . . . 24
5.1.5 SEI compounds . . . 25
5.1.6 Cations . . . 25
5.1.7 Anions . . . 25
5.1.8 Electrodes . . . 26
5.2 APPLE&P potential shapes . . . 26
5.3 APPLE&P Python functions . . . 27

6 Examples **29**
6.1 Example: single point for aspirin with UFF . . . 29
6.2 Example: partial charges and total charge . . . 30
6.3 Example: load partial charges from another engine . . . 40
6.4 Example: load atom types from a previous run . . . 42
6.5 Example: using amber or tripos . . . 45
6.6 Example: calculate some simple properties for water . . . 46
6.7 Example: single point for ammonia with TRIPOS . . . 47

7 Required citations **49**
7.1 External programs and libraries . . . 49

8 Parameter files **51**
8.1 UFF parameters . . . 51
8.1.1 User-modified force fields (expert option) . . . 51
8.2 AMBER parameters . . . 54
8.2.1 AMBER forcefield file . . . 55
8.2.2 SCM forcefield file . . . 55
8.3 APPLE&P parameters . . . 63

9 Potential shapes **67**

10 Periodic ForceField **69**

11 References **71**

12 Keywords **73**
12.1 Links to manual entries . . . 73
12.2 Summary of all keywords . . . 73
12.2.1 Engine ForceField . . . 73

13 KF output files **83**
13.1 Accessing KF files . . . 83
13.2 Sections and variables on forcefield.rkf . . . 84

Index **109**

CHAPTER

ONE

INTRODUCTION

As per 2020 UFF force fields and simple classical force fields are unified in the ForceField engine. This replaces the old UFF engine and the NewMM functionality from ADF.

1.1 Available force fields

The default force field is UFF, it requires no configuration. UFF (Universal Force Field) is a full periodic table force field that can be used to calculate single point energies, do geometry optimizations, calculate frequencies, etc. It is also the default pre-optimizer in the Graphical User interface (GUI) to clean up manually drawn structures. For details on the inner workings of the force field, see the 1992 paper by Rappe et al. [*1* (page 71)]. By default UFF uses non-zero charges for water molecules only.

Other available force fields managed by this engine include Amber95, Tripos 5.2, GAFF and APPLE&P [*9* (page 71)].

1.2 ForceField-GUI

Note that the graphical user interface UFF-GUI enables all users to set up force field calculations with a few mouse clicks. Most force fields require atom typing, and this needs to be done by hand. When using UFF, also for pre-optimizing, it is important to remember that any force field method relies on a set of parameters. This means that even though UFF supports all elements up to Z=103, it might not generate the desired structures for uncommon oxidation states in metallic structures. If this is the case, you could add new parameters to UFF or attach dummy hydrogen atoms to the metal atom.

1.3 What's new in ForceField 2024

- *APPLE&P* (page 18) parameters updated to v 1.13. See also the APPLE&P tutorial on ionic liquids.

1.4 What's new in ForceField 2023

- New *APPLE&P* (page 18) polarizable force field [*9* (page 71)] parameters. See also the APPLE&P tutorial on ionic liquids.

1.5 What's new in ForceField 2022

- New *APPLE&P* (page 18) polarizable force field [*9* (page 71)], designed for polymers, electrolytes, ionic liquids and energetic materials. See also the APPLE&P tutorial on ionic liquids.
- The GFN-FF force field by Spicher and Grimme. **Note:** GFNFF is technically not part of the ForceField engine. To use GFNFF, you should use the **GFNFF engine** (and not the ForceField engine). See the GFNFF engine manual for more info.

1.6 What's new in ForceField 2021

- The engine has been parallelized with MPI using force decomposition.
- The particle mesh Ewald method (implemented through the helPME library (https://github.com/andysim/helpme)) is now used for electrostatic interactions in 3D systems.
- The default non-bonded cutoff has been reduced to 15 Angstrom, a value more typical in the force-field community.
- The engine has been optimized to be on par with the industry standards. The total speed-up compared to version 2020 can be up to factor 500.

1.7 What's new in ForceField 2020

1.7.1 New features

- Periodic support (chains,slabs, and crystals) for all force field types
- Import force field parameters from amber ".dat" files

shell script without atom typing:

```
#!/bin/sh

# This is a shell script for the ForceField engine

# You should use '$AMSBIN/ams' instead.

$AMSBIN/ams <<eor
   # Input options for the AMS driver:

   Task GeometryOptimization

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

   # The input options for the ForceField, which are described in this manual,
   # should be specified in the 'Engine ForceField' block:

   Engine ForceField
      # the default one is UFF, requiring no options
```

(continues on next page)

(continued from previous page)

```
   EndEngine
eor
```

1.8 Force field setup for AMS applications

When you do a geometry optimization of perform a MD calculation the details of the force field are determined only once and kept constant during the run. This applies to bond orders, partial charges, etc. In case of a polarizable force field (not currently supported) the charges are of course allowed to change.

1.9 Classical force fields

Most force fields require atom typing, as well as atomic charge specifications, see *Examples* (page 29).

CHAPTER

TWO

ATOM TYPING BEHAVIOR

To use a classical forcefield each atom should be given a type, and a partial charge. The power of the UFF forcefield is that this atomtyping is done automatically. For other force fields the atomtyping for most systems still has to be done by hand, possibly with the help of an external program. Unlike UFF parameters, the availabe parameters for the other available force fields typically cover only a fraction of all chemical elements. Nonetheless, experimental atomtyping options are available for small organic molecules (GAFF), and for biological systems (AMBER).

2.1 UFF

Atom typing is the process of matching MM atom types to elements. For example, a Carbon atom becomes a C_1, C_2, C_3 or C_R, depending on the number (and type) of bonds it has to neighboring atoms. UFF is capable of finding a matching MM atom type on its own, but might not always succeed in doing so. When doing calculations with UFF, it is important to check the beginning of the output file, as the program will print the detected MM atom types there. You can also take matters into your own hands, and tell UFF what MM atom types you want it to use (see the section on Input and examples).

The atom typing in UFF is mostly controlled by the elements and mmatomtypes parameter files, however, some part of it is hidden in the code itself and is (at the moment) not accessible to users. This is done by UFF to differentiate between:

- Carbon: double-bonded sp2 carbon (C_2) vs aromatic carbon (C_R), if any of the orders are close to 2, it's C_2.
- Oxygen: having a bond to silicon gives O_3_z; otherwise, having bond order of 1.44 or higher gives O_R, provided that the partner is not a hydrogen. Otherwise, it's just an O_3.
- Sulfur: having a bond order greater than 1.3 gives aromatic S_R.

By default, zero charges are used, unless a water molecule is detected.

2.2 Other forcefields

Automatic atomtyping via an experimental integration of the Antechamber toolkit is available for the Generalized AMBER Force Field (GAFF) for small organic molecules. This option was introduced in the 2020 release of AMS, and is still considered experimental and disabled by default.

See also:

Antechamber integration for GAFF forcefields (page 9)

Some atomtyping functionality for biological systems can be accessed via the GUI, using pdb files as input. A pdb file can be loaded into amsinput and then under the regions the residues can be found. Still, charges need to be set by hand.

Custom force field parameters can be provided as forcefield files. The two supported formats are ".ff" (originating from the ADF and QUILD programs) and the much more widely used amber ".dat" files.

CHAPTER

THREE

INPUT VIA THE AMS SYSTEM BLOCK

While usually options for an AMS engine are defined in its engine block, for the ForceField engine three ingredients are defined via the AMS system block: bonds, atom types and (partial) charges.

Here are some logical options

- 1) Specify Everything: elements, coordinates, MM atom types, charges, and bonds
- 2) Specify elements, coordinates, and bonds (UFF only)
- 3) Specify elements and coordinates (UFF only, most convenient)

Currently, for most force fields, everything needs to be specified (option 1). UFF and GAFF allow the automatic determination of bonds and atom types (option 2 and 3). For GAFF this option is considered experimental as of the 2020 release and is disabled by default, see *Antechamber integration* (page 9) below. See also the *BondsUsage* (page 12) key on how bond information can be tweaked.

3.1 ForceField input example

1) Specify Elements, coordinates, MM Atom Types, Charges, and bonds:

```
$AMSBIN/ams << eor

Task GeometryOptimization

System
    Atoms
        C    1.36012328  -0.14520095   0.60144543    ForceField.Type=C_3     ␣
↪ForceField.Charge=0.000000
        C    0.00000000   0.00000000   0.00000000    ForceField.Type=C_2     ␣
↪ForceField.Charge=0.000000
        H    2.09833847  -0.46327872  -0.16560721    ForceField.Type=H_      ␣
↪ForceField.Charge=0.000000
        H    1.32657807  -0.90546800   1.40917410    ForceField.Type=H_      ␣
↪ForceField.Charge=0.000000
        H    1.67935140   0.82750664   1.02977296    ForceField.Type=H_      ␣
↪ForceField.Charge=0.000000
        H   -0.83486863   0.30434056   0.62258487    ForceField.Type=H_      ␣
↪ForceField.Charge=0.000000
        O   -0.18030374  -0.22462371  -1.18585739    ForceField.Type=O_2     ␣
↪ForceField.Charge=0.000000
    End
    BondOrders
         1 5 1.0
```

(continues on next page)

(continued from previous page)

```
         1 4 1.0
         1 3 1.0
         1 2 1.0
         2 6 1.0
         2 7 2.0
    End
End

Engine ForceField
EndEngine

eor
```

The format in the bonds section is: atom A, atom B, bond order.

2) Specify Elements, coordinates, and bonds (UFF only):

If we leave out the MM atom types and charges, UFF will determine the MM atom types automatically from the bond information:

```
$AMSBIN/ams << eor

Task GeometryOptimization

System
    Atoms
        C  1.36012328 -0.14520095  0.60144543
        C  0.00000000  0.00000000  0.00000000
        H  2.09833847 -0.46327872 -0.16560721
        H  1.32657807 -0.90546800  1.40917410
        H  1.67935140  0.82750664  1.02977296
        H -0.83486863  0.30434056  0.62258487
        O -0.18030374 -0.22462371 -1.18585739
    End
    BondOrders
         1 5 1.0
         1 4 1.0
         1 3 1.0
         1 2 1.0
         2 6 1.0
         2 7 2.0
    End
End

Engine ForceField
EndEngine

eor
```

3) Specify Elements and coordinates (UFF only):

The third input format is similar to the second, but without a Bonds section in System:

```
$AMSBIN/ams << eor

Task GeometryOptimization

System
```

(continues on next page)

(continued from previous page)

```
    Atoms
        C  1.36012328 -0.14520095  0.60144543
        C  0.00000000  0.00000000  0.00000000
        H  2.09833847 -0.46327872 -0.16560721
        H  1.32657807 -0.90546800  1.40917410
        H  1.67935140  0.82750664  1.02977296
        H -0.83486863  0.30434056  0.62258487
        O -0.18030374 -0.22462371 -1.18585739
    End
End

Engine ForceField
EndEngine

eor
```

The GUI generates inputs of the second or third type, depending on the "Use existing bonds" setting in the ForceField main tab. Note that to specify the MM Atom Types, the charges also need to be set. UFF has automatic bond guessing and a very simple automatic charge guessing only assigning charges to atoms of water molecules.

3.2 Antechamber integration

For the GAFF force field there is an experimental integration of the Antechamber toolkit (http://ambermd.org/antechamber/antechamber.html) for automatic atom typing. This allows the GAFF force field to be used with option 2 (only bonds and coordinates specified) and option 3 (coordinates only). As of the 2020 release of AMS, this option is still considered experimental and disabled by default. It can be enabled and configured from the input:

```
AntechamberIntegration Yes/No
```

AntechamberIntegration

Type Bool

Default value No

GUI name Automatic atom typing

Description EXPERIMENTAL: Use the Antechamber program to automatically determine atom types for the GAFF force field. This may run a geometry optimization with MOPAC under the hood in order to determine the charges (see keyword AntechamberTask), which might not work for very large systems.

```
AntechamberTask [GeometryOptimization | SinglePoint]
```

AntechamberTask

Type Multiple Choice

Default value GeometryOptimization

Options [GeometryOptimization, SinglePoint]

Description If antechamber is envoked to guess atomtypes and charges (GAFF force field), select the task for charge guessing with MOPAC

CHAPTER

FOUR

FORCEFIELD ENGINE OPTIONS

Details of the ForceField engine can be set via its input block. Some option are specific to UFF and others to other force fields.

4.1 Common options

These options apply to any force field.

4.1.1 Type

There are a few predefined force field types, that, if used, require no other input.

```
Type [UFF | Amber95 | GAFF | Tripos5.2 | APPLE&P | UserDefined]
```

Type

Type Multiple Choice

Default value UFF

Options [UFF, Amber95, GAFF, Tripos5.2, APPLE&P, UserDefined]

Description Type of force field to be used

4.1.2 Non-bonded screening

The long range interaction (dispersion and Coulomb) are the most expensive to evaluate. This gives you the option to screen the interaction more aggressively.

```
NonBondedCutoff float
```

NonBondedCutoff

Type Float

Default value 15.0

Unit Angstrom

Description Distance beyond which the non-bonded pair interactions (Coulomb and Van der Waals) will be ignored.

The interactions are smoothly damped starting from 0.9*NonBondedCutoff.

Has no effect on the Coulomb term for 3D-periodic systems, as Ewald summation is used.

It is usually a good idea to add some "skin" to the cutoff above when it's used for computing a neighbor list for changing geometries (e.g. during molecular dynamics or geometry optimization). This way, the neighbor list will not need to be re-computed when atoms move a little. This may save some time because generating a neighbor list can be quite costly. The following option sets the thickness of the "skin":

```
NeighborListSkin float
```

NeighborListSkin

Type Float

Default value 2.5

Unit Angstrom

Description Thickness of the buffer region added to the NonBondedCutoff when building a neighbor list.

Note: This option also affects the cutoff used when generating a neighbor list in the real-space part of the Ewald summation but then it is added to the cutoff radius is used there.

4.1.3 Feedback

If you want to know more about the details of the force field you should crank up the verbosity.

```
Verbosity [Silent | Normal | Verbose | VeryVerbose]
```

Verbosity

Type Multiple Choice

Default value Silent

Options [Silent, Normal, Verbose, VeryVerbose]

Description Controls the verbosity of the engine.

4.1.4 Bonds usage

Bonds can be specified in the input, still you may not want to use those. Here are some options to control this.

```
BondsUsage [Input | None | Guess | Auto]
```

BondsUsage

Type Multiple Choice

Default value Auto

Options [Input, None, Guess, Auto]

Description Controls what bonds are used by the engine. The choice auto means: guess in case there are no bonds. Guessing only happens at the first MD step, or first geometry optimization step.

4.1.5 Ewald summation

For periodic systems the Ewald summation is performed for the Coulomb interaction. It has a couple of options:

```
EwaldSummation
   Alpha float
   Enabled Yes/No
   GridSpacing float
   RealSpaceCutoff float
   Tolerance float
End
```

EwaldSummation

Type Block

Description Configures the details of the particle mesh Ewald (PME) summation of the Coulomb interaction.

Alpha

Type Float

Default value -1.0

Unit 1/Angstrom

Description This parameter shifts the workload from real space (smaller alpha) to reciprocal space (larger alpha). Using a larger [Alpha] without decreasing [GridSpacing] may increase the error in the reciprocal-space contribution. Set to zero to disable the reciprocal-space Ewald part. Negative value means the [Alpha] will be determined automatically from the [Tolerance] and [RealSpaceCutoff] values.

Enabled

Type Bool

Default value Yes

Description Set to false to use real-space pair summation instead of the Ewald, which is the default and the only option for molecules, 1D and 2D periodic systems.

GridSpacing

Type Float

Default value 0.5

Unit Angstrom

Description Grid spacing in the particle mesh Ewald method. Smaller grid spacing will make the reciprocal energy calculation more accurate but slower. Using a larger [Alpha] value may require a smaller GridSpacing to be accurate.

RealSpaceCutoff

Type Float

Default value 0.0

Unit Angstrom

Description Set the cutoff value for the real-space summation. Zero means the internal defaults will be used depending on the [Alpha] (if Alpha=0 then the cutoff will be set to 50 Bohr, otherwise to 20 Bohr).

Tolerance

Type Float

Default value 1e-10

Description Value of the error function that should be used to determine the cutoff radius for real-space Ewald summation if [Alpha] is set on input. Alternatively, if the [RealSpaceCutoff] is set but [Alpha] is not then the [Tolerance] value affects the [Alpha]. Larger values will make the real-space summation faster but less accurate.

4.1.6 Disabling energy terms

By default all force field energy terms are calculated, however, you can disable each one of them individually.

```
EnergyTerms
   Angle Yes/No
   Coulomb Yes/No
   Dispersion Yes/No
   Inversion Yes/No
   Stretch Yes/No
   Torsion Yes/No
End
```

EnergyTerms

Type Block

Description expert key, that allows you to disable specific energy terms.

Angle

Type Bool

Default value Yes

Description Whether to use angle (bend) energy.

Coulomb

Type Bool

Default value Yes

Description Whether to use coulomb energy.

Dispersion

Type Bool

Default value Yes

Description Whether to use dispersion energy.

Inversion

Type Bool

Default value Yes

Description Whether to use inversion energy.

Stretch

Type Bool

Default value Yes

Description Whether to use stretch energy.

Torsion

Type Bool

Default value Yes

Description Whether to use torsion energy.

4.1.7 Guessing or loading partial charges

The UFF forcefield has some very rudimentary partial charges guessing, only setting charges for atoms in water molecules. By default the partial charges in a force field calculation are zero. Essentially you will always need to specify atomic charges to make the results more realistic, either via the input or using one or the following options.

See also example *LoadCharges* (page 40), and *ChargedMolecules* (page 30).

GuessCharges

The simplest way is the use the GuessCharges key, that uses an engine that can calculate atomic charges. By default DFTB is used. DFTB is of course much more expensive than a forcefield, but if you run a MD calculation you can maybe afford a single DFTB calculation on the system.

```
GuessCharges Yes/No
```

GuessCharges

Type Bool

Default value No

Description Use another engine to calculate/guess the charges to be used by the force field.

If you want to control the engine use the GuessChargesConfig key.

```
GuessChargesConfig
   EngineType string
End
```

GuessChargesConfig

Type Block

Description Guess charges to be used by the forcefield

EngineType

Type String

Default value dftb

Description Engine that can calculate or guess charges

LoadCharges

You have more control over the charge guessing, by loading the charges of another calculation. This way you can set any engine specific detail, such as the basis set, or functional.

You can load charges form a previous calculation to be used as force field charges.

```
LoadCharges
   File string
   Section string
   Variable string
End
```

LoadCharges

Type Block

Description Load charges from a file to be used as forcefield charges

File

Type String

Description Name of the (kf) file

Section

Type String

Default value AMSResults

Description Section name of the kf file

Variable

Type String

Default value Charges

Description variable name of the kf file

4.2 Amber force field options

These options are relevant for the Amber and GAFF force fields:

```
AllowMissingParameters Yes/No
```

AllowMissingParameters

Type Bool

Default value No

Description When parameters are not found for bonds, angles, dihedrals, or inversions, the first entry in the database will be used.

```
CheckDuplicateRules Yes/No
```

CheckDuplicateRules

Type Bool

Default value Yes

Description The database could contain duplicate entries. For torsions this is a feature, and the potentials will be added. For all other terms this is no allowed, and if detected the program stops. One should fix the database or set the checking to false. As always the last entry will be used.

```
ForceFieldFile string
```

ForceFieldFile

Type String

Default value

GUI name Force field library

Description Path to the force field parameter file

4.3 UFF options

The following options are only relevant for the UFF force field:

```
UFF
   AtomTypesFile string
   Database string
   ElementsFile string
   Library [UFF | UFF4MOF | UFF4MOF-II]
End
```

UFF

Type Block

Description Option for the UFF force filed.

AtomTypesFile

Type String

Default value mmatomtypes_db

Description Expert option: Select the file that defines how UFF determines the atom types

Database

Type String

Default value general_db

Description Expert option: Select the file that defines the UFF parameters per atom type

ElementsFile

Type String

Default value elements_db

Description Expert option: Select the file that defines the elements known to UFF

Library

Type Multiple Choice

Default value UFF

Options [UFF, UFF4MOF, UFF4MOF-II]

GUI name Force field library

Description Selects the used parameter library.

4.4 APPLE&P force field options

The *ForceFieldFile* (page 17) key is mandatory and it should contain path to the APPLE&P forcefield file. This file is usually tailored for each system specifically.

Additionally, the following options are relevant for the APPLE&P force field.

```
DipoleConvergenceThreshold float
```

DipoleConvergenceThreshold

Type Float

Default value 1e-06

Unit eBohr

Description Convergence criterion for induced point dipoles, in atomic units. When the length of every atomic delta_mu vector between two iterations becomes below the tolerance, the procedure is considered converged.

The repulsion/dispersion and Coulomb interaction between atoms connected by a bond or by a valence angle are excluded in APPLE&P. Those between atoms connected by a dihedral (the so called 1-4 neighbors) may be scaled down and the scaling factors can be changed using the following options:

```
APPLE&P
   LongRangeCorrection Yes/No
   MuMu14Scaling float
   QMu14Scaling float
   QQ14Scaling float
   RD14Scaling float
End
```

APPLE&P

Type Block

Description Options for the APPLE&P force field.

LongRangeCorrection

Type Bool

Default value Yes

GUI name Add long-range correction

Description Add a long-range dispersion correction to the energy and pressure for 3D-periodic systems.

This correction should be enabled only for a homogeneous liquid.

MuMu14Scaling

Type Float

Default value 1.0

GUI name Mu-Mu 3rd-neighbor scaling

Description Scaling factor for dipole-dipole interactions between atoms connected to 3rd order (via a dihedral).

QMu14Scaling

Type Float

Default value 0.2

GUI name Q-Mu 3rd-neighbor scaling

Description Scaling factor for charge-dipole interactions between atoms connected to 3rd order (via a dihedral).

QQ14Scaling

Type Float

Default value 1.0

GUI name Q-Q 3rd-neighbor scaling

Description Scaling factor for charge-charge interactions between atoms connected to 3rd order (via a dihedral).

RD14Scaling

Type Float

Default value 1.0

GUI name RD 3rd-neighbor scaling

Description Scaling factor for repulsion/dispersion interactions between atoms connected to 3rd order (via a dihedral).

4.5 Offloading calculations to LAMMPS

The ForceField engine can optionally offload the evaluation of energies and forces to LAMMPS (https://lammps.org/) to accelerate the calculation, possibly leveraging a GPU. In this mode, the engine will still set up all force field parameters as usual, but instead of evaluating the potential directly in AMS, the engine converts the parameters into a a LAMMPS data file and then invokes LAMMPS as an external pipe worker. As of AMS2023, this option is fully supported only for `Type UFF`.

4.5.1 Setting up LAMMPS

The interface between AMS and LAMMPS relies on the LAMMPS Python package which in turn requires LAMMPS to be built as a dynamic library. LAMMPS is not distributed together with the Amsterdam Modeling Suite and has to be installed separately from lammps.org (https://lammps.org/). If you already have an installation of LAMMPS including its Python module, it might be possible to use it directly, but often the easiest solution is to compile LAMMPS from source.

Consult the LAMMPS documentation for a detailed description of the steps necessary to install the LAMMPS Python module and shared library (https://docs.lammps.org/Python_install.html). For example, the following commands can be used to build LAMMPS including the necessary packages:

```
# Start in the top-level lammps directory (which contains the cmake/, src/, and lib/␣
↪subdirectories, among others)
mkdir build
cd build
```

(continues on next page)

(continued from previous page)

```
cmake ../cmake -DBUILD_LIB=yes -DBUILD_SHARED_LIBS=yes -DLAMMPS_EXCEPTIONS=yes -
↪DBUILD_MPI=no -DBUILD_OMP=yes -DPKG_OPENMP=yes -DPKG_GPU=yes -DPKG_MOLECULE=yes -
↪DPKG_EXTRA-MOLECULE=yes -DPKG_KSPACE=yes
cmake --build .
```

- `-DBUILD_SHARED_LIBS=yes` is strictly required by the Python module
- `-DLAMMPS_EXCEPTIONS=yes` is necessary to allow AMS to display any error messages generated by LAMMPS. A LAMMPS library compiled without this option will still work, but any error in LAMMPS will make AMS print only a generic error message.
- `-DBUILD_MPI=no -DBUILD_OMP=yes -DPKG_OPENMP=yes` are strongly recommended to enable OpenMP parallelization of the CPU code of LAMMPS. AMS currently cannot use the MPI framework to parallelize LAMMPS and linking to a MPI-enabled library can cause issues.
- `-DPKG_GPU=yes` builds LAMMPS with GPU support. Both AMD and nVidia cards are able to accelerate the calculation of non-bonded interactions significantly.
- `-DPKG_MOLECULE=yes -DPKG_EXTRA-MOLECULE=yes -DPKG_KSPACE=yes` enable the potential terms required by UFF.

Before running AMS, it is necessary to set up appropriate environment variables to make sure the `amspython` interpreter can find the `lammps` Python module. This can be done by any of the approaches outlined in the LAMMPS documentation (https://docs.lammps.org/Python_install.html#installing-the-lammps-python-module-and-shared-library). However, it is necessary to use `SCM_PYTHONPATH` instead of `PYTHONPATH` and `amspython` instead of `python3`. If you have followed the example above to build LAMMPS, the following commands should work for Bash and similar shells on Linux:

```
# Here /path/to/lammps is again the path to the top-level lammps directory
export SCM_PYTHONPATH=/path/to/lammps/python
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/path/to/lammps/src
```

Afterwards, you can try running `amspython -c 'import lammps; lammps.lammps()'` to test if the environment is set up correctly. This command should print the LAMMPS version it found. Otherwise, an `ImportError` means the `SCM_PYTHONPATH` variable is probably not set correctly, and an error about `liblammps.so` suggests an incorrect `LD_LIBRARY_PATH`.

4.5.2 Input options

```
LAMMPSOffload
   Enabled Yes/No
   Input
   UseGPU Yes/No
   UseOpenMP Yes/No
   WorkerCommand string
End
```

LAMMPSOffload

Type Block

Description Offload the calculation to LAMMPS via AMSPipe.

Enabled

Type Bool

Default value No

Description Enable offloading the force field evaluation to LAMMPS instead of handling it internally in AMS. This is currently only supported for Type=UFF.

`Input`

Type Block

Description Commands to be passed to LAMMPS to set up the calculation. If this is left empty, AMS will generate a set of commands to set LAMMPS up according to the settings of the ForceField engine. Any LAMMPS commands entered in this input block will be used to set LAMMPS up instead of those generated by AMS. To merge the AMS-generated lines with your customizations, include lines like 'AMS somelammpskeyword' anywhere in this block. Any such line will be replaced by the AMS-generated line for 'somelammpskeyword'. Any text after 'somelammpskeyword' will be appended to the generated line verbatim, which can be used to modify the generated command by additional options. A special line 'AMS everything' will be replaced by the entire block of AMS-generated commands, except those overriden anywhere in this input block (defined manually or inserted using 'AMS somelammpskeyword'. Any customized Input block should probably include 'AMS read_data' near or at the end to load the AMS-generated data file defining the system.

`UseGPU`

Type Bool

Default value No

Description Accelerate LAMMPS calculations using a GPU. Requires a LAMMPS library built with the GPU package.

`UseOpenMP`

Type Bool

Default value No

Description Parallelize LAMMPS calculations using OpenMP threading. Requires a LAMMPS library built with the OMP package.

`WorkerCommand`

Type String

Default value exec "$AMSBIN/amspython" "$AMSHOME/scripting/scm/external_engines/lmpworker.py"

Description The command to execute to run the external worker. The command is executed in a subdirectory of the results directory. The LAMMPS input commands will be passed to the worker on standard input.

CHAPTER

FIVE

APPLE&P

APPLE&P is a polarizable forcefield suitable for many electrolytes and polymers.

To use the APPLE&P force field, you need a special license for APPLE&P that you can obtain from SCM.

5.1 APPLE&P use cases

APPLE&P has been specifically parametrized to the below list of molecules. For molecules not on the list, the force field may or may not be reasonable, and it is advised that you do your own testing.

5.1.1 Organic Molecules

- alcohols
- aldehydes
- alkanes, alkenes, alkynes
- amides
- esters
- ethers
- imides
- ketones
- partially fluorinated alkanes
- perfluoroalkanes

5.1.2 Energetic Materials

- FOX-7
- HMX
- hydrazines
- hydroxylammonium nitrate (HAN)
- hydroxylethyl hydrazinium nitrate (HEHN)
- PETN

- RDX
- TATB
- TNT

5.1.3 Polymers

Note: The automatic atom typing may be very slow for large polymers containing the following functional groups: nitro, isocyanate groups, phenyl rings and conjugated rings with carbon and nitrogen (i.e. imidazole), amide, imide and azide groups, as well as phosphorous groups with multiple oxygen atoms (phosphorous oxoacids and derivatives).

- polyimides
- polyamides
- polyesters
- polyethers (e.g., PEO)
- PIB
- PDMS
- PTFE
- polybutadiene
- polyalkanes

5.1.4 Solvents/Additives

- acetone
- alcohols
- cyclic carbonates (e.g., ethylene carbonate)
- ethers (e.g., dimethoxyethane)
- linear carbonates (e.g., dimethyl carbonate)
- nitriles (e.g., acetonitrile, succinonitrile, adiponitrile)
- partially fluorinated carbonates (e.g., FEC)
- partially fluorinated ethers
- perfluoroethers
- sulfones (cyclic and acyclic)
- water

5.1.5 SEI compounds

- dilithium dicarbonates
- lithium carbonates
- LiF
- Li_2CO_3

5.1.6 Cations

- ammoniums
- H_3O^+, hydronium
- hydraziniums
- imidazoliums
- K^+, potassium
- Li^+, lithium
- Mg^{2+}, magnesium
- morpholiniums
- Na^+, sodium
- piperidiniums
- pyrrolidiniums
- Zn^{2+}, zinc

5.1.7 Anions

- $B(CN)_4^-$, tetracyanoborate
- BF_4^-, tetrafluoroborate
- $BF_3CF_3^-$, $CF_3BF_3^-$, trifluoro(trifluoromethyl)borate
- $BF_3CH_3^-$, $CH_3BF_3^-$, methyltrifluoroborate
- BOB, bis(oxalate)borate
- $C(CN)_3^-$, tricyanomethanide
- CN^-, cyanide
- CN_7^-, azidotetrazolate
- CO_3^{2-}, carbonate
- F^-, fluoride
- FSI, bis(fluorosulfonyl)imide
- N_3^-, azide
- $N(CN)_2^-$, dicyanamide
- NO_3^-, nitrate

- OH^-, hydroxide
- PF_6^-, hexafluorophosphate
- $SO_3CF_3^-$, triflate
- TFSI, bistriflimide

5.1.8 Electrodes

Note: The automatic atom typing may be extremely slow for graphite sheets consisting of more than 30-40 atoms.

- graphite
- iridium

5.2 APPLE&P potential shapes

In general, APPLE&P uses similar expressions for the potentials, with some differences. For completeness' sake we list all APPLE&P potentials below.

See also:

Potential shapes (page 67) force other force fields.

- Bond: the same as the stretch potential above.

$$V^{\text{bond}} = \frac{1}{2} f_c (r - r_0)^2$$

- Bend: the same as the harmonic angle potential above.

$$V^{\text{bend}} = \frac{1}{2} f_c (\phi - \phi_0)^2$$

- Torsion: cyclic.

$$V^{\text{torsion}} = -\sum_{m=1}^{n} c_m \cos(m\phi)$$

- Out-of-plane angle: sum of three harmonic terms, each corresponding to an angle between the R_{ij} bond and the (jkl) plane, where j is the central atom and i, k, l are permutations of the other three atoms.

$$V^{\text{oop}} = \frac{1}{2} f_c (\phi_1^2 + \phi_2^2 + \phi_3^2)$$

- Dispersion: mix of the Buckingham and Lennard-Jones potentials.

$$V^{\text{dispersion}} = A e^{-Br} - \frac{C}{r^6} + \frac{D}{r^{12}}$$

- Electrostatic potential: charge-charge, charge-dipole and dipole-dipole. Interaction are excluded for the 1-2 and 1-3 neighbors and can be scaled for the 1-4 ones. For each atom, the self-consistent induced dipole moment is computed from its polarizability and the electric field due to other charges and dipoles. The latter includes the Thole damping.

$$V^{\text{elstat}} = \sum_{i>j} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} + \sum_i \vec{\mu_i} \cdot \vec{E_i}$$

5.3 APPLE&P Python functions

The `scm.appleandp` Python module contains the below functions.

For example usage, see the APPLE&P tutorial

appleandp_packmol(*molecules*, *forcefield_file*, ***kwargs*)

molecules: list of Molecule If they are ions they should have the molecule.properties.charge set to the charge

forcefield_file: str The force field file will be written here. It will be overwritten if it already exists

****kwargs: various options** See the documentation for the `packmol()` function

Returns: a PLAMS Molecule created with packmol

This function differs from the PLAMS implementation of packmol in that it also writes out a force field file for use with APPLE&P.

CHAPTER

SIX

EXAMPLES

The *$AMSHOME/examples/forcefield* directory contains many different example files, covering various aspects. This is a selection of relevant examples showing what the engine input looks like.

We do not repeat here all functionality that is available from the AMS driver level, see the AMS Examples.

6.1 Example: single point for aspirin with UFF

Download SP_aspirin.run

```
#!/bin/sh

$AMSBIN/ams << eor

Task SinglePoint

Properties Gradients=yes

System
    Atoms
      C       0.000000  0.000000  0.000000
      C       1.402231  0.000000  0.000000
      C       2.091015  1.220378  0.000000
      C       1.373539  2.425321  0.004387
      C      -0.034554  2.451759  0.016301
      C      -0.711248  1.213529  0.005497
      O      -0.709522  3.637718  0.019949
      C      -2.141910  1.166077 -0.004384
      O      -2.727881  2.161939 -0.690916
      C      -0.730162  4.530447  1.037168
      C      -0.066705  4.031914  2.307663
      H      -0.531323 -0.967191 -0.007490
      H       1.959047 -0.952181 -0.004252
      H       3.194073  1.231720 -0.005862
      H       1.933090  3.376356 -0.002746
      O      -2.795018  0.309504  0.548870
      H      -2.174822  2.832497 -1.125018
      O      -1.263773  5.613383  0.944221
      H      -0.337334  4.693941  3.161150
      H       1.041646  4.053111  2.214199
      H      -0.405932  3.005321  2.572927
    End
End
```

(continues on next page)

(continued from previous page)

```
Engine ForceField
EndEngine

eor
```

6.2 Example: partial charges and total charge

This example shows you some ways to use partial atomic charges for ions. The atomic charges should add up to the total charge.

Download ChargedMolecules.run

```
#!/bin/sh

# Here we consider a Butane molecule with the C in the end groups changed to N
# The ion has a charge of plus two.
# The (partial) atomic charges used in the forcefield need to add up to the total␣
↪charge


# There is explicit specification via the inut, Loading charges, and guessing charges


# Charges taken "manually" from a DFTB calc., specified via the System%Atoms block

export AMS_JOBNAME=C2H10N2++

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228   ForceField.
↪Charge=-0.22410670135964811E+000
             C -0.003379615065181385 0.005774705419397744 -0.4323726711546611  ␣
↪ForceField.Charge=0.48180334689251314E-001
             C 0.7147029862216027 -1.238161594687982      -0.9787933548287219  ␣
↪ForceField.Charge=0.48135748838326275E-001
             N 0.7166487936613554 -1.241183181538415       -2.471469363728673  ␣
↪ForceField.Charge=-0.22411104322926501E+000
             H -0.489254727061788 0.8470579056083724        1.393322418122902  ␣
↪ForceField.Charge=0.32704816770091061E+000
             H 0.9419603270459358 0.02859511576851682       1.443864570589139  ␣
↪ForceField.Charge=0.31201967764994959E+000
```

(continues on next page)

(continued from previous page)

```
             H -0.4954439060007188 -0.801643243084588       1.443905390218997  ␣
↪ForceField.Charge=0.31201670096380 3795E+000
             H 0.497012567535054 0.9247340386709353       -0.7538297483507286  ␣
↪ForceField.Charge=0.11242075479984 9087E+000
             H -1.049427360359161 0.0320094307806696      -0.7537906795101232  ␣
↪ForceField.Charge=0.11242585978687 3958E+000
             H 1.760678843023289 -1.264262738726653       -0.6572703272452628  ␣
↪ForceField.Charge=0.11241599762423 1642E+000
             H 0.2140471968077103 -2.156981931600252      -0.6574882069415752  ␣
↪ForceField.Charge=0.11241997010893 7881E+000
             H 1.207324110469442 -0.4308282262683778       -2.854859298860555  ␣
↪ForceField.Charge=0.31204388175039 1661E+000
             H -0.2304034502578771 -1.260970512043685      -2.855040966499596  ␣
↪ForceField.Charge=0.31204824370321 6099E+000
             H 1.200917269444972 -2.079821003490794        -2.803581935602674  ␣
↪ForceField.Charge=0.32704267574950 0744E+000
    End

    GuessBonds True
    Charge 2.0
End


Engine ForceField
EndEngine

eor


# Now the neutral molecule, to show it behaves differently

export AMS_JOBNAME=C2H10N2

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228   ForceField.
↪Charge=-0.57101127820504 4492E+000
             C -0.003379615065181385 0.005774705419397744 -0.4323726711546611  ␣
↪ForceField.Charge=-0.62820364671994 4775E-001
             C 0.7147029862216027 -1.238161594687982      -0.9787933548287219  ␣
↪ForceField.Charge=-0.62883774618314 6251E-001
             N 0.7166487936613554 -1.241183181538415       -2.471469363728673  ␣
↪ForceField.Charge=-0.57112556850037 7101E+000
             H -0.489254727061788 0.8470579056083724        1.393322418122902  ␣
↪ForceField.Charge=0.19436624728371 4433E+000
```

(continues on next page)

(continued from previous page)

```
              H 0.9419603270459358 0.02859511576851682       1.443864570589139  ␣
↪ForceField.Charge=0.194416030458134853E+000
              H -0.4954439060007188 -0.801643243084588       1.443905390218997  ␣
↪ForceField.Charge=0.194415295916033237E+000
              H 0.497012567535054 0.9247340386709353       -0.7538297483507286  ␣
↪ForceField.Charge=0.253223131130085044E-001
              H -1.049427360359161 0.0320094307806696      -0.7537906795101232  ␣
↪ForceField.Charge=0.253801306729796901E-001
              H 1.760678843023289 -1.264262738726653       -0.6572703272452628  ␣
↪ForceField.Charge=0.253595653793994673E-001
              H 0.2140471968077103 -2.156981931600252      -0.6574882069415752  ␣
↪ForceField.Charge=0.254060553518517668E-001
              H 1.207324110469442 -0.4308282262683778       -2.854859298860555  ␣
↪ForceField.Charge=0.194371032691725371E+000
              H -0.2304034502578771 -1.260970512043685      -2.855040966499596  ␣
↪ForceField.Charge=0.194368207022870332E+000
              H 1.200917269444972 -2.079821003490794        -2.803581935602674  ␣
↪ForceField.Charge=0.194436108106012751E+000
    End

    GuessBonds True
End


Engine ForceField
EndEngine

eor

# The remaining calculations are again on the +2 ion

# Charges obtained with GuessCharges, using by default dftb.

export AMS_JOBNAME=C2H10N2++.guessed

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228
              C -0.003379615065181385 0.005774705419397744 -0.4323726711546611
              C 0.7147029862216027 -1.238161594687982      -0.9787933548287219
              N 0.7166487936613554 -1.241183181538415       -2.471469363728673
              H -0.489254727061788 0.8470579056083724        1.393322418122902
              H 0.9419603270459358 0.02859511576851682       1.443864570589139
              H -0.4954439060007188 -0.801643243084588       1.443905390218997
              H 0.497012567535054 0.9247340386709353       -0.7538297483507286
```

(continues on next page)

(continued from previous page)

```
                H -1.049427360359161 0.0320094307806696      -0.7537906795101232
                H 1.760678843023289 -1.264262738726653       -0.6572703272452628
                H 0.2140471968077103 -2.156981931600252      -0.6574882069415752
                H 1.207324110469442 -0.4308282262683778       -2.854859298860555
                H -0.2304034502578771 -1.260970512043685      -2.855040966499596
                H 1.200917269444972 -2.079821003490794        -2.803581935602674
    End

    GuessBonds True
    Charge 2.0
End


Engine ForceField
   GuessCharges True
EndEngine

eor


# Charges are now both specified on input and also Guessed
# The values on the input (ForceField.Charge) are on purpose unreasonable
# The Guessed charges have a higher priority and the input values are ignored
# The output is a bit confusing as unused ForceField.Charge is printed

export AMS_JOBNAME=C2H10N2++.inputandguessed

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228   ForceField.
↪Charge=-1.224106970135964811E+000
                C -0.003379615065181385 0.005774705419397744 -0.4323726711546611  ␣
↪ForceField.Charge=0.481803346892513140E-001
                C 0.7147029862216027 -1.238161594687982      -0.9787933548287219  ␣
↪ForceField.Charge=0.481357488383262755E-001
                N 0.7166487936613554 -1.241183181538415       -2.471469363728673  ␣
↪ForceField.Charge=-0.224111043229265006E+000
                H -0.489254727061788 0.8470579056083724        1.393322418122902  ␣
↪ForceField.Charge=1.327048167700910608E+000
                H 0.9419603270459358 0.02859511576851682       1.443864570589139  ␣
↪ForceField.Charge=0.312019677649949589E+000
                H -0.4954439060007188 -0.801643243084588       1.443905390218997  ␣
↪ForceField.Charge=0.312016700963803795E+000
                H 0.497012567535054 0.9247340386709353       -0.7538297483507286  ␣
↪ForceField.Charge=0.112420754799849087E+000
```

(continues on next page)

(continued from previous page)

```
              H -1.049427360359161 0.0320094307806696      -0.7537906795101232  ␣
↪ForceField.Charge=0.11242585976873958E+000
              H 1.760678843023289 -1.264262738726653       -0.6572703272452628  ␣
↪ForceField.Charge=0.11241599762423164 2E+000
              H 0.2140471968077103 -2.156981931600252      -0.6574882069415752  ␣
↪ForceField.Charge=0.11241997010893788 1E+000
              H 1.207324110469442 -0.4308282262683778       -2.854859298860555  ␣
↪ForceField.Charge=0.31204388175039166 1E+000
              H -0.2304034502578771 -1.260970512043685      -2.855040966499596  ␣
↪ForceField.Charge=0.31204824370321609 9E+000
              H 1.200917269444972 -2.079821003490794        -2.803581935602674  ␣
↪ForceField.Charge=0.32704267574950074 4E+000
    End

    GuessBonds True
    Charge 2.0
End


Engine ForceField
   GuessCharges True
EndEngine

eor







# Charges obtained with Load Charges from a DFTB calc.

export AMS_JOBNAME=C2H10N2++.dftb

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task SinglePoint

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228
              C -0.003379615065181385 0.005774705419397744 -0.4323726711546611
              C 0.7147029862216027 -1.238161594687982      -0.9787933548287219
              N 0.7166487936613554 -1.241183181538415       -2.471469363728673
              H -0.489254727061788 0.8470579056083724        1.393322418122902
              H 0.9419603270459358 0.02859511576851682       1.443864570589139
              H -0.4954439060007188 -0.801643243084588       1.443905390218997
              H 0.497012567535054 0.9247340386709353       -0.7538297483507286
              H -1.049427360359161 0.0320094307806696      -0.7537906795101232
```

(continues on next page)

(continued from previous page)

```
                H 1.760678843023289 -1.264262738726653       -0.6572703272452628
                H 0.2140471968077103 -2.156981931600252      -0.6574882069415752
                H 1.207324110469442 -0.4308282262683778       -2.854859298860555
                H -0.2304034502578771 -1.260970512043685      -2.855040966499596
                H 1.200917269444972 -2.079821003490794        -2.803581935602674
    End

    Charge 2.0
End


Engine DFTB
EndEngine

eor

loadChargeFile='C2H10N2++.dftb.results/dftb.rkf'

export AMS_JOBNAME=C2H10N2++.loaded

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228
                C -0.003379615065181385 0.005774705419397744 -0.4323726711546611
                C 0.7147029862216027 -1.238161594687982      -0.9787933548287219
                N 0.7166487936613554 -1.241183181538415       -2.471469363728673
                H -0.489254727061788 0.8470579056083724        1.393322418122902
                H 0.9419603270459358 0.02859511576851682       1.443864570589139
                H -0.4954439060007188 -0.801643243084588       1.443905390218997
                H 0.497012567535054 0.9247340386709353       -0.7538297483507286
                H -1.049427360359161 0.0320094307806696      -0.7537906795101232
                H 1.760678843023289 -1.264262738726653       -0.6572703272452628
                H 0.2140471968077103 -2.156981931600252      -0.6574882069415752
                H 1.207324110469442 -0.4308282262683778       -2.854859298860555
                H -0.2304034502578771 -1.260970512043685      -2.855040966499596
                H 1.200917269444972 -2.079821003490794        -2.803581935602674
    End

    GuessBonds True
    Charge 2.0
End


Engine ForceField
   LoadCharges
```

(continues on next page)

(continued from previous page)

```
      File $loadChargeFile
   End
EndEngine

eor


# Charges are now both specified on input and also Guessed
# The values on the input (ForceField.Charge) are on purpose unreasonable
# The Guessed charges have a higher priority and the input values are ignored
# The output is a bit confusing as unused ForceField.Charge is printed

export AMS_JOBNAME=C2H10N2++.inputandloaded

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228   ForceField.
↪Charge=-1.224106970135964811E+000
             C -0.003379615065181385 0.005774705419397744 -0.4323726711546611  ␣
↪ForceField.Charge=0.481803346892513140E-001
             C 0.7147029862216027 -1.238161594687982      -0.9787933548287219  ␣
↪ForceField.Charge=0.481357488383262755E-001
             N 0.7166487936613554 -1.241183181538415       -2.471469363728673  ␣
↪ForceField.Charge=-0.224111043229265006E+000
             H -0.489254727061788 0.8470579056083724        1.393322418122902  ␣
↪ForceField.Charge=1.327048167700910608E+000
             H 0.9419603270459358 0.02859511576851682       1.443864570589139  ␣
↪ForceField.Charge=0.312019677649949589E+000
             H -0.4954439060007188 -0.801643243084588       1.443905390218997  ␣
↪ForceField.Charge=0.312016700963803795E+000
             H 0.497012567535054 0.9247340386709353       -0.7538297483507286  ␣
↪ForceField.Charge=0.112420754799849087E+000
             H -1.049427360359161 0.0320094307806696      -0.7537906795101232  ␣
↪ForceField.Charge=0.112425859786873958E+000
             H 1.760678843023289 -1.264262738726653       -0.6572703272452628  ␣
↪ForceField.Charge=0.112415997624231642E+000
             H 0.2140471968077103 -2.156981931600252      -0.6574882069415752  ␣
↪ForceField.Charge=0.112419970108937881E+000
             H 1.207324110469442 -0.4308282262683778       -2.854859298860555  ␣
↪ForceField.Charge=0.312043881750391661E+000
             H -0.2304034502578771 -1.260970512043685      -2.855040966499596  ␣
↪ForceField.Charge=0.312048243703216099E+000
             H 1.200917269444972 -2.079821003490794        -2.803581935602674  ␣
↪ForceField.Charge=0.327042675749500744E+000
    End
```

(continues on next page)

(continued from previous page)

```
    GuessBonds True
    Charge 2.0
End


Engine ForceField
   LoadCharges
      File $loadChargeFile
   End
EndEngine

eor


# this calculation should succeed, ignoring the default zero total charge and using␣
↪the atomic charges instead

export AMS_JOBNAME=C2H10N2++.warning

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228   ForceField.
↪Charge=-0.22410697013596 4811E+000
              C -0.003379615065181385 0.005774705419397744 -0.4323726711546611  ␣
↪ForceField.Charge=0.481803346892513140E-001
              C 0.7147029862216027 -1.238161594687982      -0.9787933548287219  ␣
↪ForceField.Charge=0.481357488383262755E-001
              N 0.7166487936613554 -1.241183181538415       -2.471469363728673  ␣
↪ForceField.Charge=-0.224111043229265006E+000
              H -0.489254727061788 0.8470579056083724        1.393322418122902  ␣
↪ForceField.Charge=0.327048167700910608E+000
              H 0.9419603270459358 0.02859511576851682       1.443864570589139  ␣
↪ForceField.Charge=0.312019677649949589E+000
              H -0.4954439060007188 -0.801643243084588       1.443905390218997  ␣
↪ForceField.Charge=0.312016700963803795E+000
              H 0.497012567535054 0.9247340386709353       -0.7538297483507286  ␣
↪ForceField.Charge=0.112420754799849087E+000
              H -1.049427360359161 0.0320094307806696      -0.7537906795101232  ␣
↪ForceField.Charge=0.112425859786873958E+000
              H 1.760678843023289 -1.264262738726653       -0.6572703272452628  ␣
↪ForceField.Charge=0.112415997624231642E+000
              H 0.2140471968077103 -2.156981931600252      -0.6574882069415752  ␣
↪ForceField.Charge=0.112419970108937881E+000
              H 1.207324110469442 -0.4308282262683778       -2.854859298860555  ␣
↪ForceField.Charge=0.312043881750391661E+000
```

(continues on next page)

(continued from previous page)

```
              H -0.2304034502578771 -1.260970512043685      -2.855040966499596  ␣
↪ForceField.Charge=0.31204824370321609E+000
              H 1.200917269444972 -2.079821003490794        -2.803581935602674  ␣
↪ForceField.Charge=0.32704267574950074E+000
    End

    GuessBonds True
End


Engine ForceField
EndEngine

eor


# this calculation should trigger a warning because the explicitly set Charge doesn't␣
↪match

export AMS_JOBNAME=C2H10N2++.warning

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228   ForceField.
↪Charge=-0.22410697013596481E+000
              C -0.003379615065181385 0.005774705419397744 -0.4323726711546611  ␣
↪ForceField.Charge=0.48180334689251314E-001
              C 0.7147029862216027 -1.238161594687982      -0.9787933548287219  ␣
↪ForceField.Charge=0.48135748838326275E-001
              N 0.7166487936613554 -1.241183181538415       -2.471469363728673  ␣
↪ForceField.Charge=-0.22411104322926501E+000
              H -0.489254727061788 0.8470579056083724        1.393322418122902  ␣
↪ForceField.Charge=0.32704816770091061E+000
              H 0.9419603270459358 0.02859511576851682       1.443864570589139  ␣
↪ForceField.Charge=0.31201967764994959E+000
              H -0.4954439060007188 -0.801643243084588       1.443905390218997  ␣
↪ForceField.Charge=0.31201670096380379E+000
              H 0.497012567535054 0.9247340386709353        -0.7538297483507286  ␣
↪ForceField.Charge=0.11242075479984909E+000
              H -1.049427360359161 0.0320094307806696       -0.7537906795101232  ␣
↪ForceField.Charge=0.11242585978687396E+000
              H 1.760678843023289 -1.264262738726653        -0.6572703272452628  ␣
↪ForceField.Charge=0.11241599762423164E+000
              H 0.2140471968077103 -2.156981931600252       -0.6574882069415752  ␣
↪ForceField.Charge=0.11241997010893788E+000
```

(continues on next page)

(continued from previous page)

```
              H 1.207324110469442 -0.4308282262683778       -2.854859298860555  ␣
↪ForceField.Charge=0.31204388175039166E+000
              H -0.2304034502578771 -1.260970512043685      -2.855040966499596  ␣
↪ForceField.Charge=0.31204824370321609E+000
              H 1.200917269444972 -2.079821003490794        -2.803581935602674  ␣
↪ForceField.Charge=0.32704267574950074E+000
    End

    GuessBonds True
    Charge 1.0
End


Engine ForceField
EndEngine

eor

# finally let us make on purpose an error (total charge not set with strict checking␣
↪enabled)
# this calculation is supposed to fail, and not print an energy

echo "The following error is intended"

export AMS_JOBNAME=C2H10N2++.inputerror

rm -rf $AMS_JOBNAME.results

"$AMSBIN/ams" << eor

Task GeometryOptimization

Properties Charges=yes

GeometryOptimization
  Convergence Gradients=1.0e-6
End

System
    Atoms
        N -0.005073166519502884 0.008770355996886468   1.060120635725228   ForceField.
↪Charge=-0.22410697013596481E+000
              C -0.003379615065181385 0.005774705419397744 -0.4323726711546611  ␣
↪ForceField.Charge=0.48180334689251314E-001
              C 0.7147029862216027 -1.238161594687982      -0.9787933548287219  ␣
↪ForceField.Charge=0.48135748838326276E-001
              N 0.7166487936613554 -1.241183181538415       -2.471469363728673  ␣
↪ForceField.Charge=-0.22411104322926501E+000
              H -0.489254727061788 0.8470579056083724        1.393322418122902  ␣
↪ForceField.Charge=0.32704816770091061E+000
              H 0.9419603270459358 0.02859511576851682       1.443864570589139  ␣
↪ForceField.Charge=0.31201967764994959E+000
              H -0.4954439060007188 -0.801643243084588       1.443905390218997  ␣
↪ForceField.Charge=0.31201670096380379E+000
              H 0.497012567535054 0.9247340386709353        -0.7538297483507286  ␣
↪ForceField.Charge=0.11242075479984909E+000
              H -1.049427360359161 0.0320094307806696       -0.7537906795101232  ␣
↪ForceField.Charge=0.11242585978687396E+000
```

(continues on next page)

(continued from previous page)

```
              H 1.760678843023289 -1.264262738726653       -0.6572703272452628  ␣
↪ForceField.Charge=0.11241599762423164E+000
              H 0.2140471968077103 -2.156981931600252      -0.6574882069415752  ␣
↪ForceField.Charge=0.11241997010893788E+000
              H 1.207324110469442 -0.4308282262683778       -2.854859298860555  ␣
↪ForceField.Charge=0.31204388175039166E+000
              H -0.2304034502578771 -1.260970512043685      -2.855040966499596  ␣
↪ForceField.Charge=0.31204824370321609E+000
              H 1.200917269444972 -2.079821003490794        -2.803581935602674  ␣
↪ForceField.Charge=0.32704267574950074E+000
    End

    GuessBonds True
End


Engine ForceField
    DoChargeCheck True
EndEngine

eor
```

6.3 Example: load partial charges from another engine

You can use *LoadCharges* (page 16) to load charges from another calculation. The key *GuessCharges* (page 15) is a simplified version of this, being more convenient, but less flexible.

Download LoadCharges.run

```
#!/bin/sh

# First we calculate the charges for a system

# Here we use the dftb engine, but any engine can be used for this purpose


export AMS_JOBNAME=CalculateCharges

rm -rf $AMS_JOBNAME.results

$AMSBIN/ams << eor
Task SinglePoint

Properties Charges=yes

System
   Atoms
      C 0.0 0.0 0.0
      O 1.13 0.0 0.0
      C 0.0 0.0 2.0
      O 1.13 0.0 2.0
   End
End
```

(continues on next page)

(continued from previous page)

```
Engine DFTB
EndEngine

eor

# let us first optimize without charges

export AMS_JOBNAME=DoNotUseCharges

rm -rf $AMS_JOBNAME.results

$AMSBIN/ams << eor
Task GeometryOptimization

GeometryOptimization
   Convergence Step=1.0e-3
End

System
   Atoms
      C 0.0 0.0 0.0
      O 1.13 0.0 0.0
      C 0.0 0.0 2.1
      O 1.13 0.0 1.9
   End
End

Engine ForceField
EndEngine

eor


# Now that we have charges from our previous fancy calculation, let us use them for a␣
↪UFF geometry optimization
# * The name of the file depends on the engine used (in this case dftb.rkf)
# * The geometry does not need to be the same

export AMS_JOBNAME=LoadCharges

rm -rf $AMS_JOBNAME.results

$AMSBIN/ams << eor
Task GeometryOptimization

GeometryOptimization
   Convergence Step=1.0e-3
End

System
   Atoms
      C 0.0 0.0 0.0
      O 1.13 0.0 0.0
      C 0.0 0.0 2.1
      O 1.13 0.0 1.9
   End
End
```

(continues on next page)

(continued from previous page)

```
Engine ForceField
   Verbosity Verbose
   LoadCharges File=CalculateCharges.results/dftb.rkf
EndEngine

eor

# Finally let us use the charge guessing, by default dftb is used for charge guessing

export AMS_JOBNAME=GuessCharges

rm -rf $AMS_JOBNAME.results

$AMSBIN/ams << eor
Task GeometryOptimization

GeometryOptimization
   Convergence Step=1.0e-3
End

System
   Atoms
      C 0.0 0.0 0.0
      O 1.13 0.0 0.0
      C 0.0 0.0 2.1
      O 1.13 0.0 1.9
   End
End

Engine ForceField
   Verbosity Verbose
   GuessCharges True
EndEngine

eor
```

6.4 Example: load atom types from a previous run

You can load the atom types from another force field calculation.

Observe that this is done with the `LoadForceFieldAtomTypes` inside the `System` block, see the System definition section of the AMS manual.

Download LoadTypes.run

```
#!/bin/sh

# In this example we use the amber forcefield that cannot guess atom types

# First we "calculate" the types

# We do this by specifying them as atom attributes (ForceField.Type)

export AMS_JOBNAME=CalculateTypes
```

(continues on next page)

(continued from previous page)

```
rm -rf $AMS_JOBNAME.results

$AMSBIN/ams << eor
Task GeometryOptimization

Properties Charges=yes

System
   Atoms
     C       1.94807   3.58290  -0.58162       ForceField.Charge=0.0       ForceField.
↪Type=CT
     H       1.69949   4.49893  -1.05273       ForceField.Charge=0.0       ForceField.
↪Type=HC
     H       2.99455   3.17964  -0.86304       ForceField.Charge=0.0       ForceField.
↪Type=HC
     C       0.94659   2.40054  -0.92364       ForceField.Charge=0.0       ForceField.
↪Type=CT
     C       1.94191   3.61595   1.09448       ForceField.Charge=0.0       ForceField.
↪Type=CT
     N      -1.74397  -3.46417   0.31178       ForceField.Charge=-0.9530   ForceField.
↪Type=N2
     C      -1.00720  -2.20758   0.33536       ForceField.Charge=0.8185    ForceField.
↪Type=CA
     C      -1.66928  -1.00652   0.31001       ForceField.Charge=-0.5215   ForceField.
↪Type=CM
     C      -0.92847   0.25653   0.34895       ForceField.Charge=0.0053    ForceField.
↪Type=CM
     N       0.43971   0.26735   0.38232       ForceField.Charge=-0.0484   ForceField.
↪Type=N*
     N       0.36409  -2.20477   0.28992       ForceField.Charge=-0.7584   ForceField.
↪Type=NC
     C       1.09714  -0.95413   0.22469       ForceField.Charge=0.7538    ForceField.
↪Type=C
     H      -2.89781  -3.50815   0.31746       ForceField.Charge=0.4234    ForceField.
↪Type=H
     H      -1.21484  -4.49217   0.31721       ForceField.Charge=0.4234    ForceField.
↪Type=H
     H      -2.80940  -0.93497   0.30550       ForceField.Charge=0.1928    ForceField.
↪Type=HA
     H      -1.55324   1.21497   0.33885       ForceField.Charge=0.1958    ForceField.
↪Type=H4
     C       1.23309   1.44017   0.30994       ForceField.Charge=0.0066    ForceField.
↪Type=CT
     O       2.58277  -1.01636   0.23914       ForceField.Charge=-0.6252   ForceField.
↪Type=O
     H       2.37276   1.25557   0.29984       ForceField.Charge=0.2902    ForceField.
↪Type=H2
     O       1.02358   2.43085   1.50880       ForceField.Charge=-0.2033   ForceField.
↪Type=OS
     H       1.17136   1.95097  -1.87367       ForceField.Charge=0.0       ForceField.
↪Type=HC
     H      -0.10600   2.77333  -0.80348       ForceField.Charge=0.0       ForceField.
↪Type=HC
     H       1.62170   4.54039   1.51392       ForceField.Charge=0.0       ForceField.
↪Type=H1
     H       2.99608   3.28749   1.41345       ForceField.Charge=0.0       ForceField.
↪Type=H1
```

(continues on next page)

(continued from previous page)

```
   End
End

Engine ForceField
   Type Amber95
EndEngine

eor


# Now we load the types from the previous result

# The result should be identical to the first calculation

export AMS_JOBNAME=LoadTypes

rm -rf $AMS_JOBNAME.results

$AMSBIN/ams << eor
Task GeometryOptimization

System
   Atoms
     C       1.94807   3.58290  -0.58162
     H       1.69949   4.49893  -1.05273
     H       2.99455   3.17964  -0.86304
     C       0.94659   2.40054  -0.92364
     C       1.94191   3.61595   1.09448
     N      -1.74397  -3.46417   0.31178
     C      -1.00720  -2.20758   0.33536
     C      -1.66928  -1.00652   0.31001
     C      -0.92847   0.25653   0.34895
     N       0.43971   0.26735   0.38232
     N       0.36409  -2.20477   0.28992
     C       1.09714  -0.95413   0.22469
     H      -2.89781  -3.50815   0.31746
     H      -1.21484  -4.49217   0.31721
     H      -2.80940  -0.93497   0.30550
     H      -1.55324   1.21497   0.33885
     C       1.23309   1.44017   0.30994
     O       2.58277  -1.01636   0.23914
     H       2.37276   1.25557   0.29984
     O       1.02358   2.43085   1.50880
     H       1.17136   1.95097  -1.87367
     H      -0.10600   2.77333  -0.80348
     H       1.62170   4.54039   1.51392
     H       2.99608   3.28749   1.41345
   End
   LoadForceFieldCharges    file=CalculateTypes.results
   LoadForceFieldAtomTypes  file=CalculateTypes.results
End

Engine ForceField
   Type Amber95
   Verbosity Verbose

EndEngine
```

(continues on next page)

(continued from previous page)

```
eor
```

6.5 Example: using amber or tripos

Download 2h2o.run

```
#! /bin/sh

AMS_JOBNAME=amber95 $AMSBIN/ams << eor

Task SinglePoint

System
  Atoms
       O      0.0000     0.0000     0.0000      ForceField.Type=OW          ForceField.
↪Charge=-0.8340
       H     -0.5220     0.2660    -0.7570      ForceField.Type=HW          ForceField.
↪Charge=0.4170
       H     -0.5220     0.2660     0.7570      ForceField.Type=HW          ForceField.
↪Charge=0.4170
       O      0.0000    -3.2000     0.0000      ForceField.Type=OW          ForceField.
↪Charge=-0.8340
       H      0.0570    -2.2440     0.0000      ForceField.Type=HW          ForceField.
↪Charge=0.4170
       H      0.9110    -3.4950     0.0000      ForceField.Type=HW          ForceField.
↪Charge=0.4170
  End
  BondOrders
        1  2  1.0
        1  3  1.0
        4  5  1.0
        4  6  1.0
  End
End

Engine ForceField
  Type Amber95
EndEngine
eor


AMS_JOBNAME=tripos5.2 $AMSBIN/ams << eor

Task SinglePoint

System
  Atoms
       O      0.0000     0.0000     0.0000      ForceField.Type=O.3         ForceField.
↪Charge=-0.8340
       H     -0.5220     0.2660    -0.7570      ForceField.Type=H           ForceField.
↪Charge=0.4170
       H     -0.5220     0.2660     0.7570      ForceField.Type=H           ForceField.
↪Charge=0.4170
```

(continues on next page)

(continued from previous page)

```
       O      0.0000    -3.2000     0.0000      ForceField.Type=O.3       ForceField.
↪Charge=-0.8340
       H      0.0570    -2.2440     0.0000      ForceField.Type=H         ForceField.
↪Charge=0.4170
       H      0.9110    -3.4950     0.0000      ForceField.Type=H         ForceField.
↪Charge=0.4170
  End
  BondOrders
        1   2   1.0
        1   3   1.0
        4   5   1.0
        4   6   1.0
  End
End

Engine ForceField
  Type Tripos5.2
EndEngine
eor
```

6.6 Example: calculate some simple properties for water

Download water_properties.run

```
#! /bin/sh

$AMSBIN/ams << eor

Task GeometryOptimization

Properties
  BondOrders Yes
  Charges Yes
  DipoleMoment Yes
  NormalModes Yes
End

System
  Atoms
       O  0.0000  0.0000   0.0000  ForceField.Type=OW  ForceField.Charge=-0.8340
       H -0.5220  0.2660  -0.7570  ForceField.Type=HW  ForceField.Charge=0.4170
       H -0.5220  0.2660   0.7570  ForceField.Type=HW  ForceField.Charge=0.4170
  End
  BondOrders
        1   2   1.0
        1   3   1.0
  End
End

Engine ForceField
  Type Amber95
  ForceFieldFile $AMSRESOURCES/ForceFields/amber95.ff
EndEngine
```

(continues on next page)

(continued from previous page)

```
eor

echo "MM Charges:"
$AMSBIN/amsreport ams.results/forcefield.rkf -r "AMSResults%Charges"
```

6.7 Example: single point for ammonia with TRIPOS

Download Ammonia_Tripos.run

```
#!/bin/sh

#
# This is a technical example comparing analytical and numerical gradients for the␣
↪Tripos forcefield
#
# The allowMissingParameters is a trick to run this molecule even though it is not␣
↪fully supported by the definition file
#

for num in no yes
do

export AMS_JOBNAME=tripos3.num=$num

rm -rf $AMS_JOBNAME.results

$AMSBIN/ams << eor

Task SinglePoint
Properties Gradients=yes

EngineDebugging IgnoreGradientsRequest=$num IgnorePreviousResults=true

NumericalDifferentiation NuclearStepSize=1.0e-4


System
  Atoms
    N       0.00000000       0.00000000       0.26448000  ForceField.Charge=0.0 ␣
↪ForceField.Type=N.2
    H      -0.48379000       0.83795000      -0.08816000  ForceField.Charge=0.0 ␣
↪ForceField.Type=H
    H      -0.48379000      -0.83795000      -0.08816000  ForceField.Charge=0.0 ␣
↪ForceField.Type=H
    H       0.96758000      -0.00000000      -0.08816000  ForceField.Charge=0.0 ␣
↪ForceField.Type=H
  End

  BondOrders
         1 2 1.0
         1 3 1.0
         1 4 1.0
  End
End
```

(continues on next page)

(continued from previous page)

```
Engine ForceField
  Type Tripos5.2
  AllowMissingParameters yes
EndEngine
eor

done
```

CHAPTER

SEVEN

REQUIRED CITATIONS

When you publish results in the scientific literature that were obtained with programs of the AMS package, you are required to include references to the program package with the appropriate release number, and a few key publications.

For calculations with the UFF4MOF parameters: M.A. Addicoat, N. Vankova, I.F. Akter, and T. Heine, *An extension of the Universal Force Field to Metal-Organic Frameworks*, J. Chem. Theory Comput. 10, 880-891 (2013) (https://doi.org/10.1021/ct400952t)

For calculations with the UFF4MOFII parameters: D.E. Coupry, M.A. Addicoat, and T. Heine, *An Extension of the Universal Force Field for Metal-Organic Frameworks*, J. Chem. Theory Comput. 12, 5215-5225 (2016) (https://doi.org/10.1021/acs.jctc.6b00664)

For calculations with the APPLE&P forcefield: O. Borodin, *Polarizable Force Field Development and Molecular Dynamics Simulations of Ionic Liquids*, J. Phys. Chem. B 113, 11463–11478 (2009) (https://doi.org/10.1021/jp905220k)

7.1 External programs and libraries

Click here for the list of programs and/or libraries used in the AMS package. On some platforms optimized libraries have been used and/or vendor specific MPI implementations.

CHAPTER

EIGHT

PARAMETER FILES

The parameters of a forcefield are defined via one or more files, automatically set when using a standard *type* (page 11) . One way to change it is to use a parametrization made by someone else, that works better for systems of your interest.

A far more advanced use is to tweak an existing file, or create a new one, possibly to improve the results for specific systems of interest. For such a project the PARAMS tool might be helpful.

Here we distinguish the UFF and the AMBER case.

8.1 UFF parameters

The parameter set used by UFF can be changed via the *UFF* (page 17) key block. Aside from the standard UFF forcefield, we ship two parameter sets for Metal-Organic Frameworks:

UFF4MOF parameters

We ship the extended parameter set for Metal-Organic Frameworks created by M.A. Addicoat et al. (2013). Select the UFF4MOF_general_db, UFF4MOF_elements_db and UFF4MOF_mmatomtypes_db files to use these parameters, and check that the proper atom types are detected for your system or set them manually. Please see [*7* (page 71)] for details on the parameters.

UFF4MOFII parameters

We ship a second extended parameter set for Metal-Organic Frameworks created by D.E. Coupry et al. (2016). Select the UFF4MOFII_general_db, UFF4MOFII_elements_db and UFF4MOFII_mmatomtypes_db files to use these parameters, and check that the proper atom types are detected for your system or set them manually. Please see [*8* (page 71)] for details on the parameters.

8.1.1 User-modified force fields (expert option)

Finding good UFF parameters can be a challenging task, and any results with modified parameters should be checked very carefully. SCM has no experience with this, and the parameters supplied for UFF have not been generated by us. Feel free to test new parameters, and feel free to let us know if you have a good working set for a specialized situation.

General parameters file

The general_db file ($AMSHOME/atomicdata/UFF/general_db) contains all the parameters used to calculate the forces and energies. The format is:

MMAtomType, ri, phi, xi, di, psi, zmm, vsp3, vsp2, chi, nc.

The items in the list are:

- MMAtomType: name, max 5 characters

- ri: Valence Bond [Å]
- phi: Valence Angle [Degree]
- xi: Nonbond Distance [Å]
- di: Nonbond Energy [kcal/mol]
- psi: Nonbond scale [Number]
- zmm: Effective Charge [Charge]
- vsp3: sp3 Torsional Barrier [kcal/mol]
- vsp2: sp2 Torsional Barrier [kcal/mol]
- chi: Electronegativity
- nc: Number of directly attached atoms, aka coordination number. This is required for counting the number of possible dihedrals, and is defined only for the sp2 and sp3 centers (types 2, R, and 3)

The current set of parameters comes from the deMonNano program, and is a combination of published parameters and fitted data to fill in the gaps. The deMonNano documentation says the following about the parameters:

```
Implementation of the Universal Force Field (UFF) in deMonNano
--------------------------------------------------------------

As far as possible, UFF molecular mechanics forcefield in deMon
follows the published forcefield definition in [1]. In several
cases, the definitions and expressions in [1] are not consistent
with the published applications of the forcefield [1,5,6].
In those cases, an attempt was made to correct the errors and omissions,
using information from [2].

The following changed were made, compared to the published UFF
forcefield description (all equation and page numbers refer to [1]).

1. Sign error in Eq. 2 (equilibrium bond length) was corrected
   - electronegativity correction must be negative!

2. Equilibrium valence angle for O_3_z was corrected from 146.0
   degree to 145.45 degree.

3. Bending periodicity (Eq. 10) for linear coordination was
   corrected from 1 to 2.

4. Sign errors were corrected in eqs. 13 and and unnumbered equation
   for the beta parameter (between eqs. 13 and 14).

5. The reference value of the UFF amide force constant, of 105.5
   kcal/mol/rad**2 (p. 10028) is wrong. The results are consistent
   with the force constant of 211.0 kcal/mol/rad**2.

6. Equilibrium torsional angle for a bond between a group-6A atom
   (oxygen ...) and an sp2 atom (90 degree) is wrong (p. 10028).
   It should be 0 degree.

7. The conditional for the special-case sp2-sp3 torsion (p. 10029)
   is wrong, and should be inverted - see [4].

8. The overall shape of the UFF torsional potential degenerates to
```

(continues on next page)

(continued from previous page)

```
   a Heavyside function when one of the bond angles becomes linear,
   leading to failures in geometry optimization and force constant
   evaluation. The UFF torsional term was augmented with a smooth
   masking function, to avoid this.

9. UFF inversion potential is not defined in [1] for group 5A
   elements (from phosphorus down). Taking the equilibrium inversion
   coordinate of 87 degree, and the suitable expressions for the
   cosine weights (see uff_get_inversion_shape in "uff_database.f90")
   appears to reproduce published UFF structures and energetics.

The following atom types have been fully tested, and are believed
to reproduce published UFF forcefield results exactly. The examples
refer to the $deMon/examples/test.mm directory.

  Atom type  Example     Description
  ---------  -------     -----------

    Al3      alme3tma    Trivalent aluminum
    As3+3    asf3        Trivalent arsenic
    B_2      bcl3        Planar (sp2) boron
    B_3      b2h5nme2    Tetrahedral (sp2) boron, including
                         charge transfer adducts and borohydrates
    Br       bbr3        Univalent bromine
    C_1      c2h2, co    Linear (sp) carbon
    C_2      acetone     Planar tricoordinated (sp2) carbon
    C_3      c2h6        Tetrahedral (sp3) carbon
    C_R      c4h6        Resonant, variable bond order (sp2) carbon.
    Cl       socl2       Univalent chlorine
    F_       sof2ncl     Univalent fluorine
    Ge3      geh3ogeh3   Tetrahedral (sp3) germanium
    H_       h2o         Normal, non-bridging hydrogen
    H_b      b2h5nme2    Bridging hydrogen, for use in boranes
                         (NOT SUITABLE FOR H-BONDS!)
    I_       bi3         Univalent iodine
    N_1      ch3cn       Monocoordinated (sp) nitrogen, triple bond
    N_2      ch3n2ch3    Dicoordinated (sp2) nitrogen, single-double bond
    N_3      ch3nh2      Amine (sp3) nitrogen, three single bonds
    N_3+4    b2h5nme2    Charged amine (sp3) nitrogen, four single bonds
                         (THIS IS NOT A STANDARD UFF TYPE!)
    N_R      c5h5n       Resonant planar (sp2) nitrogen, for use in
                         aromatics and amides. For amides, use 1.41 bond
                         order!
    O_1      co          Special "co" type, one triple bond.
    O_2      acetone     One-coordinated (sp2) oxygen, one double bond.
    O_3      h2o         Two-coordinated (sp3) oxygen, two single bonds
    O_3_z    sih3osih3   Special two-coordinated oxygen, for use in
                         Si-O bonds
    O_R      c4h4o       Resonant planar (sp2) oxygen, also for use in
                         nitro groups and such.
    P_3+3    ph3         Pyramidal (sp3) phosphorus, three single bonds
    P_3+5    p4o7        Tetrahedral hypervalent phosphorus
    P_3+q    bh3ph3      Dative tetrahedral (sp3) phosphorus, watch for
                         the bond order!
    S_3+2    ch3sch3     Bent two-coordinated sulfur (sp3), two single bonds
    S_3+4    socl2       Pyramidal three-coordinated hypervalent sulfur
    S_3+6    so2cl2      Tetrahedral four-coordinated hypervalent sulfur
```

(continues on next page)

(continued from previous page)

```
    Se3+2    h2se        Bent two-coordinated (sp3) selenium
    Si3      si4o4h8     Tetrahedral silicon

Additionally, parameter sets for the following atom types are believed
be complete, and may be expected produce results identical to the published
UFF data: Li, Na, K_, Rb, Cs (Note that UFF does not specify atomic charges
- it is your responsibility to assign those, if charges are needed!)

For the remaining atom types, UFF definition [1] relies on an unpublished
set of electronegativities [2]. In deMon, these values were replaced by
Pauling electronegativities, scaled to fit published UFF electronegativities.
This can be expected to produce small deviations in bond lengths and bond
angles, compared to published UFF results.
```

If you wish to use other parameters, you should copy the general_db file, and rename it. This new file can also be placed outside of $AMSHOME/atomicdata/UFF.

Elements file

The elements_db file holds all the elements known to UFF. Keep in mind that these are not the MMAtomTypes, but pure chemical elements. The table contains for every element: atomic number, symbol, minimal valence number, maximum valence number, minimal bond order, maximal bond order. The data in the elements_db is mainly used for cleaning up the Pauling bond orders guessed by UFF, and will probably not need to be modified.

MM Atom Types file

The mmatomtypes_db file contains the matching rules for assigning MM atom types to chemical elements, based on their valence number, and the number of neighbor (bonded) atoms. The current implementation of UFF is limited to 6 MM atom types per element. The table contains for every MM atom type: Number of the element it belongs to, the i-th type of this element, the valence number corresponding to this MM atom type, number of neighbors this MM atom type has, the name of this MM atom type. The naming convention follows the original UFF paper [1]:

A five-character mnemonic label is used to describe the atom types. The first two characters correspond to the chemical symbol; an underscore appears in the second column if the symbol has one letter (e.g., `N_` is nitrogen, `Rh` is rhodium). The third column describes the hybridization or geometry: 1 = linear, 2 = trigonal, R = resonant, 3 = tetrahedral, 4 = square planar, 5 = trigonal bipyramidal, 6 = octahedral. Thus N_3 is tetrahedral nitrogen, while Rh6 is octahedral rhodium. The forth and fifth columns are used as indicators of alternate parameters such as formal oxidation state: Rh6+3 indicates an octahedral rhodium formally in the +3 oxidation state, e.g., Rh(NH3)_6^3+. H_b indicates a bridging hydrogen as in B2Hs O_3_z is an oxygen suited for framework oxygens of a zeolite lattice. P_3_q is a tetrahedral four-coordinate phosphorus used to describe organo-metallic coordinated phosphines.

You can copy the mmatomtypes_db and change it if you need to modify the atom typing behavior of UFF.

8.2 AMBER parameters

If you want to use a non-standard forcefield you can specify the *ForceFieldFile* (page 17).

Currently two formats are supported for non-uff forcefields. The first is the ADF/SCM related ".ff" format. The other is the much more widely used AMBER ".dat" format.

8.2.1 AMBER forcefield file

The format of the AMBER ".dat" files is described here http://ambermd.org/FileFormats.php#parm.dat. Currently the following features are not supported

- Parameter modification files
- The torsions have extra info about 1-4 scaling for nonbonded terms, ignored
- AMBER term 8: INPUT FOR H-BOND 10-12 POTENTIAL PARAMETERS
- AMBER term 9: INPUT FOR EQUIVALENCING ATOM SYMBOLS FOR THE NON-BONDED 6-12 POTENTIAL PARAMETERS

8.2.2 SCM forcefield file

An example of this is `$AFDHOME/atomicdata/ForceField/amber95.ff`. It has a flexible format and is fully self documented. It is not used outside of the ADF/SCM context.

The file must contain the force field parameters and the MM potential for each kind of MM interaction. Although predefined force field files (AMBER and SYBYL) are provided, these force field files can be customized. For example, one may want to change a particular force constant, or one may need to introduce a new atom type, for instance a transition metal. This section provides a detailed description of the force field file.

Format

The force field file is keyword driven with each key block defining parameters for each molecular mechanics interaction type such as bond types, angle types, torsion types, …etc. The key block begins with the keyword, such as "BONDS".

The lines that actually contain the parameters are sandwiched between two lines that contain "========". The lines between the keyword and the first line containing "========" are not read by the program. These lines are intended for the user to define the columns as shown below. There can be as many lines between the keyword and the first '=====' as needed.

Example:

```
BONDS
Atoms  pot  K              ro    Notes
i - j  type (kcal/molA^2) (Ang)
================================
...
CA  CA 1    938.0          1.400 amber95
CT  CT 1    620.0          1.526 amber95
...
================================
```

Force Field Atom types

The force field atom types are the labels given to each atom in the real system, which determine all interaction parameters involving that atom.

There are some limitations to the force field label types that the user can specify:

- Labels can be a maximum of four characters long, with no spaces.
- The atom types are case sensitive.
- They can contain letters, numbers and other characters except ',' '.' or '=' and tabs.

Example of atom types that are not compatible with the program: C.3, C 3, C=3, C_sp3, *

Examples of atom types that are correct: C_3, C3, Csp3, and C*

Wild Cards

Wild cards can be specified with the asterisk, '*'. Wild cards can be specified for angles, torsions and out-of-plane bends. Please refer to the specific sections for the limitations.

CAUTION: When using wild cards, place the wild cards at the beginning of the data section, beginning with the parameters with the most wild cards and ending with those that posses the least wild cards.

Example:

*	C_3	*	100.310	111.000	*two wild cards*
*	C_3	C_3	100.310	111.000	*one wild card*
C_2	C_3	C_3	100.310	111.000	*no wild cards*

If this ordering is not followed, then the wild card parameters will over-ride the specific parameters.

Miscellaneous Notes

- Do not remove the '======' separator lines.
- Units are in kcal/mol, Angstroms, degrees, amu unless otherwise specified
- Sections can be in any order; i.e. BENDS can come before BONDS.
- All keywords are case sensitive and most are in ALL-CAPS
- Input is all free format
- Blank lines will be ignored
- Comment lines can be added to parameter data sections by beginning the line with the '#' symbol.

Example:

```
H H 1.0080
HC H 1.0080
   # example of comment line denoted with # mark.
H1 H 1.0080
H2 H 1.0080
```

A (partial) Example File

Here we provide an example force field file to illustrate the format of the file. Only a limited number of parameters are included. A detailed description of each section of the force field file is provided in the next section.

```
FORCE_FIELD_SETTINGS
================================
ELSTAT_1-4_SCALE          1.0000
VDW_1-4_SCALE             1.0000
VDW_DEFAULT_POTENTIAL     1      (1:6-12 2:exp-6 3:exp purely repulsive)
DIELECTRIC_CONSTANT       1.000
================================

MASSES & ATOM LABELS
--------------------------------
force_field atomic
atom_type   symbol mass    NOTES
=========================
```

(continues on next page)

(continued from previous page)

```
C_3          C      12.0110 sp3 hybridized carbon
C_2          C      12.0110 sp2 hybridized carbon
C_1          C      12.0110 sp1 hybridized carbon
C_ar         C      12.0110 aromatic
N_3          N      14.0070
N_2          N      14.0070
O_3          O      15.9990
============================

BONDS Ebond = 0.5*K(r-ro)**2
------------------------------------
Atoms   pot
i - j   type K       R     NOTES
==========================
C_2 C_2 1    1340.00 1.335 WHITE_77
C_2 C_3 1     639.00 1.501 WHITE_75
C_3 C_3 1     633.60 1.540 *
C_3 N_2 1     760.20 1.440 *
==========================
BENDS Ebend = 0.5*k(a-ao)^2
-----------------------------------
Atoms           pot
i - j - k       type  K      theta  NOTES
===================================
*    C_2  *     1      78.79 120.00 WHITE_77
*    C_3  *     1      65.66 109.50 WHITE_77
*    C_ar *     1      78.79 120.00 *
C_ar C_2  N_2   1     131.31 120.00 *
C_3  C_3  C_ar  1      78.79 109.50 *
===================================

TORSIONS
-------------------------------------------
Atoms               pot
i - j - k - l       type  k       per         NOTES
===========================================
*    C_2  C_2  *    2     12.5000 -2.0
*    C_1  C_3  *    2      0.0000  1.0
C_2  C_2  C_3  *    2      0.1260 -3.0
C_3  C_2  C_3  *    2      0.1260  3.0
H    C_2  C_3  *    2      0.2740  3.0
*    C_ar C_ar C_ar 2      2.3500 -2.0
*    C_2  C_3  C_2  2      0.1260  3.0
*    C_2  C_3  C_3  2      0.1260  3.0
C_3  C_3  C_3  C_3  0      0.5000  3.0       no torsion potential
C_2  C_2  C_3  C_2  2      0.1260 -3.0
C_3  C_3  N_2  C_2  1      0.5000  4   180.0 This and the next 3 lines
&                          0.1500  3   180.0 are part of a multi-component
&                          0.5300  1     0.0 Fourier potential
C_3  C_3  C_2  N_2  1      0.1000  4     0.0
&                          0.0700  2     0.0 '&' is a continuation marker
===========================================

OUT-OF-PLANE
------------------------
Atoms           pot
i - j - k - l   type  K   NOTES
```

(continues on next page)

(continued from previous page)

```
========================
*   *   C_2 *   2     480 TRIPOS_85
*   *   N_2 *   2     120 TRIPOS_85
H   H   N_2 C_3 2     120 TRIPOS_85
C_3 H   N_2 *   2     120 TRIPOS_85
========================

VAN DER WAALS
atom(s)      Emin     Rmin    gamma NOTES
================================
C_3          0.1070  3.4000  12.00
C_2          0.1070  3.4000  12.00
C_ca         0.1070  3.4000  12.00
C_ar         0.1070  3.4000  12.00
C_1          0.1070  3.4000  12.00
N_3          0.0950  3.1000  12.00
N_2          0.0950  3.1000  12.00
N_2 - N_2 2 0.0950  3.1000  12.00 purely repulsive potential for this pair
================================

type  charge(e)  NOTES
===============
OW    -0.82      TIP3P water model
HW     0.41      TIP3P water model
===============
```

Section by Section Description

FORCE_FIELD_SETTINGS Key block (required) This key block specifies various global options for the force field file, mostly concerned with the treatment of the non-bonded potentials.

```
FORCE_FIELD_SETTINGS
================================
ELSTAT_1-4_SCALE           0.5
VDW_1-4_SCALE              0.5
VDW_DEFAULT_POTENTIAL      1     (1:6-12)
DIELECTRIC_CONSTANT        1.000
================================
```

ELSTAT_1-4_SCALE & VDW_1-4_SCALE Most force fields scale the non-bonded interactions by a factor of 0.5 if the atoms are the terminal atoms of a defined torsion. This scaling factor, which is termed the 1-4 scaling factor, can also be different for the electrostatic potential and for the Van der Waals potentials and thus they are separately defined in the input.

VDW_DEFAULT_POTENTIAL Only the Lennard Jones potential (option 1) is implemented. The Lennard Jones interaction energy between atoms A and B is defined as a function of the distance between atoms A and B (R^{AB}). The two parameters are D_0 and R_0.

$$E_{VDW}(R^{AB}) = D_0\left[\left(\frac{R_0}{R^{AB}}\right)^{12} - 2\left(\frac{R_0}{R^{AB}}\right)^{6}\right]$$

DIELECTRIC_CONSTANT This option is ignored.

BONDS Key block (required) This key block specifies the potential type and parameters for each kind of MM bond stretching interaction. An example is given below.

```
BONDS
Atoms pot   K              ro    NOTES
i - j type  (kcal/molA^2) (Ang)
================================
CA CA 1     938.0          1.400 amber95
CT CT 1     620.0          1.526 amber95
HC Zr 0       0.0                no potential found
================================
```

The first two columns are the atom types (up to four characters long) and the third column is an integer specifying the potential type.

BOND potential type		constants required (in order)
0	no potential	none
1	simple harmonic: $E_b^{ij} = 1/2\ K\ (R_{ij} - R_o)^2$ AMBER95, Sybyl	K, R_o

BENDS Key block (required) This key block specifies the potential type and parameters for each kind of MM bond angle interaction. An example is given below.

```
BENDS
Atoms     pot   k          ao     NOTES
i - j - k type  (kcal/mol) deg
================================
*  CA *   1      70.00      120.00 example of wild card
*  CA CA  1     126.00      120.00
CA CA N2  1     140.00      120.10 amber95 N2-CA-CM
CA CA CT  1     140.00      120.00 amber95
================================
```

The first three columns specify the atom types and the fourth column is an integer specifying the potential type. The angle bend potential types are described in the table below with the additional constants required.

BEND potential type		constants required (in order)
0	no potential	none
1	theta harmonic: $E_\theta^{ijk} = 1/2\ K_\theta\ (\theta_{ijk} - \theta_o)^2$ AMBER95, SYBYL	K_θ, θ_o (θ in degrees)

Notice that wild cards can be specified for both terminal positions of the bend or just one as in the example above. It is important that the parameters be ordered from the least specific (those containing the most wild cards) to the most specific parameters.

TORSIONS Key block (required) This key block specifies the potential type and parameters for each kind of MM bond torsion interaction. For the bond stretching and bending potentials, only one potential has to date been implemented since both AMBER and SYBYL both use simple harmonic potentials. However, AMBER and SYBYL use different

functional forms to represent the torsion potentials, each with their own set of parameters. The AMBER and SYBYL torsional potentials used in this program are defined in the table below.

TORSION potential type		constants required (in order)
0	no potential	none
1	AMBER: $E_{tors} = \frac{K_{tor}}{N_{tors}}[1 + \cos(n\phi - \phi_o)]$	K_i, n_i (periodicity-integer), $\varphi_{o,i}$ (phase shift)
2	SYBYL: $E_{tors} = \frac{1}{2}K_{tor}[1 + \cos(n\phi_o)\cos(n\phi)]$	K, s

Notice that the two potentials have a different number of parameters. For example, when the program reads 'potential type' number 1, it will expect three parameters K_i , n_i , $\phi_{o,i}$. Further notice that the AMBER torsional potential is a sum of Fourier components (this is what the index i refers to).

Below is an example of the TORSIONS key block, made up of AMBER force field types.

```
TORSIONS
Atoms                    pot            per.  shift
i - j - k - l            type  k        n     to       NOTES
=================================================
*     CV    NB    *      1     2.4000   2     180.0    JCC,7,(1986),230
*     CW    NA    *      1     1.5000   2     180.0    JCC,7,(1986),230
&                          0.1000   3       0.0
C     N     CT    C      1     0.2000   2     180.0
N     CT    C     N      1     0.4000   4     180.0
&                          1.3500   2     180.0
&                          0.7500   1     180.0
CT    CT    N     C      1     0.5000   4     180.0
=================================================
```

Most AMBER torsion potentials are not specific to all four atoms i-j-k-l, but only on the central two, j-k. Wild cards are specified with the '*' symbol as illustrated above. Again, the ordering is important. The parameters should be ordered from least specific (those containing the most wild cards) to most specific. The AMBER torsion potential can be composed of more than one Fourier component for a single torsion potential. Additional Fourier components are specified with the '&' continuation symbol as in the example above. At the moment, up to 6 Fourier components are allowed. Notice that the individual components need not be specified in any particular order. In the above example key block, there are only 5 torsional potentials defined, not 8. Two of the potentials are composed of more than one Fourier component as indicated by the '&' continuation line. Below is an example of the TORSIONS key block for the SYBYL force field. Notice that the potential types are all '2'. There are fewer parameters and no multi component potentials. Also, some potentials are defined with two or only one wild card.

```
TORSIONS
-----------------------------------------
Atoms                    pot
i - j - k - l            type  k        per      NOTES
=============================================
*     C_ar  S_3   *      2     1.0000    3.0     *
*     S_3   S_3   *      2     0.0000    2.0     EXP
```

(continues on next page)

(continued from previous page)

```
C_2    C_2    C_3    *      2      0.1260    -3.0      WHITE_77
C_3    C_2    C_3    *      2      0.1260     3.0      WHITE_77
H      C_2    C_3    *      2      0.2740     3.0      *
*      C_ar   C_ar   C_ar   2      2.3500    -2.0      *
*      C_2    C_3    C_2    2      0.1260     3.0      WHITE_77
*      C_2    C_3    C_3    2      0.1260     3.0      WHITE_77
*      C_2    C_3    H      2      0.2740     3.0      WHITE_77
*      C_3    C_3    H      2      0.3200     3.0      MC_88
O_2    C_2    C_3    C_3    2      0.7000    -3.0      JL_ES_
O_co   C_2    C_3    C_3    2      0.7000    -3.0      MAC_1
C_2    C_3    C_3    C_2    2      0.0400     3.0      WHITE_77
C_2    C_3    C_3    C_3    2      0.1260     3.0      WHITE_77
================================================
```

One can also mix different potential types within the same force field file, as illustrated below. In this example, three are three potentials. The first two are SYBYL type potentials whereas the last one is a multi component AMBER potential.

```
H   C_2    C_3    *      2   0.2740    3.0
*   C_ar   C_ar   C_ar   2   2.3500    -2.0
N   CT     C      N      1   0.4000    4      180.0
&                           1.3500    2      180.0
&                           0.7500    1      180.0
```

OUT-OF_PLANE Key block (required) This key block specifies the potential type and parameters for each kind of MM out of plane bend. This potential is sometimes referred to as the inversion potential or improper torsions (depending on the force field). The potential types currently supported are provided in the table below.

out-of-plane potential type	description	constants required (in order)
0	no potential	none
1	AMBER: $E_{inv} = K\,[1 + \cos(n\varphi - \varphi_0)]$	K, n, φ_0 (n=2, φ_0 = 180° for planar, n=3, φ_0 = 120° for tetrahedral)
2	SYBYL: $E_{oopl} = K\,d^2$ d is the distance of the plane in Å	K

An example of the key block for the AMBER type potentials is given below. It is important to realize that the atom k is the **atom k is the central atom**. (We have adopted the somewhat odd standard of AMBER in this respect).

```
OUT-OF-PLANE
-------------------------
Atoms                    pot
i - j - k - l            type  K       to     NOTES
================================================
*      *      CA     H4     1      1.10   180.0  bsd.on C6H6 nmodes
*      *      CA     H5     1      1.10   180.0  bsd.on C6H6 nmodes
*      O2     C      O2     1     10.50   180.0  JCC,7,(1986),230
*      N2     CA     N2     1     10.50   180.0  JCC,7,(1986),230
*      CT     N      CT     1      1.00   180.0  JCC,7,(1986),230
CK     CB     N*     CT     1      1.00   180.0
```

(continues on next page)

(continued from previous page)

```
=========================================
```

VAN DER WAALS Key block (required) This key block specifies the potential type and parameters for each kind of MM van der Waals interaction between two atoms. A sample key block is shown below:

```
atom(s) type  emin    rmin      alpha     NOTES
=================================
CA           -.0860   3.81600   12.00     amber95
HA           -.0150   2.91800   12.00     amber95
Ni - HA  2   -.0480   2.7       12.00     NOTE potential type
Ni - CA  D   -.0480   2.7       12.00     default potential
=================================
```

The van der Waals key block is somewhat different than the previous key blocks, because generally not every atom pair is defined with its own parameters. Rather, the parameters are assigned on a per atom basis and then special combination rules are used to construct the parameters for each atom pair combination. For this reason, the type is defined separately in the FORCE_FIELD_SETTINGS key block, although currently only the Lennard-Jones 6-12 potential is implement, although currently only the Lennard-Jones 6-12 potential is implemented

For each type of van der Waals interaction, the program first scans the key block for pair specific parameters. The three sample lines below specify pair-specific potentials. The two atom types must be separated by a hyphen with spaces between the hyphen and the atom type. Following the specification of the atom pair, the potential type is defined. If D or d is specified here, then this means to use the default potential type. Following the potential type are the parameters needed for that potential type (see above table).

```
CA - CA  1  0.0860   3.81600   12.00 amber95
Ni - HA  0
Ni - CA  D  0.0480   2.7       12.00 default potential type
```

If a pair specific parameter can't be found, then the program looks for individual atom parameters corresponding to each of the atom types in the pair. The pair specific parameters are then constructed from combination of the two individual atom parameters using the following combination rules:

VDW potential type		
1	Lennard-Jones 12-6	$D_{ij} = (D_i*D_j)^{1/2}$, $R_{ij} = (R_i+R_j)/2$
2	Exponential-6 or Buckingham	$D_{ij} = (D_i*D_j)^{1/2}$, $R_{ij} = (R_i+R_j)/2$ $\zeta_{ij} = (\zeta_i*\zeta_j)^{1/2}$
3	Purely Repulsive	$D_{ij} = (D_i*D_j)^{1/2}$, $R_{ij} = (R_i+R_j)/2$ $\zeta_{ij} = (\zeta_i*\zeta_j)^{1/2}$

When individual atom parameters are not used, no potential type is specified since the default potential type is always used. An example is given below.

```
CA 0.0860   3.81600   12.00 amber95
HA 0.0150   2.91800   12.00 amber95
```

The ability to define pair specific parameters is especially useful for those force fields that have different combination rules than used in the program. For example, Jorgensen's TIP3P water force field uses geometric averages for both

D_{ij} and R_{ij} .

MASSES & ATOM LABELS Key block (required) This key block specifies the default masses for each MM atom type and the element label for each MM atom type. The masses are not used by the engine.

A sample key block is shown below:

```
MASSES & ATOM LABELS
====================================
Ni              Ni              58.70
CM              C               12.011
CA              C               12.011
CT              C               12.011
HC              H                1.0079
HA              H                1.0079
====================================
```

The first column is the MM atom type, the second is the corresponding atom element and the third column is the mass of the atom type. The atoms do not have to be specified in any particular order.

CHARGES Key block (optional) This key block specifies the parameters for the charges on the atoms by atom type. NOTE: Charges can also be specified on a per atom basis in the `System` block of the AMS input file.

```
CHARGES
atoms       initial
label       charge
=======================
OW          -0.8
HW           0.4
=======================
```

8.3 APPLE&P parameters

APPLE&P parameters are generated for a given system and written to a jobname.ff file by AMSinput. The format of this file is similar to that used by the WMI-MD program and consists of seven sections, in this order: repulsion/dispersion (RD) types, non-default RD cross terms, bonds, valence angles (bends), torsions, impropers, and lone pairs (LPs). A line beginning with an asterisk character is a comment and is ignored. Unless otherwise specified all parameters are in the units of electron, kcal/mol, Angstrom, radian and combinations thereof. An RD type name is case-sensitive and it is checked against the atom type specified by the atom's `Forcefield.Type` suffix in the System%Atoms block.

Example force-field file for water:

```
* Repulsion/dispersion types
3
 Hw          0.00    0.0000      0.00   0.5537  0.0000
 Ow     880783.40    0.0000    851.26   0.0000  1.0425
 Lp          0.00    0.0000      0.00  -1.1074  0.0000
* Non-default cross terms
 Hw  Ow       0.00    0.0000      0.00
 Hw  Lp       0.00    0.0000      0.00
 Ow  Lp       0.00    0.0000      0.00
!
* Bond potentials
3
 999.00    0.9572    0.9572   T   Ow    Hw
 999.00    1.5140    1.5140   T   Hw    Hw
```

(continues on next page)

(continued from previous page)

```
   0.00     0.2381     0.2381    T    Ow     Lp
* Bend potentials
1
 100.00      104.52     F    Hw    Ow    Hw
* Dihedral potentials (do not make sense for water; provided for illustration)
1
 3  0.00 0.00 20.00 0.00 0.00 0.00    Hw    Ow    Ow    Hw
* Improper/out-of-plane potentials (do not make sense for water; provided for␣
↪illustration)
1
 20.00    Hw    Ow    Hw    Hw
* Lone pair definitions
1
* RD      A        B        C     type    I     J     K
  3      0.5    -0.2381   0.00     1      1     2     1
```

Section Description

`Repulsion/dispersion types` The block begins with the number of RD types followed by the specified number of lines, one per RD type. Each line begins with the RD type name followed by three dispersion parameters (A', B', and C'), atomic charge and polarizability. The (A', B', C') triplet translates to the (A, B, C, D) quartet of the dispersion part of *APPLE&P potentials* (page 23) as follows:

- if B' is zero then the dispersion potential is converted to pure Lennard-Jones with C = C' and D = A'.
- if the B' parameter non-zero then the A', B', and C' parameters translate directly to A, B, and C, while D is set to 0.00005*(12/B')^{12}.

The atomic charge specified does not include that of the LPs belonging to the atom. In the water example above, the oxygen atom's charge is carried by its lone pair Lp, so the atomic value is set to zero while the Lp's value is set to -1.1074 to compensate that of the two hydrogen atoms.

`Non-default RD cross terms` By default, the cross-term RD parameters are computed from the values in the previous section. For the pure Lennard-Jones (B' = 0), they are computed as a geometric mean of the corresponding per-type values. For the mixed Buckingham-LJ potential, the expressions are more complicated[1]. Since the A' parameter has different units in these two cases, then by default there is no dispersion interaction between them. In this case, one should specify the corresponding values in this block. Each line begins with the names of the RD types followed by a (A', B', C') triplet, similar to the previous block. The block ends with line containing a single exclamation mark.

`Bond potentials` The block begins with the number of bond types followed by the specified number of lines, one per bond type. The parameters are: the force constant, equilibrium distance, constrained distance, constraint flag (T or F), followed by the two RD type names defining the bond. The constrained distance and the constraint flag are currently not used by AMS.

`Bend potentials` The block begins with the number of bend types followed by the specified number of lines, one per bend type. The parameters are: the force constant, equilibrium angle in degrees, constraint flag (T or F), followed by the three RD type names defining the angle. The constraint flag is currently not used.

`Dihedral potentials` The block begins with the number of dihedral types followed by the specified number of lines, one per dihedral type. The parameters are: the number of terms N in the cyclic sum, up to twelve force constant (only the first N will be used), followed by the four RD type names defining the dihedral angle.

`Improper/out-of-plane potentials` The block begins with the number of improper types followed by the specified number of lines, one per improper type. Each line contains a single parameter, the force constant, followed

[1] O. Borodin, *Polarizable Force Field Development and Molecular Dynamics Simulations of Ionic Liquids*, J. Phys. Chem. B 113, 11463–11478 (2009) (https://doi.org/10.1021/jp905220k)

by the four RD types defining the OOP angle. The second RD type name corresponds to the central atom.

`Lone pair definitions` The block begins with the number of LP types followed by the specified number of lines, one per LP type. The parameters are: index of RD type corresponding to the LP in the first section, three real parameters (A, B, C), LP type, indices (I, J, K) of the RD types of the atoms used to determine coordinates of the LP(s). The number and the position of the LPs is determined by the its type while the charge and the dispersion parameters are defined by its RD type. A lone pair is a dummy atom that exist only when computing non-bonded (electrostatic and dispersion) interactions and it does not contribute to the bonded interactions in any way. It does not need to (and cannot) be specified in the Atoms block. For each valence angle (ijk) of the RD type I, J, and K, respectively, one or two LPs are created, depending on the type. The position of the LP is fixed with respect to the atoms, thus any force acting on the LP is effectively projected on those atoms.

Currently, there are two LP types defined in AMS: type 1 and type 2. Type 1 is a single LP attached to atom j, lying in the (ijk) plane at the distance abs(B) on the line connecting atom j with the point X on the (ik) line. The parameters A defines the ratio between distances R(iX) and R(ik). Thus, if A=0.5 (a typical value) the type 1 lone pair lies on the median line of the ijk triangle. If B is less than zero then the LP is inside the triangle, otherwise it's outside. For type 2, two LPs are added for each angle: one above the (ijk) plane and one below it. The parameter C defines the distance from the LP to the plane and B defines the distance between atom j and the projection of the LPs onto the (ijk) plane.

CHAPTER

NINE

POTENTIAL SHAPES

Setting the *Verbosity* (page 12) to Verbose the engine prints the potential formula and the parameters used.

The ForceField engine has a couple of potentials defined.

- Stretch: harmonic

$$V^{\text{stretch/harm}} = \frac{1}{2} f_c (r - r_0)^2$$

- Angle: harmonic and cyclic

$$V^{\text{bend/harm}} = \frac{1}{2} f_c (\phi - \phi_0)^2$$
$$V^{\text{bend/cycl}} = f_c \sum_{m=0}^{n} c_m \cos(m\phi)$$

- Torsions: cyclic, possibly linearly combined. The same torsion occurs more than once in the printed table, and the energies are added.

$$V^{\text{torsion/harm}} = \frac{1}{2} f_c (\phi - \phi_0)^2$$
$$V^{\text{torsion/cycl}} = f_c \sum_{m=0}^{n} c_m \cos(m\phi)$$

- Inversions: either angle or distance based. The angle based one depends on the order of the three atoms connected to the central atom. UFF averages over the three permutations.

$$V^{\text{inversion/harm}} = \frac{1}{2} f_c (\phi - \phi_0)^2$$
$$V^{\text{inversion/cycl}} = f_c \sum_{m=0}^{n} c_m \cos(m\phi)$$
$$V^{\text{inversion/amber}} = f_c (1 + \cos(2\phi - \phi_0))$$
$$V^{\text{inversion/dist}} = f_c \, d^2$$

- Dispersion: Lennard-Jones. Neglect up to second neighbors, possibly scale contribution from third neighbors.

$$V^{\text{dispersion/LJ}} = d \, ((x/r)^{12} - 2 * (x/r)^6)$$

- Coulomb: Neglect up to second neighbors, possibly scale contributions from third neighbors.

In general which formula is used depends on the parameter files. Note that the scaling of the third neighbors contributions is only possible when using .ff parameter files.

See also: *APPLE&P* (page 23)

CHAPTER

TEN

PERIODIC FORCEFIELD

The forcefield engine can be used to optimize the geometries of periodic systems. When specifying bonds, via the System%Bondorders key block, one should also specify the bonds that pass through a cell boundary. The GUI does this automatically. You can also simply set the key System%GuessBonds to true, and then UFF-guessed bonds will be used.

When you are having charges, Ewald summation will be used to calculate the Coulomb interaction. Currently this will be fairly slow for 1D and 2D periodic systems, as the classical Ewald trick cannot be applied.

CHAPTER

ELEVEN

REFERENCES

1. A.K. Rappe, C.J. Casewit, K.S. Colwell, W.A. Goddard III, and W. M. Skiff, *UFF, a Full Periodic Table Force Field for Molecular Dynamics Simulations*, Journal of the American Chemical Society 114, 10024-10035 (1992). (https://doi.org/10.1021/ja00051a040)
2. A.K. Rappe, W.A. Goddard III, *Charge Equilibration for Molecular Dynamics Simulations*, The Journal of Physical Chemistry 95, 3358-3363 (1991). (https://doi.org/10.1021/j100161a070)
3. M. O'Keeffe and N.E. Brese, *Atom sizes and bond lengths in Molecules and Crystals*, Journal of the American Chemical Society 113, 3226-3229 (1991). (https://doi.org/10.1021/ja00009a002)
4. S.L. Mayo, B.D. Olafson, W.A. Goddard III, *DREIDING: A Generic Force Field for Molecular Simulations*, The Journal of Physical Chemistry 94, 8897-8909 (1990). (https://doi.org/10.1021/j100389a010)
5. C.J. Casewit, K.S. Colwell, A.K. Rappe, *Applications of a Universal Force Field to Main Group Compounds*, Journal of the American Chemical Society 114, 10046-10053 (1992). (https://doi.org/10.1021/ja00051a042)
6. C.J. Casewit, K.S. Colwell, A.K. Rappe, *Application of a Universal Force Field to Organic Molecules*, Journal of the American Chemical Society 114, 10035-10046 (1992). (https://doi.org/10.1021/ja00051a041)

For calculations with the UFF4MOF parameters:

7. M.A. Addicoat, N. Vankova, I.F. Akter, and T. Heine, *An extension of the Universal Force Field to Metal-Organic Frameworks*, J. Chem. Theory Comput. 10, 880-891 (2013) (https://doi.org/10.1021/ct400952t)

For calculations with the UFF4MOFII parameters:

8. D.E. Coupry, M.A. Addicoat, and T. Heine, *An Extension of the Universal Force Field for Metal-Organic Frameworks*, J. Chem. Theory Comput. 12, 5215-5225 (2016) (https://doi.org/10.1021/acs.jctc.6b00664)

For calculations with the APPLE&P forcefield:

9. O. Borodin, *Polarizable Force Field Development and Molecular Dynamics Simulations of Ionic Liquids*, J. Phys. Chem. B 113, 11463–11478 (2009) (https://doi.org/10.1021/jp905220k)

CHAPTER

TWELVE

KEYWORDS

12.1 Links to manual entries

forcefield:

- *APPLE&P* (page 18)
- *AllowMissingParameters* (page 16)
- *AntechamberIntegration* (page 9)
- *AntechamberTask* (page 9)
- *BondsUsage* (page 12)
- *CheckDuplicateRules* (page 16)
- *DipoleConvergenceThreshold* (page 18)
- *EnergyTerms* (page 14)
- *EwaldSummation* (page 13)
- *ForceFieldFile* (page 17)
- *GuessCharges* (page 15)
- *GuessChargesConfig* (page 15)
- *LAMMPSOffload* (page 20)
- *LoadCharges* (page 16)
- *NeighborListSkin* (page 12)
- *NonBondedCutoff* (page 11)
- *Type* (page 11)
- *UFF* (page 17)
- *Verbosity* (page 12)

12.2 Summary of all keywords

12.2.1 Engine ForceField

`AllowMissingParameters`

Type Bool

Default value No

Description When parameters are not found for bonds, angles, dihedrals, or inversions, the first entry in the database will be used.

`AntechamberIntegration`

Type Bool

Default value No

GUI name Automatic atom typing

Description EXPERIMENTAL: Use the Antechamber program to automatically determine atom types for the GAFF force field. This may run a geometry optimization with MOPAC under the hood in order to determine the charges (see keyword AntechamberTask), which might not work for very large systems.

`AntechamberTask`

Type Multiple Choice

Default value GeometryOptimization

Options [GeometryOptimization, SinglePoint]

Description If antechamber is envoked to guess atomtypes and charges (GAFF force field), select the task for charge guessing with MOPAC

APPLE&P

Type Block

Description Options for the APPLE&P force field.

LongRangeCorrection

Type Bool

Default value Yes

GUI name Add long-range correction

Description Add a long-range dispersion correction to the energy and pressure for 3D-periodic systems.

This correction should be enabled only for a homogeneous liquid.

MuMu14Scaling

Type Float

Default value 1.0

GUI name Mu-Mu 3rd-neighbor scaling

Description Scaling factor for dipole-dipole interactions between atoms connected to 3rd order (via a dihedral).

QMu14Scaling

Type Float

Default value 0.2

GUI name Q-Mu 3rd-neighbor scaling

Description Scaling factor for charge-dipole interactions between atoms connected to 3rd order (via a dihedral).

QQ14Scaling

Type Float

Default value 1.0

GUI name Q-Q 3rd-neighbor scaling

Description Scaling factor for charge-charge interactions between atoms connected to 3rd order (via a dihedral).

RD14Scaling

Type Float

Default value 1.0

GUI name RD 3rd-neighbor scaling

Description Scaling factor for repulsion/dispersion interactions between atoms connected to 3rd order (via a dihedral).

BondsUsage

Type Multiple Choice

Default value Auto

Options [Input, None, Guess, Auto]

Description Controls what bonds are used by the engine. The choice auto means: guess in case there are no bonds. Guessing only happens at the first MD step, or first geometry optimization step.

CheckDuplicateRules

Type Bool

Default value Yes

Description The database could contain duplicate entries. For torsions this is a feature, and the potentials will be added. For all other terms this is no allowed, and if detected the program stops. One should fix the database or set the checking to false. As always the last entry will be used.

DipoleConvergenceThreshold

Type Float

Default value 1e-06

Unit eBohr

Description Convergence criterion for induced point dipoles, in atomic units. When the length of every atomic delta_mu vector between two iterations becomes below the tolerance, the procedure is considered converged.

DoChargeCheck

Type Bool

Default value No

Description Check that the sum of atomic (partial) charges equals the total charge of the system.

EnergyTerms

Type Block

Description expert key, that allows you to disable specific energy terms.

Angle

Type Bool

Default value Yes

Description Whether to use angle (bend) energy.

Coulomb

Type Bool

Default value Yes

Description Whether to use coulomb energy.

Dispersion

Type Bool

Default value Yes

Description Whether to use dispersion energy.

Inversion

Type Bool

Default value Yes

Description Whether to use inversion energy.

Stretch

Type Bool

Default value Yes

Description Whether to use stretch energy.

Torsion

Type Bool

Default value Yes

Description Whether to use torsion energy.

EngineConstraints

Type Bool

Default value Yes

Description Set to false to ingore constraints implied by the engine.

EwaldSummation

Type Block

Description Configures the details of the particle mesh Ewald (PME) summation of the Coulomb interaction.

Alpha

Type Float

Default value -1.0

Unit 1/Angstrom

Description This parameter shifts the workload from real space (smaller alpha) to reciprocal space (larger alpha). Using a larger [Alpha] without decreasing [GridSpacing] may increase the error in the reciprocal-space contribution. Set to zero to disable the reciprocal-space Ewald part. Negative value means the [Alpha] will be determined automatically from the [Tolerance] and [RealSpaceCutoff] values.

Enabled

Type Bool

Default value Yes

Description Set to false to use real-space pair summation instead of the Ewald, which is the default and the only option for molecules, 1D and 2D periodic systems.

GridSpacing

Type Float

Default value 0.5

Unit Angstrom

Description Grid spacing in the particle mesh Ewald method. Smaller grid spacing will make the reciprocal energy calculation more accurate but slower. Using a larger [Alpha] value may require a smaller GridSpacing to be accurate.

RealSpaceCutoff

Type Float

Default value 0.0

Unit Angstrom

Description Set the cutoff value for the real-space summation. Zero means the internal defaults will be used depending on the [Alpha] (if Alpha=0 then the cutoff will be set to 50 Bohr, otherwise to 20 Bohr).

Tolerance

Type Float

Default value 1e-10

Description Value of the error function that should be used to determine the cutoff radius for real-space Ewald summation if [Alpha] is set on input. Alternatively, if the [RealSpaceCutoff] is set but [Alpha] is not then the [Tolerance] value affects the [Alpha]. Larger values will make the real-space summation faster but less accurate.

ForceFieldFile

Type String

Default value

GUI name Force field library

Description Path to the force field parameter file

ForceFieldPatchFile

Type String

Default value

GUI name Force field patch file

Description Path to the force field patch parameter file (additional parameters, missing from main file). Cannot be used when atomtypes are guessed.

GuessCharges

Type Bool

Default value No

Description Use another engine to calculate/guess the charges to be used by the force field.

GuessChargesConfig

Type Block

Description Guess charges to be used by the forcefield

EngineType

Type String

Default value dftb

Description Engine that can calculate or guess charges

KeepAntechamberFolder

Type Bool

Default value No

Description If atom-typing is performed with antechamber, keep the folder after the call to antechamber

LAMMPSOffload

Type Block

Description Offload the calculation to LAMMPS via AMSPipe.

Enabled

Type Bool

Default value No

Description Enable offloading the force field evaluation to LAMMPS instead of handling it internally in AMS. This is currently only supported for Type=UFF.

Input

Type Block

Description Commands to be passed to LAMMPS to set up the calculation. If this is left empty, AMS will generate a set of commands to set LAMMPS up according to the settings of the ForceField engine. Any LAMMPS commands entered in this input block will be used to set LAMMPS up instead of those generated by AMS. To merge the AMS-generated lines with your customizations, include lines like 'AMS somelammpskeyword' anywhere in this block. Any such line will be replaced by the AMS-generated line for 'somelammpskeyword'. Any text after 'somelammpskeyword' will be appended to the generated line verbatim, which can be used to modify the generated command by additional options. A special line 'AMS everything' will be replaced by the entire block of AMS-generated commands, except those overriden anywhere in this input block (defined manually or inserted using 'AMS somelammpskeyword'. Any customized Input block should probably include 'AMS read_data' near or at the end to load the AMS-generated data file defining the system.

UseGPU

Type Bool

Default value No

Description Accelerate LAMMPS calculations using a GPU. Requires a LAMMPS library built with the GPU package.

UseOpenMP

Type Bool

Default value No

Description Parallelize LAMMPS calculations using OpenMP threading. Requires a LAMMPS library built with the OMP package.

WorkerCommand

Type String

Default value exec "$AMSBIN/amspython" "$AMSHOME/scripting/scm/external_engines/lmpworker.py"

Description The command to execute to run the external worker. The command is executed in a subdirectory of the results directory. The LAMMPS input commands will be passed to the worker on standard input.

`LinearizationEnergyForRepulsion`

Type Float

Default value 3.0

Unit Hartree

Description The Lennard-Jones potential becomes extremely repulsive at short distances. The distance is determined where the potential reaches this threshold, for smaller distances a linear expression is used, reducing the repulsion.

`LoadCharges`

Type Block

Description Load charges from a file to be used as forcefield charges

`File`

Type String

Description Name of the (kf) file

`Section`

Type String

Default value AMSResults

Description Section name of the kf file

`Variable`

Type String

Default value Charges

Description variable name of the kf file

`NeighborListSkin`

Type Float

Default value 2.5

Unit Angstrom

Description Thickness of the buffer region added to the NonBondedCutoff when building a neighbor list.

`NonBondedCutoff`

Type Float

Default value 15.0

Unit Angstrom

Description Distance beyond which the non-bonded pair interactions (Coulomb and Van der Waals) will be ignored.

The interactions are smoothly damped starting from 0.9*NonBondedCutoff.

Has no effect on the Coulomb term for 3D-periodic systems, as Ewald summation is used.

PairInteractionTapering

Type Multiple Choice

Default value Potential

Options [None, Potential, Force, CHARMM, CHARMM-Force]

Description Select a method for smoothing non-bonded pair interactions in the distance range between 90% and 100% of the [NonBondedCutoff] to avoid energy and force jump near the cutoff.

Potential - use a 7th order polynomial switching function that has zero 1st, 2nd and 3rd derivatives at both ends of the interval (force matches the energy derivative). Force - the same switching function is applied both to the potential and the force (so the force does not match the energy), which may break the total energy conservation during MD. CHARMM - use a different polynomial that does not have a decaying 2nd derivative at NonBondedCutoff. CHARMM-Force - use the same switching function as CHARMM but apply it both to the energy and the forces.

TaperPairInteractions

Type Bool

Default value Yes

Description Smooth non-bonded pair interactions in the distance range between 90% and 100% of the [NonBondedCutoff] to avoid energy and force jump near the cutoff. See PairInteractionTapering for more precise tuning.

Type

Type Multiple Choice

Default value UFF

Options [UFF, Amber95, GAFF, Tripos5.2, APPLE&P, UserDefined]

Description Type of force field to be used

UFF

Type Block

Description Option for the UFF force filed.

AtomTypesFile

Type String

Default value mmatomtypes_db

Description Expert option: Select the file that defines how UFF determines the atom types

Database

Type String

Default value general_db

Description Expert option: Select the file that defines the UFF parameters per atom type

ElementsFile

Type String

Default value elements_db

Description Expert option: Select the file that defines the elements known to UFF

`Library`

Type Multiple Choice

Default value UFF

Options [UFF, UFF4MOF, UFF4MOF-II]

GUI name Force field library

Description Selects the used parameter library.

`Verbosity`

Type Multiple Choice

Default value Silent

Options [Silent, Normal, Verbose, VeryVerbose]

Description Controls the verbosity of the engine.

CHAPTER

THIRTEEN

KF OUTPUT FILES

13.1 Accessing KF files

KF files are Direct Access binary files. KF stands for Keyed File: KF files are keyword oriented, which makes them easy to process by simple procedures. Internally all the data on KF files is organized into sections containing variables, so each datum on the file can be identified by the combination of section and variable.

All KF files can be opened using the KFbrowser GUI program:

```
$AMSBIN/kfbrowser path/to/ams.rkf
```

By default KFbrowser shows a just a curated summary of the results on the file, but you can make it show the raw section and variable structure by switching it to expert mode. To do this, click on **File → Expert Mode** or press **ctrl/cmd + e**.

KF files can be opened and read with Command line tools.

For working with the data from KF files, it is often useful to be able to read them from Python. Using the AMS Python Stack, this can easily be done with the AKFReader class:

```
>>> from scm.akfreader import AKFReader
>>> kf = AKFReader("path/to/ams.rkf")
>>> "Molecule%Coords" in kf
True
>>> kf.description("Molecule%Coords")
{
    '_type': 'float_array',
    '_shape': [3, 'nAtoms'],
    '_comment': 'Coordinates of the nuclei (x,y,z)',
    '_unit': 'Bohr'
}
>>> kf.read("Molecule%Coords")
array([[-11.7770694 ,  -4.19739597,   0.04934546],
       [ -9.37471321,  -2.63234227,  -0.13448698],
        ...
       [ 10.09508738,  -1.06191208,   1.45286913],
       [ 10.11689333,  -1.5080196 ,  -1.87916127]])
```

Tip: For a full overview of the available methods in AKFReader, see the AKFReader API documentation.

13.2 Sections and variables on forcefield.rkf

AMSResults Section content: Generic results of the ForceField Engine evaluation.

AMSResults%AtomicDipoleMoments

Type float_array

Description Atomic dipole moments computed by the engine.

Unit e*bohr

Shape [3, Molecule%nAtoms]

AMSResults%AtomTyping

Type subsection

Description The atom-typing for the ForceField engine.

AMSResults%AtomTyping%atomIndexToType

Type int_array

Description Array containing the index for the atom type for all the atoms in the system, i.e. iAtomType = atomIndexToType(iAtom).

Shape [Molecule%nAtoms]

AMSResults%AtomTyping%atomTypes

Type ftl_string_array

Description Strings defining the atoms types.

Shape [nAtomTypes]

AMSResults%AtomTyping%nAtomTypes

Type int

Description The number of distinct force-field atom types.

AMSResults%BondInfo

Type subsection

Description FIXME: this section should include the file shared/ArchivedBondInfo.json, but there is a problem: the variable 'BondInfo.LatticeDisplacements@dim ('BondInfo.LatticeDisplacements@dim)' is longer than 32 characters (the KF limit) and this messes up thigs. For now I'll just ignore all the variables in here…

AMSResults%Bonds

Type subsection

Description Bond info

AMSResults%Bonds%Atoms

Type archived_int_array

Description ?

AMSResults%Bonds%CellShifts

Type archived_int_array

Description ?

AMSResults%Bonds%description

Type string

Description A string containing a description of how the bond orders were calculated / where they come from

AMSResults%Bonds%hasCellShifts

Type bool

Description Whether there are cell shifts (relevant only in case of periodic boundary conditions)

AMSResults%Bonds%Index

Type archived_int_array

Description index(i) points to the first element of Atoms, Orders, and CellShifts belonging to bonds from atom 'i'. Index(1) is always 1, Index(nAtoms+1) is always nBonds + 1

AMSResults%Bonds%Orders

Type archived_float_array

Description The bond orders.

AMSResults%BulkModulus

Type float

Description The Bulk modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit hartree/bohr^3

AMSResults%Charges

Type float_array

Description Net atomic charges as computed by the engine (for example, the Charges for a water molecule might be [-0.6, 0.3, 0.3]). The method used to compute these atomic charges depends on the engine.

Unit e

Shape [Molecule%nAtoms]

AMSResults%Config

Type subsection

Description Configuration of the ForceField engine.

AMSResults%DipoleGradients

Type float_array

Description Derivative of the dipole moment with respect to nuclear displacements.

Shape [3, 3, Molecule%nAtoms]

AMSResults%DipoleMoment

Type float_array

Description Dipole moment vector (x,y,z)

Unit e*bohr

Shape [3]

AMSResults%ElasticTensor

Type float_array

Description The elastic tensor in Voigt notation (6x6 matrix for 3D periodic systems, 3x3 matrix for 2D periodic systems, 1x1 matrix for 1D periodic systems).

Unit hartree/bohr^nLatticeVectors

Shape [:, :]

AMSResults%Energy

Type float

Description The energy computed by the engine.

Unit hartree

AMSResults%Gradients

Type float_array

Description The nuclear gradients.

Unit hartree/bohr

Shape [3, Molecule%nAtoms]

AMSResults%Hessian

Type float_array

Description The Hessian matrix

Unit hartree/bohr^2

Shape [3*Molecule%nAtoms, 3*Molecule%nAtoms]

AMSResults%Molecules

Type subsection

Description Molecules

AMSResults%Molecules%AtCount

Type archived_int_array

Description shape=(nMolType), Summary: number of atoms per formula.

AMSResults%Molecules%Atoms

Type archived_int_array

Description shape=(nAtoms), atoms(index(i):index(i+1)-1) = atom indices of molecule i

AMSResults%Molecules%Count

Type archived_int_array

Description Mol count per formula.

AMSResults%Molecules%Formulas

Type string

Description Summary: unique molecule formulas

AMSResults%Molecules%Index

Type archived_int_array

Description shape=(nMol+1), index(i) = index of the first atom of molecule i in array atoms(:)

AMSResults%Molecules%Type

Type archived_int_array

Description shape=(nMol), type of the molecule, reference to the summary arrays below

AMSResults%PESPointCharacter

Type string

Description The character of a PES point.

Possible values ['local minimum', 'transition state', 'stationary point with >1 negative frequencies', 'non-stationary point']

AMSResults%PoissonRatio

Type float

Description The Poisson ratio

AMSResults%ShearModulus

Type float

Description The Shear modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit hartree/bohr^3

AMSResults%StressTensor

Type float_array

Description The clamped-ion stress tensor in Cartesian notation.

Unit hartree/bohr^nLatticeVectors

Shape [:, :]

AMSResults%YoungModulus

Type float

Description The Young modulus (conversion factor from hartree/bohr^3 to GPa: 29421.026)

Unit hartree/bohr^3

BZcell(primitive cell) Section content: The Brillouin zone of the primitive cell.

BZcell(primitive cell)%boundaries

Type float_array

Description Normal vectors for the boundaries.

Shape [ndim, nboundaries]

BZcell(primitive cell)%distances

Type float_array

Description Distance to the boundaries.

Shape [nboundaries]

BZcell(primitive cell)%idVerticesPerBound

Type int_array

Description The indices of the vertices per bound.

Shape [nvertices, nboundaries]

BZcell(primitive cell)%latticeVectors

Type float_array

Description The lattice vectors.

Shape [3, :]

BZcell(primitive cell)%nboundaries

Type int

Description The nr. of boundaries for the cell.

BZcell(primitive cell)%ndim

Type int

Description The nr. of lattice vectors spanning the Wigner-Seitz cell.

BZcell(primitive cell)%numVerticesPerBound

Type int_array

Description The nr. of vertices per bound.

Shape [nboundaries]

BZcell(primitive cell)%nvertices

Type int

Description The nr. of vertices of the cell.

BZcell(primitive cell)%vertices

Type float_array

Description The vertices of the bounds.

Unit a.u.

Shape [ndim, nvertices]

DOS_Phonons **Section content:** Phonon Density of States

DOS_Phonons%DeltaE

Type float

Description The energy difference bewteen sampled DOS energies. When there is no DOS at all a certain energy range can be skipped.

Unit hartree

DOS_Phonons%Energies

Type float_array

Description The energies at which the DOS is sampled.

Unit hartree

Shape [nEnergies]

DOS_Phonons%Fermi Energy

Type float

Description The fermi energy.

Unit hartree

DOS_Phonons%IntegrateDeltaE

Type bool

Description If enabled it means that the DOS is integrated over intervals of DeltaE. Sharp delta function like peaks cannot be missed this way.

DOS_Phonons%nEnergies

Type int

Description The nr. of energies to use to sample the DOS.

DOS_Phonons%nSpin

Type int

Description The number of spin components for the DOS.

Possible values [1, 2]

DOS_Phonons%Total DOS

Type float_array

Description The total DOS.

Shape [nEnergies, nSpin]

General Section content: General information about the ForceField calculation.

General%account

Type string

Description Name of the account from the license

General%engine input

Type string

Description The text input of the engine.

General%engine messages

Type string

Description Message from the engine. In case the engine fails to solves, this may contains extra information on why.

General%file-ident

Type string

Description The file type identifier, e.g. RKF, RUNKF, TAPE21…

General%jobid

Type int

Description Unique identifier for the job.

General%program

Type string

Description The name of the program/engine that generated this kf file.

General%release

Type string

Description The version of the program that generated this kf file (including svn revision number and date).

General%termination status

Type string

Description The termination status. Possible values: 'NORMAL TERMINATION', 'NORMAL TERMINATION with warnings', 'NORMAL TERMINATION with errors', 'ERROR', 'IN PROGRESS'.

General%title

Type string

Description Title of the calculation.

General%uid

Type string

Description SCM User ID

General%version

Type int

Description Version number?

KFDefinitions **Section content:** The definitions of the data on this file

KFDefinitions%json

Type string

Description The definitions of the data on this file in json.

kspace(primitive cell) **Section content:** should not be here!!!

kspace(primitive cell)%avec

Type float_array

Description The lattice stored as a 3xnLatticeVectors matrix. Only the ndimk,ndimk part has meaning.

Unit bohr

Shape [3, :]

kspace(primitive cell)%bvec

Type float_array

Description The inverse lattice stored as a 3x3 matrix. Only the ndimk,ndimk part has meaning.

Unit 1/bohr

Shape [ndim, ndim]

kspace(primitive cell)%kt

Type int

Description The total number of k-points used by the k-space to sample the unique wedge of the Brillouin zone.

kspace(primitive cell)%kuniqu

Type int

Description The number of symmetry unique k-points where an explicit diagonalization is needed. Smaller or equal to kt.

kspace(primitive cell)%ndim

Type int

Description The nr. of lattice vectors.

kspace(primitive cell)%ndimk

Type int

Description The nr. of dimensions used in the k-space integration.

kspace(primitive cell)%xyzpt

Type float_array

Description The coordinates of the k-points.

Unit 1/bohr

Shape [ndimk, kt]

Low Frequency Correction **Section content:** Configuration for the Head-Gordon Dampener-powered Free Rotor Interpolation.

Low Frequency Correction%Alpha

Type float

Description Exponent term for the Head-Gordon dampener.

Low Frequency Correction%Frequency

Type float

Description Frequency around which interpolation happens, in 1/cm.

Low Frequency Correction%Moment of Inertia

Type float

Description Used to make sure frequencies of less than ca. 1 1/cm don't overestimate entropy, in kg m^2.

Mobile Block Hessian **Section content:** Mobile Block Hessian.

Mobile Block Hessian%Coordinates Internal

Type float_array

Description ?

Mobile Block Hessian%Free Atom Indexes Input

Type int_array

Description ?

Mobile Block Hessian%Frequencies in atomic units

Type float_array

Description ?

Mobile Block Hessian%Frequencies in wavenumbers

Type float_array

Description ?

Mobile Block Hessian%Input Cartesian Normal Modes

Type float_array

Description ?

Mobile Block Hessian%Input Indexes of Block #

Type int_array

Description ?

Mobile Block Hessian%Intensities in km/mol

Type float_array

Description ?

Mobile Block Hessian%MBH Curvatures

Type float_array

Description ?

Mobile Block Hessian%Number of Blocks

Type int

Description Number of blocks.

Mobile Block Hessian%Sizes of Blocks

Type int_array

Description Sizes of the blocks.

Shape [Number of Blocks]

Molecule Section content: The input molecule of the calculation.

Molecule%AtomicNumbers

Type int_array

Description Atomic number 'Z' of the atoms in the system

Shape [nAtoms]

Molecule%AtomMasses

Type float_array

Description Masses of the atoms

Unit a.u.

Values range [0, '\infinity']

Shape [nAtoms]

Molecule%AtomSymbols

Type string

Description The atom's symbols (e.g. 'C' for carbon)

Shape [nAtoms]

Molecule%bondOrders

Type float_array

Description The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

Molecule%Charge

Type float

Description Net charge of the system

Unit e

Molecule%Coords

Type float_array

Description Coordinates of the nuclei (x,y,z)

Unit bohr

Shape [3, nAtoms]

Molecule%eeAttachTo

Type int_array

Description A multipole may be attached to an atom. This influences the energy gradient.

Molecule%eeChargeWidth

Type float

Description If charge broadening was used for external charges, this represents the width of the charge distribution.

Molecule%eeEField

Type float_array

Description The external homogeneous electric field.

Unit hartree/(e*bohr)

Shape [3]

Molecule%eeLatticeVectors

Type float_array

Description The lattice vectors used for the external point- or multipole- charges.

Unit bohr

Shape [3, eeNLatticeVectors]

Molecule%eeMulti

Type float_array

Description The values of the external point- or multipole- charges.

Unit a.u.

Shape [eeNZlm, eeNMulti]

Molecule%eeNLatticeVectors

Type int

Description The number of lattice vectors for the external point- or multipole- charges.

Molecule%eeNMulti

Type int

Description The number of external point- or multipole- charges.

Molecule%eeNZlm

Type int

Description When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

Molecule%eeUseChargeBroadening

Type bool

Description Whether or not the external charges are point-like or broadened.

Molecule%eeXYZ

Type float_array

Description The position of the external point- or multipole- charges.

Unit bohr

Shape [3, eeNMulti]

Molecule%EngineAtomicInfo

Type string_fixed_length

Description Atom-wise info possibly used by the engine.

Molecule%fromAtoms

Type int_array

Description Index of the first atom in a bond. See the bondOrders array

Molecule%latticeDisplacements

Type int_array

Description The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

Molecule%LatticeVectors

Type float_array

Description Lattice vectors

Unit bohr

Shape [3, nLatticeVectors]

Molecule%nAtoms

Type int

Description The number of atoms in the system

Molecule%nAtomsTypes

Type int

Description The number different of atoms types

Molecule%nLatticeVectors

Type int

Description Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values [0, 1, 2, 3]

Molecule%toAtoms

Type int_array

Description Index of the second atom in a bond. See the bondOrders array

MoleculeSuperCell Section content: The system used for the numerical phonon super cell calculation.

MoleculeSuperCell%AtomicNumbers

Type int_array

Description Atomic number 'Z' of the atoms in the system

Shape [nAtoms]

MoleculeSuperCell%AtomMasses

Type float_array

Description Masses of the atoms

Unit a.u.

Values range [0, '\infinity']

Shape [nAtoms]

MoleculeSuperCell%AtomSymbols

Type string

Description The atom's symbols (e.g. 'C' for carbon)

Shape [nAtoms]

MoleculeSuperCell%bondOrders

Type float_array

Description The bond orders for the bonds in the system. The indices of the two atoms participating in the bond are defined in the arrays 'fromAtoms' and 'toAtoms'. e.g. bondOrders[1]=2, fromAtoms[1]=4 and toAtoms[1]=7 means that there is a double bond between atom number 4 and atom number 7

MoleculeSuperCell%Charge

Type float

Description Net charge of the system

Unit e

MoleculeSuperCell%Coords

Type float_array

Description Coordinates of the nuclei (x,y,z)

Unit bohr

Shape [3, nAtoms]

MoleculeSuperCell%eeAttachTo

Type int_array

Description A multipole may be attached to an atom. This influences the energy gradient.

MoleculeSuperCell%eeChargeWidth

Type float

Description If charge broadening was used for external charges, this represents the width of the charge distribution.

MoleculeSuperCell%eeEField

Type float_array

Description The external homogeneous electric field.

Unit hartree/(e*bohr)

Shape [3]

MoleculeSuperCell%eeLatticeVectors

Type float_array

Description The lattice vectors used for the external point- or multipole- charges.

Unit bohr

Shape [3, eeNLatticeVectors]

MoleculeSuperCell%eeMulti

Type float_array

Description The values of the external point- or multipole- charges.

Unit a.u.

Shape [eeNZlm, eeNMulti]

MoleculeSuperCell%eeNLatticeVectors

Type int

Description The number of lattice vectors for the external point- or multipole- charges.

MoleculeSuperCell%eeNMulti

Type int

Description The number of external point- or multipole- charges.

MoleculeSuperCell%eeNZlm

Type int

Description When external point- or multipole- charges are used, this represents the number of spherical harmonic components. E.g. if only point charges were used, eeNZlm=1 (s-component only). If point charges and dipole moments were used, eeNZlm=4 (s, px, py and pz).

MoleculeSuperCell%eeUseChargeBroadening

Type bool

Description Whether or not the external charges are point-like or broadened.

MoleculeSuperCell%eeXYZ

Type float_array

Description The position of the external point- or multipole- charges.

Unit bohr

Shape [3, eeNMulti]

MoleculeSuperCell%EngineAtomicInfo

Type string_fixed_length

Description Atom-wise info possibly used by the engine.

MoleculeSuperCell%fromAtoms

Type int_array

Description Index of the first atom in a bond. See the bondOrders array

MoleculeSuperCell%latticeDisplacements

Type int_array

Description The integer lattice translations for the bonds defined in the variables bondOrders, fromAtoms and toAtoms.

MoleculeSuperCell%LatticeVectors

Type float_array

Description Lattice vectors

Unit bohr

Shape [3, nLatticeVectors]

MoleculeSuperCell%nAtoms

Type int

Description The number of atoms in the system

MoleculeSuperCell%nAtomsTypes

Type int

Description The number different of atoms types

MoleculeSuperCell%nLatticeVectors

Type int

Description Number of lattice vectors (i.e. number of periodic boundary conditions)

Possible values [0, 1, 2, 3]

MoleculeSuperCell%toAtoms

Type int_array

Description Index of the second atom in a bond. See the bondOrders array

phonon_curves **Section content:** Phonon dispersion curves.

phonon_curves%brav_type

Type string

Description Type of the lattice.

phonon_curves%Edge_#_bands

Type float_array

Description The band energies

Shape [nBands, nSpin, :]

phonon_curves%Edge_#_direction

Type float_array

Description Direction vector.

Shape [nDimK]

phonon_curves%Edge_#_kPoints

Type float_array

Description Coordinates for points along the edge.

Shape [nDimK, :]

phonon_curves%Edge_#_labels

Type lchar_string_array

Description Labels for begin and end point of the edge.

Shape [2]

phonon_curves%Edge_#_lGamma

Type bool

Description Is gamma point?

phonon_curves%Edge_#_nKPoints

Type int

Description The nr. of k points along the edge.

phonon_curves%Edge_#_vertices

Type float_array

Description Begin and end point of the edge.

Shape [nDimK, 2]

phonon_curves%Edge_#_xFor1DPlotting

Type float_array

Description x Coordinate for points along the edge.

Shape [:]

phonon_curves%indexLowestBand

Type int

Description ?

phonon_curves%nBands

Type int

Description Number of bands.

phonon_curves%nBas

Type int

Description Number of basis functions.

phonon_curves%nDimK

Type int

Description Dimenion of the reciprocal space.

phonon_curves%nEdges

Type int

Description The number of edges. An edge is a line-segment through k-space. It has a begin and end point and possilby points in between.

phonon_curves%nEdgesInPath

Type int

Description A path is built up from a number of edges.

phonon_curves%nSpin

Type int

Description Number of spin components.

Possible values [1, 2]

phonon_curves%path

Type int_array

Description If the (edge) index is negative it means that the vertices of the edge abs(index) are swapped e.g. path = (1,2,3,0,-3,-2,-1) goes though edges 1,2,3, then there's a jump, and then it goes back.

Shape [nEdgesInPath]

phonon_curves%path_type

Type string

Description ?

Phonons Section content: Information on the numerical phonons (super cell) setup. NB: the reciprocal cell of the super cell is smaller than the reciprocal primitive cell.

Phonons%Modes

Type float_array

Description The normal modes with the translational symmetry of the super cell.

Shape [3, nAtoms, 3, NumAtomsPrim, nK]

Phonons%nAtoms

Type int

Description Number of atoms in the super cell.

Phonons%nK

Type int

Description Number of gamma-points (of the super cell) that fit into the primitive reciprocal cell.

Phonons%NumAtomsPrim

Type int

Description Number of atoms in the primitive cell.

Phonons%xyzKSuper

Type float_array

Description The coordinates of the gamma points that fit into the primitive reciprocal cell.

Shape [3, nK]

Properties Section content: Generic container for properties. The program band uses different rules for Types and Subtypes.

Properties%nEntries

Type int

Description Number of properties.

Properties%Subtype(#)

Type string_fixed_length

Description Extra detail about the property. For a charge propety this could be Mulliken.

Properties%Type(#)

Type string

Description Type of the property, like energy, gradients, charges, etc.

Properties%Value(#)

Type float_array

Description The value(s) of the propety.

Thermodynamics Section content: Thermodynamic properties computed from normal modes.

Thermodynamics%Enthalpy

Type float_array

Description Enthalpy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Entropy rotational

Type float_array

Description Rotational contribution to the entropy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Entropy total

Type float_array

Description Total entropy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Entropy translational

Type float_array

Description Translational contribution to the entropy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Entropy vibrational

Type float_array

Description Vibrational contribution to the entropy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Gibbs free Energy

Type float_array

Description Gibbs free energy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Heat Capacity rotational

Type float_array

Description Rotational contribution to the heat capacity.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Heat Capacity total

Type float_array

Description Total heat capacity.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Heat Capacity translational

Type float_array

Description Translational contribution to the heat capacity.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Heat Capacity vibrational

Type float_array

Description Vibrational contribution to the heat capacity.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Inertia direction vectors

Type float_array

Description Inertia direction vectors.

Shape [3, 3]

Thermodynamics%Internal Energy rotational

Type float_array

Description Rotational contribution to the internal energy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Internal Energy total

Type float_array

Description Total internal energy.

Unit a.u.

Thermodynamics%Internal Energy translational

Type float_array

Description Translational contribution to the internal energy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Internal Energy vibrational

Type float_array

Description Vibrational contribution to the internal energy.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%lowFreqEntropy

Type float_array

Description Entropy contributions from low frequencies (see 'lowFrequencies').

Unit a.u.

Shape [nLowFrequencies]

Thermodynamics%lowFreqHeatCapacity

Type float_array

Description Heat capacity contributions from low frequencies (see 'lowFrequencies').

Unit a.u.

Shape [nLowFrequencies]

Thermodynamics%lowFreqInternalEnergy

Type float_array

Description Internal energy contributions from low frequencies (see 'lowFrequencies').

Unit a.u.

Shape [nLowFrequencies]

Thermodynamics%lowFrequencies

Type float_array

Description Frequencies below 20 cm^-1 (contributions from frequencies below 20 cm^-1 are not included in vibrational sums, and are saved separately to 'lowFreqEntropy', 'lowFreqInternalEnergy' and 'lowFreqInternalEnergy'). Note: this does not apply to RRHO-corrected quantities.

Unit cm^-1

Shape [nLowFrequencies]

Thermodynamics%Moments of inertia

Type float_array

Description Moments of inertia.

Unit a.u.

Shape [3]

Thermodynamics%nLowFrequencies

Type int

Description Number of elements in the array lowFrequencies.

Thermodynamics%nTemperatures

Type int

Description Number of temperatures.

Thermodynamics%Pressure

Type float

Description Pressure used.

Unit atm

Thermodynamics%RRHOCorrectedHeatCapacity

Type float_array

Description Heat capacity T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%RRHOCorrectedInternalEnergy

Type float_array

Description Internal energy T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%RRHOCorrectedTS

Type float_array

Description T*S corrected using the 'low vibrational frequency free rotor interpolation corrections'.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%Temperature

Type float_array

Description List of temperatures at which properties are calculated.

Unit a.u.

Shape [nTemperatures]

Thermodynamics%TS

Type float_array

Description T*S, i.e. temperature times entropy.

Unit a.u.

Shape [nTemperatures]

Vibrations **Section content:** Information related to molecular vibrations.

Vibrations%ExcitedStateLifetime

Type float

Description Raman excited state lifetime.

Unit hartree

Vibrations%ForceConstants

Type float_array

Description The force constants of the vibrations.

Unit hartree/bohr^2

Shape [nNormalModes]

Vibrations%Frequencies[cm-1]

Type float_array

Description The vibrational frequencies of the normal modes.

Unit cm^-1

Shape [nNormalModes]

Vibrations%Intensities[km/mol]

Type float_array

Description The intensity of the normal modes.

Unit km/mol

Shape [nNormalModes]

Vibrations%IrReps

Type lchar_string_array

Description Symmetry symbol of the normal mode.

Shape [nNormalModes]

Vibrations%ModesNorm2

Type float_array

Description Norms of the rigid motions.

Shape [nNormalModes+nRigidModes]

Vibrations%ModesNorm2*

Type float_array

Description Norms of the rigid motions (for a given irrep…?).

Shape [nNormalModes+nRigidModes]

Vibrations%nNormalModes

Type int

Description Number of normal modes.

Vibrations%NoWeightNormalMode(#)

Type float_array

Description ?.

Shape [3, Molecule%nAtoms]

Vibrations%NoWeightRigidMode(#)

Type float_array

Description ?

Shape [3, Molecule%nAtoms]

Vibrations%nRigidModes

Type int

Description Number of rigid modes.

Vibrations%nSemiRigidModes

Type int

Description Number of semi-rigid modes.

Vibrations%PVDOS

Type float_array

Description Partial vibrational density of states.

Values range [0.0, 1.0]

Shape [nNormalModes, Molecule%nAtoms]

Vibrations%RamanDepolRatioLin

Type float_array

Description Raman depol ratio (lin).

Shape [nNormalModes]

Vibrations%RamanDepolRatioNat

Type float_array

Description Raman depol ratio (nat).

Shape [nNormalModes]

Vibrations%RamanIncidentFreq

Type float

Description Raman incident light frequency.

Unit hartree

Vibrations%RamanIntens[A^4/amu]

Type float_array

Description Raman intensities

Unit A^4/amu

Shape [nNormalModes]

Vibrations%ReducedMasses

Type float_array

Description The reduced masses of the normal modes.

Unit a.u.

Values range [0, '\infinity']

Shape [nNormalModes]

Vibrations%RotationalStrength

Type float_array

Description The rotational strength of the normal modes.

Shape [nNormalModes]

Vibrations%TransformationMatrix

Type float_array

Description ?

Shape [3, Molecule%nAtoms, nNormalModes]

Vibrations%VROACIDBackward

Type float_array

Description VROA Circular Intensity Differential: Backward scattering.

Unit 10¯3

Shape [nNormalModes]

Vibrations%VROACIDDePolarized

Type float_array

Description VROA Circular Intensity Differential: Depolarized scattering.

Unit 10¯3

Shape [nNormalModes]

Vibrations%VROACIDForward

Type float_array

Description VROA Circular Intensity Differential: Forward scattering.

Unit 10¯3

Shape [nNormalModes]

Vibrations%VROACIDPolarized

Type float_array

Description VROA Circular Intensity Differential: Polarized scattering.

Unit 10¯3

Shape [nNormalModes]

Vibrations%VROADeltaBackward

Type float_array

Description VROA Intensity: Backward scattering.

Unit 10¯3 A^4/amu

Shape [nNormalModes]

Vibrations%VROADeltaDePolarized

Type float_array

Description VROA Intensity: Depolarized scattering.

Unit 10¯3 A^4/amu

Shape [nNormalModes]

Vibrations%VROADeltaForward

Type float_array

Description VROA Intensity: Forward scattering.

Unit 10¯3 A^4/amu

Shape [nNormalModes]

Vibrations%VROADeltaPolarized

Type float_array

Description VROA Intensity: Polarized scattering.

Unit 10^-3 A^4/amu

Shape [nNormalModes]

`Vibrations%ZeroPointEnergy`

Type float

Description Vibrational zero-point energy.

Unit hartree

INDEX

A

antechamber, 9
appleandp_packmol() (*in module scm.appleandp*), 27